

Practical Corpus Linguistics

Practical Corpus Linguistics

An Introduction to Corpus-Based
Language Analysis

Martin Weisser

WILEY Blackwell

This edition first published 2016
© 2016 John Wiley & Sons, Inc

Registered Office

John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, UK

Editorial Offices

350 Main Street, Malden, MA 02148-5020, USA

9600 Garsington Road, Oxford, OX4 2DQ, UK

The Atrium, Southern Gate, Chichester, West Sussex, PO19 8SQ, UK

For details of our global editorial offices, for customer services, and for information about how to apply for permission to reuse the copyright material in this book please see our website at www.wiley.com/wiley-blackwell.

The right of Martin Weisser to be identified as the author of this work has been asserted in accordance with the UK Copyright, Designs and Patents Act 1988.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, except as permitted by the UK Copyright, Designs and Patents Act 1988, without the prior permission of the publisher.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

Designations used by companies to distinguish their products are often claimed as trademarks. All brand names and product names used in this book are trade names, service marks, trademarks or registered trademarks of their respective owners. The publisher is not associated with any product or vendor mentioned in this book.

Limit of Liability/Disclaimer of Warranty: While the publisher and authors have used their best efforts in preparing this book, they make no representations or warranties with respect to the accuracy or completeness of the contents of this book and specifically disclaim any implied warranties of merchantability or fitness for a particular purpose. It is sold on the understanding that the publisher is not engaged in rendering professional services and neither the publisher nor the author shall be liable for damages arising herefrom. If professional advice or other expert assistance is required, the services of a competent professional should be sought.

Library of Congress Cataloging-in-Publication Data

Weisser, Martin, author.

Practical corpus linguistics : an introduction to corpus-based language analysis / Martin Weisser. – First edition.

pages cm

Includes bibliographical references and index.

ISBN 978-1-118-83187-8 (hardback) – ISBN 978-1-118-83188-5 (paper) 1. Linguistic analysis (Linguistics)–Databases. 2. Linguistic analysis (Linguistics)–Software. 3. Corpora (Linguistics)–Methodology. 4. Corpora (Linguistics)–Technological innovations. 5. Computational linguistics–Methodology. 6. Computer network resources–Evaluation. 7. Citation of electronic information resources. I. Title.

P128.D37W45 2016

410.1'88–dc23

2015023709

A catalogue record for this book is available from the British Library.

Cover image: © Martin Weisser

Set in 10.5/13pt Galliard by Aptara Inc., New Delhi, India

To Ye & Emma,
who've had to suffer
from an undue lack of attention
throughout the final months
of writing this book

Contents

List of Figures	xiii
List of Tables	xv
Acknowledgements	xvii
1 Introduction	1
1.1 Linguistic Data Analysis	3
1.1.1 What's data?	3
1.1.2 Forms of data	3
1.1.3 Collecting and analysing data	7
1.2 Outline of the Book	8
1.3 Conventions Used in this Book	10
1.4 A Note for Teachers	11
1.5 Online Resources	11
2 What's Out There?	13
2.1 What's a Corpus?	13
2.2 Corpus Formats	13
2.3 Synchronic vs. Diachronic Corpora	15
2.3.1 'Early' synchronic corpora	15
2.3.2 Mixed corpora	18
2.3.3 Examples of diachronic corpora	20
2.4 General vs. Specific Corpora	21
2.4.1 Examples of specific corpora	22
2.5 Static Versus Dynamic Corpora	25
2.6 Other Sources for Corpora	26

	Solutions to/Comments on the Exercises	26
	Note	28
	Sources and Further Reading	28
3	Understanding Corpus Design	29
3.1	Food for Thought – General Issues in Corpus Design	29
3.1.1	Sampling	30
3.1.2	Size	31
3.1.3	Balance and representativeness	32
3.1.4	Legal issues	32
3.2	What’s in a Text? – Understanding Document Structure	33
3.2.1	Headers, ‘footers’ and meta-data	34
3.2.2	The structure of the (text) body	36
3.2.3	What’s (in) an electronic text? – understanding file formats and their properties	37
3.3	Understanding Encoding: Character Sets, File Size, etc.	38
3.3.1	ASCII and legacy encodings	38
3.3.2	Unicode	39
3.3.3	File sizes	40
	Solutions to/Comments on the Exercises	41
	Sources and Further Reading	42
4	Finding and Preparing Your Data	43
4.1	Finding Suitable Materials for Analysis	44
4.1.1	Retrieving data from text archives	44
4.1.2	Obtaining materials from Project Gutenberg	44
4.1.3	Obtaining materials from the Oxford Text Archive	45
4.2	Collecting Written Materials Yourself (‘Web as Corpus’)	46
4.2.1	A brief note on plain-text editors	46
4.2.2	Browser text export	48
4.2.3	Browser HTML export	49
4.2.4	Getting web data using ICEweb	50
4.2.5	Downloading other types of files	52
4.3	Collecting Spoken Data	53
4.4	Preparing Written Data for Analysis	56
4.4.1	‘Cleaning up’ your data	56
4.4.2	Extracting text from proprietary document formats	58
4.4.3	Removing unnecessary header and ‘footer’ information	58
4.4.4	Documenting what you’ve collected	59
4.4.5	Preparing your data for distribution or archiving	60
	Solutions to/Comments on the Exercises	62
	Sources and Further Reading	66
5	Concordancing	67
5.1	What’s Concordancing?	67

5.2	Concordancing with AntConc	69
5.2.1	Sorting results	74
5.2.2	Saving, pruning and reusing your results	75
	Solutions to/Comments on the Exercises	78
	Sources and Further Reading	81
6	Regular Expressions	82
6.1	Character Classes	84
6.2	Negative Character Classes	86
6.3	Quantification	86
6.4	Anchoring, Grouping and Alternation	87
6.4.1	Anchoring	87
6.4.2	Grouping and alternation	88
6.4.3	Quoting and using special characters	90
6.4.4	Constraining the context further	91
6.5	Further Exercises	92
	Solutions to/Comments on the Exercises	93
	Sources and Further Reading	100
7	Understanding Part-of-Speech Tagging and Its Uses	101
7.1	A Brief Introduction to (Morpho-Syntactic) Tagsets	103
7.2	Tagging Your Own Data	109
	Solutions to/Comments on the Exercises	113
	Sources and Further Reading	120
8	Using Online Interfaces to Query Mega Corpora	121
8.1	Searching the BNC with BNCweb	122
8.1.1	What is BNCweb?	122
8.1.2	Basic standard queries	123
8.1.3	Navigating through and exploring search results	124
8.1.4	More advanced standard query options	126
8.1.5	Wildcards	126
8.1.6	Word and phrase alternation	128
8.1.7	Restricting searches through PoS tags	129
8.1.8	Headword and lemma queries	131
8.2	Exploring COCA through the BYU Web-Interface	132
8.2.1	The basic syntax	133
8.2.2	Comparing corpora in the BYU interface	135
	Solutions to/Comments on the Exercises	137
	Sources and Further Reading	145
9	Basic Frequency Analysis – or What Can (Single) Words Tell Us About Texts?	146
9.1	Understanding Basic Units in Texts	146
9.1.1	What’s a word?	147
9.1.2	Types and tokens	149

9.2	Word (Frequency) Lists in AntConc	151
9.2.1	Stop words – good or bad?	156
9.2.2	Defining and using stop words in AntConc	158
9.3	Word Lists in BNCweb	160
9.3.1	Standard options	160
9.3.2	Investigating subcorpora	162
9.3.3	Keyword lists	169
9.4	Keyword Lists in AntConc and BNCweb	169
9.4.1	Keyword lists in AntConc	169
9.4.2	Keyword lists in BNCweb	172
9.5	Comparing and Reporting Frequency Counts	175
9.6	Investigating Genre-Specific Distributions in COCA	178
	Solutions to/Comments on the Exercises	179
	Sources and Further Reading	192
10	Exploring Words in Context	193
10.1	Understanding Extended Units of Text	194
10.2	Text Segmentation	195
10.3	N-Grams, Word Clusters and Lexical Bundles	196
10.4	Exploring (Relatively) Fixed Sequences in BNCweb	198
10.5	Simple, Sequential Collocations and Colligations	198
10.5.1	‘Simple’ collocations	198
10.5.2	Colligations	200
10.5.3	Contextually constrained and proximity searches	201
10.6	Exploring Colligations in COCA	202
10.7	N-grams and Clusters in AntConc	205
10.8	Investigating Collocations Based on Statistical Measures in AntConc, BNCweb and COCA	207
10.8.1	Calculating collocations	207
10.8.2	Computing collocations in AntConc	209
10.8.3	Computing collocations in BNCweb	210
10.8.4	Computing collocations in COCA	211
	Solutions to/Comments on the Exercises	212
	Sources and Further Reading	226
11	Understanding Markup and Annotation	227
11.1	From SGML to XML – A Brief Timeline	229
11.2	XML for Linguistics	230
11.2.1	Why bother?	230
11.2.2	What does markup/annotation look like?	230
11.2.3	The ‘history’ and development of (linguistic) markup	232
11.2.4	XML and style sheets	234
11.3	‘Simple XML’ for Linguistic Annotation	236
11.4	Colour Coding and Visualisation	240
11.5	More Complex Forms of Annotation	246

Solutions to/Comments on the Exercises	248
Sources and Further Reading	253
12 Conclusion and Further Perspectives	254
Appendix A: The CLAWS C5 Tagset	259
Appendix B: The Annotated Dialogue File	261
Appendix C: The CSS Style Sheet	269
Glossary	271
References	277
Index	283

List of Figures

3.1	Illustration of basic document structure	34
4.1	The ICEweb interface	50
5.1	Example of a KWIC concordance output	69
5.2	The AntConc startup screen	70
5.3	AntConc file opening options	71
5.4	AntConc file settings	71
5.5	AntConc ‘Corpus Files’ window (two files loaded)	72
5.6	AntConc ‘Search Term’ and search options	72
5.7	AntConc results for <i>round</i> in two novels by Jane Austen	73
5.8	AntConc ‘Search Window Size’ options	73
5.9	AntConc ‘Kwic Sort’ options	74
6.1	Sample paragraph for practising and understanding regex patterns	84
7.1	Sample output of the Simple PoS Tagger	110
8.1	The BNCweb startup screen	122
8.2	Results for simple search for <i>assume</i>	123
8.3	BNCweb query follow-on options	125
8.4	The basic COCA interface	133
8.5	Display of antonyms <i>thoughtful</i> and <i>thoughtless</i> as alternatives	134
8.6	Side-by-side comparison for the lemma of <i>movie</i> in the COCA and BNC	136
9.1	Output of a basic frequency list in AntConc	152
9.2	Token (word) (re-)definition in AntConc	154
9.3	AntConc Word List preferences	159
9.4	BNCweb frequency list selection options	161
9.5	Options for defining subcorpora in BNCweb	162
9.6	Excel text import wizard (stage 1)	164

9.7	Excel text import wizard (stage 3)	165
9.8	Sort options in Excel	166
9.9	Options for defining subcorpora according to genre	167
9.10	BNCweb keyword and title scan	168
9.11	AntConc Keyword preferences	170
9.12	Keyword options in BNCweb	172
9.13	Keyword comparison of university essays and written component of the BNC (top 31 entries)	173
10.1	Illustration of a collocation span	208
10.2	Options for statistical collocation measures in BNCweb	209
11.1	A brief SGML sample	232
11.2	CSS sample paragraph styling	235
11.3	TEI header for BNC file KST	247

List of Tables

2.1	Extract from <i>Beowulf</i> , encoded/represented in two different ways	14
2.2	Early written corpora	16
2.3	Composition of the Brown Corpus	17
2.4	Some examples of (earlier) spoken corpora	18
2.5	Early mixed corpora	19
2.6	Modern mega corpora	20
2.7	Examples of diachronic corpora	21
2.8	Examples of academic corpora	22
2.9	Examples of learner corpora	23
2.10	Selection of pragmatically annotated corpora	24
3.1	Common file formats and their properties	37
7.1	Ambiguous PoS tags in the Brown Corpus	102
7.2	The Penn Treebank tagset (based on Taylor et al. 2003: 8)	103
7.3	The CLAWS 7 (C7) tagset	105
8.1	Wildcards and their uses for investigating linguistic features	129
8.2	Simplified tags in BNCweb	132
8.3	Word + PoS tags breakdown for <i>mind</i>	142
9.1	Top 15 most frequent word types in section A of the LOB Corpus	157
9.2	Recalculated norming sample from Biber, Conrad & Reppen (1998: 263)	175
11.1	Annotation types listed in Garside et al. 1997	228
11.2	CSS properties for XML visualisation exercise	240
11.3	Colour semantics and CSS styles	243

Acknowledgements

I'd first like to start by thanking my former students in Chemnitz, Bayreuth and Hong Kong who 'suffered through' the initial sets of teaching materials that eventually formed the basis for writing this textbook. The next big thanks needs to go to my colleagues here at Guangdong University of Foreign Studies, who attended a series of workshops where I tested out the materials from the preliminary drafts of several chapters and who provided me with highly useful feedback. Particular mention here deserves to go to Junyu (Mike) ZHANG, who not only commented on the content of several chapters, but also pointed out certain issues of style that have hopefully helped me to make the writing more accessible to an international readership.

My next round of thank yous goes to Yanping DONG and Hai XU, for allowing me to join the National Key Research Center for Linguistics and Applied Linguistics at Guangdong University of Foreign Studies, which has provided me with more of the desperately needed time to focus on writing this book, while also allowing me to conduct other types of research that have influenced the contents of the book in various ways. To Hai XU, I give additional thanks for sharing his experience in, and knowledge of, corpora of Chinese, which has, unfortunately, only partially found its way into this book, due to limits of space. To my other colleagues, especially Yiqiong ZHANG, I also give thanks for providing a more moral type of support through engaging in further discussions and making me feel at home in the Center.

I also owe a great debt to Laurence Anthony for allowing me to pester him with a series of questions about and suggestions for improving AntConc. More or less the same, though possibly to a slightly lesser extent, goes for Sebastian Hoffmann and Mark Davies for answering questions about particular features of, and again

partly responding to requests for improving, BNCweb and the COCA interface, respectively.

Next, I'd sincerely like to thank the anonymous reviewers of this book, who, through their many invaluable constructive comments, have not only encouraged me in writing the book, but also hopefully allowed me to improve the contents substantially.

My final – but most important and heartfelt – note of thanks goes to Geoff Leech. Although credit for introducing me to the study of corpus linguistics has to go to someone else, he's certainly been the single most important influence on my career and thinking as a corpus linguist. I'll forever be grateful for his ongoing support throughout my years spent at Lancaster, working with him, as external examiner of my PhD, and later up to his untimely demise in August 2014. I sincerely hope that he would have appreciated the design and critical aims of this textbook, and perhaps also recognised his implicit hand in shaping it...

I

Introduction

This textbook aims to teach you how to analyse and interpret language data in written or orthographically transcribed form (i.e. represented as if it were written, if the original data is spoken). It will do so in a way that should not only provide you with the technical skills for such an analysis for your own research purposes, but also raise your awareness of how corpus evidence can be used in order to develop a better understanding of the forms and functions of language. It will also teach you how to use corpus data in more applied contexts, such as e.g. in identifying suitable materials/examples for language teaching, investigating socio-linguistic phenomena, or even trying to verify existing linguistic theories, as well as to develop your own hypotheses about the many different aspects of language that can be investigated through corpora. The focus will primarily be on English-language data, although we may occasionally, whenever appropriate, refer to issues that could be relevant to the analysis of other languages. In doing so, we'll try to stay as theory-neutral as possible, so that no matter which 'flavour(s)' of linguistics you may have been exposed to before, you should always be able to understand the background to all the exercises or questions presented here.

The book is aimed at a variety of readers, ranging mainly from linguistics students at senior undergraduate, Masters, or even PhD levels who are still unfamiliar with corpus linguistics, to language teachers or textbook developers who want to create or employ more real-life teaching materials. As many of the techniques we'll be dealing with here also allow us to investigate issues of style in both literary and non-literary text, and much of the data we'll initially use actually consists of fictional works because these are easier to obtain and often don't cause any copyright

issues, the book should hopefully also be useful to students of literary stylistics. To some extent, I also hope it may be beneficial to computer scientists working on language processing tasks, who, at least in my experience, often lack some crucial knowledge in understanding the complexities and intricacies of language, and frequently tend to resort to mathematical methods when more linguistic (symbolic) ones would be more appropriate, even if these may make the process of writing ‘elegant’ and efficient algorithms more difficult.

You may also be asking yourself why you should still be using a textbook at all in this day and age, when there are so many video tutorials available, and most programs offer at least some sort of online help to get you started. Essentially, there are two main reasons for this: a) such sources of information are only designed to provide you with a basic overview, but don’t actually teach you, simply demonstrating how things are done. In other words they may do a relatively good job in showing you one or more ways of doing a few things, but often don’t really allow you to use a particular program independently and for more complex tasks than the author of the tutorial/help file may actually have envisaged. And b) online tutorials, such as the ones on YouTube, may not only take a rather long time to (down)load, but might not even be (easily) accessible in some parts of the world at all, due to internet censorship.

If you’re completely new to data analysis on the computer and working with – as opposed to simply opening and reading – different file types, some of the concepts and methods we’ll discuss here may occasionally make you feel like you’re doing computer science instead of working with language. This is, unfortunately, something you’ll need to try and get used to, until you begin to understand the intricacies of working with language data on the computer better, and, by doing so, will also develop your understanding of the complexity inherent in language (data) itself. This is by no means an easy task, so working with this book, and thereby trying to develop a more complete understanding of language and how we can best analyse and describe it, be it for linguistic or language teaching purposes, will often require us to do some very careful reading and thinking about the points under discussion, so as to be able to develop and verify our own hypotheses about particular language features. However, doing so is well worth it, as you’ll hopefully realise long before reaching the end of the book, as it opens up possibilities for understanding language that go far beyond a simple manual, small-scale, analysis of texts.

In order to achieve the aims of the book, we’ll begin by discussing which types of data are already readily available, exploring ways of obtaining our own data, and developing an understanding of the nature of electronic documents and what may make them different from the more traditional types of printed documents we’re all familiar with. This understanding will be developed further throughout the book, as we take a look at a number of computer programs that will help us to conduct our analyses at various levels, ranging from words to phrases, and to even larger units of text. At the same time, of course, we cannot ignore the fact that there may be issues in corpus linguistics related to lower levels, such

as that of morphology, or even phonology. Having reached the end of the book, you'll hopefully be aware of many of the different issues involved in collecting and analysing a variety of linguistic – as well as literary – data on the computer, which potential problems and pitfalls you may encounter along the way, and ideally also how to deal with them efficiently. Before we start discussing these issues, though, let's take a few minutes to define the notion of (linguistic) data analysis properly.

1.1 Linguistic Data Analysis

1.1.1 What's data?

In general, we can probably see all different types of language manifestation as language data that we may want/need to investigate, but unfortunately, it's not always possible to easily capture all such 'available' material for analysis. This is why, apart from the 'armchair' data available through introspection (cf. Fillmore 1992: 35), we usually either have to collect our materials ourselves or use data that someone else has previously collected and provided in a suitable form, or at least a form that we can adapt to our needs with relative ease. In both of these approaches, there are inherent difficulties and problems to overcome, and therefore it's highly important to be aware of these limitations in preparing one's own research, be it in order to write a simple assignment, a BA dissertation, MA/PhD thesis, research paper, etc.

Before we move on to a more detailed discussion of the different forms of data, it's perhaps also necessary to clarify the term *data* itself a little more, in order to avoid any misunderstandings. The word itself originally comes from the plural of the Latin word *datum*, which literally means '(something) given', but can usually be better translated as 'fact'. In our case, the data we'll be discussing throughout this book will therefore represent the 'facts of language' we can observe. And although the term itself, technically speaking, is originally a plural form referring to the individual facts or features of language (and can be used like this), more often than not we tend to use it as a singular mass noun that represents an unspecified amount or body of such facts.

1.1.2 Forms of data

Essentially, linguistic data comes in two general forms, written or spoken. However, there are also intermediate categories, such as texts that are written to be spoken (e.g. lectures, plays, etc.), and which may therefore exhibit features that are in between the two clear-cut variants. The two main media types often require rather radically different ways of 'recording' and analysis, although at least some of the techniques for analysing written language can also be used for analysing transliterated or (orthographically) transcribed speech, as we'll see later when looking at some dialogue data. Beyond this distinction based on medium, there are of

course other classification systems that can be applied to data, such as according to *genre*, *register*, *text type*, etc., although these distinctions are not always very clearly formalised and distinguished from one another, so that different scholars may sometimes be using distinct, but frequently also overlapping, terminology to represent similar things. For a more in-depth discussion of this, see Lee (2002).

To illustrate some of the differences between the various forms of language data we might encounter, let's take a look at some examples, taken from the Corpus of English Novels (CEN) and Corpus of Late Modern English Texts, version 3.0 (CLMET3.0; De Smet, 2005), respectively. To get more detailed information on these corpora, you can go to <https://perswww.kuleuven.be/~u0044428/>, but for our purposes here, it's sufficient for you to know that these are corpora that are mainly of interest to researchers engaged in literary stylistic analyses or historical developments within the English language. However, as previously stated, throughout the book, we'll often resort to literary data to illustrate specific points related to both the mechanics of processing language and as examples of genuinely linguistic features. In addition to being fictional, this data will often not be contemporary, simply because much contemporary data is often subject to copyright. Once you understand more about corpora and how to collect and compile them yourself, though, you'll be able to gather your own contemporary data, should you wish so, and explore actual, modern language in use.

Apart from being useful examples of register differences, the extracts provided below also exhibit some characteristics that make them more difficult to process using the computer. We'll discuss these further below, but I've here highlighted them with boxes.

Sample A – from *The Glimpses Of The Moon* by Edith Wharton, published 1922

IT rose for them--their honey-moon--over the waters of a lake so famed as the scene of romantic raptures that they were rather proud of not having been afraid to choose it as the setting of their own.

“It required a total lack of humour, or as great a gift for it as ours, to risk the experiment,” Susy Lansing opined, as they hung over the inevitable marble balustrade and watched their tutelary orb roll its magic carpet across the waters to their feet.

“Yes--or the loan of Strefford's villa,” her husband emended, glancing upward through the branches at a long low patch of paleness to which the moonlight was beginning to give the form of a white house-front.

Sample B – from: *Eminent Victorians* by Lytton Strachey, published 1918

Preface

THE history of the Victorian Age will never be written; we know too much about it. For ignorance is the first requisite of the historian—ignorance, which simplifies and clarifies, which selects and omits, with a placid perfection unattainable by the highest art. Concerning the Age which has just passed, our fathers and our grandfathers have poured forth and accumulated so vast a quantity of information that the industry of a Ranke would be submerged by it, and the perspicacity of a Gibbon would quail

before it. It is not by the direct method of a scrupulous narration that the explorer of the past can hope to depict that singular epoch. If he is wise, he will adopt a subtler strategy. He will attack his subject in unexpected places; he will fall upon the flank, or the rear; he will shoot a sudden, revealing searchlight into obscure recesses, hitherto undivined. He will row out over that great ocean of material, and lower down into it, here and there, a little bucket, which will bring up to the light of day some characteristic specimen, from those far depths, to be examined with a careful curiosity.

Sample C – from *The Big Drum* by Arthur Wing Pinero, published 1915

Noyes.

[Announcing Philip.] Mr. Mackworth.

Roope.

[A simple-looking gentleman of fifty, scrupulously attired—jumping up and shaking hands warmly with Philip as the servant withdraws.] My dear Phil!

Philip.

[A negligently—almost shabbily—dressed man in his late thirties, with a handsome but worn face.] My dear Robbie!

Roope.

A triumph, to have dragged you out! [Looking at his watch.] Luncheon isn't till a quarter-to-two. I asked you for half-past-one because I want to have a quiet little jaw with you beforehand.

Philip.

Delightful.

Roope.

[Er—I]’d better tell you at once, old chap, whom you’ll meet here [to-day].

Sample A is clearly a piece of narrative fiction, mixing narrative description and simulated reported speech, references to characters and situations that are depicted as life-like, as well as featuring a number of at least partly evaluative reporting verbs, such as *opined* and *emended*. **Sample B**, on the other hand, contains no reported speech and reporting verbs, although it’s clearly also narrative – albeit non-fictional –, with a relatively complex sentence structure, including numerous relative and adverbial clauses, and an overall high degree of formality. **Sample C**, in contrast, exhibits clear characteristics of (simulated) spoken language, much shorter and less complex syntax, even single-word ‘sentences’, with names, titles and informal terms of address (*old chap*) used when the characters are addressing/introducing each other, exclamations, contractions, and at least one hesitation marker (*Er*). And even though the language in the latter sample seems fairly natural, we can still easily see that it comes from a scripted text, partly because of the indication of speakers (which I’ve highlighted in bold-face), and partly due to the stage instructions included in square brackets.

As we haven’t discussed any of the issues in processing such text samples yet, it may not be immediately obvious to you that these different types of register may potentially require different analysis approaches, depending on what our exact

aims in analysing them are. For instance, for **Sample A**, do we want to conceptually treat the reported speech as being of the same status as the descriptive parts, and do we thus want to analyse them together or separately? Or are we possibly just interested in how the author represents the direct speech of the characters in the novel, and would therefore want to extract only that? And if so, how would we best go about this?

Sample B is probably relatively straightforward to analyse in terms of perhaps a frequency analysis of the words, but what if we're also interested in particular aspects of syntax or lexis that may be responsible for its textual complexity or the perceived level of formality, respectively? And, last but not least, concerning **Sample C**, similarly to **Sample A**, which parts of the text would we be interested in here and how would we extract them? Are the stage instructions equally important to us as the direct speech exchanges between the characters? Or, if, for example, we're interested in the average number of words uttered by each character, how do we deal with hesitation markers? Do we treat them as words or 'non-words' simply to be deleted? As I've already tried to hint at in the beginning of this paragraph, the answers to these questions really depend on our research purpose(s), and can thus not be conclusively stated here.

Something else you may have noticed when looking at the samples I've provided above is that they're all from the early 20th century. As such, the language we encounter in them may sometimes appear overly formal (or even archaic) to some extent, compared to the perhaps more 'colloquial' language we're used to from the different registers these days. I've chosen extracts from these three particular texts and period for a number of reasons: a) their authors all died more than 70 years ago so the texts are in the public domain; in other words, there are no copyright issues, even when quoting longer passages; b) they are included in corpus compilations; and c) they not only illustrate register/genre differences but also how the conventions for these may change over time, as can be seen, for example, in the spelling of *to-day* in the final extract.

As pointed out above, another interesting aspect of these samples is that they exhibit particular formatting issues, which again may not be immediately apparent to you yet, but are due to somewhat bizarre typographical conventions. If you look closely at the samples, you can see that in **Sample A** there are double dashes marking the parenthetical counterpart (i.e. reference resolution) "their honey-moon" to the sentence-initial cataphoric pronoun "IT". What is in fact problematic to some extent for processing the text is that these dashes actually look like double hyphens, i.e. they're not surrounded by spaces on either side, as would be the normal convention. Now, many computer programs designed to count words will split the input text on spaces and punctuation. Unfortunately, though, this would leave us with some very strange 'words' (that superficially look like hyphenated compounds), *them—their* and *honey-moon—over*, in any resulting word-frequency list. This is obviously something we do not want and which introduces errors into any automatic analysis of the data. Something similar, albeit not to signal a parenthetical but instead some kind of pseudo-punctuation, happens

again for “Yes—or” a little further down in the text. We can already see, therefore, from this relatively short sample of text that a failure to deal with this feature could cause issues in a number of places throughout the text. The same problem occurs in the other two samples, only that there the dash doesn’t actually consist of two separate characters, but one single *m-dash*.

A different problem occurs in the use of initial capitals in **Samples A** and **B**. As you can see, the words *it* and *the* appear in capital letters throughout, signalling the beginning of the chapter typographically. Again, as ‘human consumers’ of the text, this will not cause any processing problems, but for the computer, *the*, *The*, and *THE* are in fact three different ‘words’, or at least word forms. Thus, even single initial capitals at the beginning of sentences may become issues in identifying and counting words on the computer. We’ll talk more about this type of issue in Section 4.4.1, where we’ll explore ways of dealing with such features of the text in order to retain relatively ‘clean’ data.

1.1.3 Collecting and analysing data

When collecting our own data, we obviously need to consider *methodologies* that allow us to collect the right types and amount(s) of data to answer our particular *research questions*. This, however, isn’t the only type of consideration necessary, but we also need to bear in mind *ethical issues* involved in the collection – such as asking people for permission to record them or to publish their recordings, etc. – and which type of *format* that data should be stored in so as to be most useful to us, and potentially also other researchers.

When using other people’s existing data, there are usually issues in accessing data stored in their specific format(s) or converting the data to a format that is more suitable to one’s own needs, as we’ve just seen above, such as removing unwanted types of information or transforming overly specific information into simpler forms of representation. In this textbook, we’ll also look at some of the important aspects of collecting or adapting data to one’s needs, as well as how to go about analysing and presenting them in various ways, once a suitable format has been established.

In order to be able to work with electronic data, we also need to become familiar with a variety of different programs, some written specifically for linguistic analysis, some for more general purposes of working with texts. One of the key features of this book is that the programs I’ll recommend to you are almost exclusively obtainable free of charge, i.e. so-called *freeware*. This doesn’t mean that there aren’t other excellent programs out there that may do some of the tasks we want to perform even better, or in simpler or more powerful ways, but simply reflects the fact that there are already many useful free programs available, and also my own philosophy that we shouldn’t need to spend substantial amounts of money just to enable us to do research. This is at least part of the reason why I make most of my own programs available to the research community in this way, apart from the fact that this makes my own research (results) more easily reproducible by

others, and therefore caters for the aims of satisfying *accountability* and academic honesty. For the sake of completeness, though, I'll generally try to at least refer to alternative commercial programs, but without discussing them in any detail.

Corpus linguistics, as a form of data analysis methodology, can of course be carried out on a number of different operating systems, so I'll also try to make recommendations as to which programs may be useful for the most commonly used ones, Windows, Mac OS X, and Linux. Because there are many different 'flavours' of Linux, though, with a variety of different windowing interfaces, I'll restrict my discussions to two of the most popular ones, KDE and Gnome. Unfortunately, I won't be able to provide detailed support on how to actually install the programs themselves, as this may sometimes involve relatively detailed information about your system that I cannot predict. Instead, however, I'll actually try to avoid/pre-empt such issues by recommending default programs that are probably already installed, provided that they do in fact fulfil all or at least most of our needs.

1.2 Outline of the Book

This book is organised into four sections. The first section (comprising Chapters 1 and 2) begins with a very brief introduction to the history and general design of corpora, simply to 'set the scene', rather than to provide an extensive coverage of the multitude of corpora that have been created for different purposes and possibly also made available for free or in the form of various types of interfaces. More extensive coverage on the subject, including more theoretical implications, is already provided in books like Kennedy (1998), Meyer (2002), or Lindquist (2009), so these texts can always be consulted for reference if necessary, and we can instead focus on more practical issues. For a more detailed interesting discussion of some of the different 'philosophical' approaches to corpus linguistics, you can consult McEnery and Hardie (2012).

The introductory section is followed by an overview of different methods to compile and prepare corpora from available online resources, such as *text archives* or the *WWW*. This section (spanning Chapters 3 and 4) should essentially provide the basis for you to start building your own corpora, but also introduces you to various issues related to handling language on the computer, including explanations of different file types you may encounter or want to use, as well as certain types of meta-information about texts.

Section 3 (Chapters 5 to 10) then deals with different approaches to corpus-based linguistic data analysis, ranging from basic searching (concordancing) via learning about more complex linguistic patterns, expressed in the form of regular expressions, to simple and extended word (frequency) list analyses. This part already contains information on how to tag your data morpho-syntactically, using freely available tagging resources, and how to make use of tagging in your analyses. The final section then takes the notion of adding linguistic information to

your data further, and illustrates how to enrich corpus data using basic forms of XML in order to cyclically improve your analyses or publish/visualise analysis results effectively.

As corpus linguistics is a methodology that allows us to develop insights into how language works by ‘consulting’ real-life data, it should be fairly obvious that we cannot learn how to do corpus research on a purely theoretical basis. Therefore, as far as possible, all sections of this book will be accompanied by practical exercises. Some of these will appear to be relatively straightforward, almost mechanical, ones where you simply get to follow a sequence of steps in order to learn how to use a specific function inside a program or web interface, while others are more explicitly designed to enable you to develop your own strategies for solving problems and testing hypotheses in linguistics. Please bear in mind, though, that for the former type of exercise, simply following the steps blindly without trying to understand why you’re doing them will not allow you to learn properly. So, as far as possible, at each point you should try to understand what we’re trying to achieve and how the particular program we’re using only gives us a handle on producing the relevant data, but does not actually answer our research questions for us. In the same vein, it’s also important to understand that once we actually have extracted some relevant data from a corpus, this is rarely ever the ‘final product’. Such data generally either still needs to be interpreted, filtered, or evaluated as to its usefulness, if necessary by (re-)adjusting the search strategy or initial hypotheses and/or conclusions, or, if it’s to be used for more practical purposes, such as in the creation of teaching materials or exercises, to be brought into an appropriate form.

As we move on and you learn more and more techniques, the exercises will also get more complex, sometimes assuming the size of small research projects, if carried out in full detail. As a matter of fact, as these exercises require and consolidate a lot of the knowledge gained in prior sections, they might well be suitable for small research projects to be set by teachers, and possibly even form the basis of BA theses or MA dissertations.

Of course, you won’t be left alone in figuring out the solutions to these exercises; both types will be solved at the end of each respective section, either in the form of detailed and precise explanations, or, whenever the results might be open to interpretation, by appropriate comments illustrating what you could/should be able to observe. For the more extensive exercises referred to in the previous paragraph, I’ll often start you off with suitable explanations regarding the procedures to follow, and also hint at some potential issues that may arise, but will leave the completion up to you, to help you develop your awareness independently. Furthermore, as real corpus linguistics is not just about getting some ‘impressive’ numbers but should in fact allow you to gain real insights into different aspects of language, you should always try to relate your results to what you know from established theories and other methods used in linguistics, or even other related disciplines, such as for example sociology, psychology, etc., as far as they may be relevant to answering your research questions. This is also why the solutions to,

and discussions of, the exercises may often represent those parts of the book that cover some of the more theoretical aspects of corpus linguistics, aspects that you'll hopefully be able to master once you've acquired the more practical tools of the trade. Thus, even if you may think you've already found a perfect answer to an exercise, you should probably still spend some time reading carefully through each solution.

As this textbook is more practical in nature than other textbooks on corpus linguistics, at the end of almost all chapters, I've also added a section entitled 'Sources and Further Reading'. These sections essentially provide lists of references I've consulted and/or have found most useful and representative in illustrating the particular topic(s) discussed in the chapter. You can consult these references if you want to know more about theoretical or practical issues that I am unable to cover here, due to reasons of space. These sections may not necessarily contain highly up-to-date references, for the simple reason that, unfortunately, later writings may not always represent improvements over the more fundamental works produced in some of the areas covered. Once you understand more about corpus linguistics, though, you may want to consult the individual chapters in two of the recent handbooks, O'Keefe & McCarthy (2010) and Lüdeling & Kytö (2008), so that you can evaluate the progress made over recent years yourself.

1.3 Conventions Used in this Book

In linguistics, there are many conventions that help us to distinguish between different levels of analysis and/or description, so as to better illustrate which different types of language phenomena we're dealing with at any given point in time. Throughout this book, I'm going to make use of many, if not most, of these conventions, so it's important to introduce them at this point. In addition to using these conventions as is done in linguistics, I may also use some of them to indicate special types of textual content relevant to the presentation of resources in this book, etc.

Double quotes ("...") indicate direct speech or short passages quoted from books.

Single quotes ('...') signal that an expression is being used in an unusual or unconventional way, that we're referring to the meaning of a word or construction on the semantic level, or to refer to menu items or sometimes button text in programs used. The latter may also be represented by a stylised button text, e.g.

Start

Curly brackets ({...}) are used to represent information pertaining to the level of morphology.

Angle brackets (<...>) indicate that we're dealing with issues related to orthography or spelling. Alternatively, they're also used in certain types of linguistic annotation.

Forward slashes/square brackets generally indicate that we're discussing issues on the levels of phonology or phonetics. Within quoted material, they may also signal amendments to the original material made in order to fit it into the general sentence structure.

Italics are used to represent words or expressions, sometimes whole sentences, that illustrate language materials under discussion. In some cases, they may also be used to indicate emphasis/highlighting, especially if co-occurring with boldface.

Small caps are used to indicate lemmas, i.e. forms that allow us to conveniently refer to all instances of a verb, noun, etc.

`Monospaced font` indicates instructions/text to be typed into the computer, such as a search string or regular expression.

1.4 A Note for Teachers

The relatively low number of chapters may make this book appear deceptively short, and you might be wondering whether it would be suitable for a course that runs for a whole semester of up to 18 weeks; there's no need to worry, though, that you may necessarily have to supplement it with further materials, although this is of course possible.

The sections and chapters of the book have been arranged to be thematically coherent, but, if you're planning to use it as a textbook in class, you'll frequently find that one chapter corresponds to more than one classroom unit. I'd therefore suggest that, while preparing specific topics, even – or especially – if you may already be an expert in the field, you at least try out the exercises carefully yourself, and then attempt to gauge how long it may take your students to carry them out. If your audience is already highly technically literate and has a strong background in linguistics, then obviously the exercises can be done much more quickly. If, on the other hand, your students are somewhat 'technophobic' or do not yet have a strong background in linguistics, you may either wish to spread the content over multiple units, or set at least some of the exercises as homework. In order to save time, you can also ask your course participants to perform certain preparatory tasks, such as downloading and installing different pieces of software, or registering for online resources, prior to coming to class.

1.5 Online Resources

This book also has an accompanying web page, where you'll be able to find some online exercises, links to my own software, updated information about programs or features discussed in the book, etc. The web address for this page is http://martinweisser.org/pract_cl/online_materials.html, and you'll probably want to bookmark this straight away, so that you'll be able to access it for future reference.

All my own software is provided under GPL 3 licence, so you can download and distribute it freely. The programs were originally designed to run on Windows, but can easily be used through Wine (<https://www.winehq.org/>) on Mac OS X or Linux. Additional information on how to do this can be found at http://martinweisser.org/ling_soft.html.

2

What's Out There? *A General Introduction to Corpora*

2.1 What's a Corpus?

A corpus (pl. *corpora*) is a collection of spoken or written texts to be used for linguistic analysis and based on a specific set of design criteria influenced by its purpose and scope. There are various, and sometimes conflicting, definitions in the relevant literature (c.f. e.g. Kennedy, 1998: 3 or Meyer, 2002: xi) as to what exactly constitutes a corpus, but for our purposes, we'll adopt the relatively simple and straightforward one given above. This basically means that any collection of texts that has been *systematically* assembled in order to investigate one or more linguistic phenomena can be termed a corpus, even if it may only contain a handful of classroom transcripts, interviews, or plays.

Although, theoretically, corpora can simply consist of texts that are in non-electronic form, and indeed some of the earliest corpora were just collections of index cards or slips of paper (McCarthy & O'Keeffe 2010: 4), these days, almost all corpora in use are computerised. When we talk about corpora from now on, we'll thus always be referring to computerised ones, unless otherwise stated.

2.2 Corpus Formats

Most corpora – unless they're only accessible through an online interface – are stored in *plain-text format* (to be explained in more detail in Section 3.2.3) and can therefore easily be viewed using any basic *text editor*, but if a corpus example

contains phonetic transcriptions, then of course a specialised typeface (*font*) may be required in order to view it. Complications may also arise if the *character set* is not directly supported by the computer the corpus is viewed on. This may for example happen when the corpus is in a language that uses a different alphabet from the standard Western European ones that are supported on all computers by default. Even displaying European languages, such as Greek, may represent a (minor) problem, but the real difficulties start when one wants to work with East Asian character sets (such as for Chinese, Japanese, Korean & Vietnamese), Indic languages (such as Hindi or Urdu), or right-to-left languages like Arabic and Hebrew. For a simple simulation of this, see the online resources at: http://martinweisser.org/pract_cl/encoding.html. In order to overcome these encoding problems, these days more and more corpora are encoded in one of the *Unicode encodings*, most commonly UTF-8. To illustrate this, let's take a look at an extract from the *Helsinki Corpus* (see Section 2.3.3) of the first lines of the Old English *Beowulf* epic, presented in the original (slightly modified) representation in the corpus and a modern recoding in UTF-8.

Table 2.1 Extract from *Beowulf*, encoded/represented in two different ways

<i>Original (slightly modified)</i>	<i>Re-coded Version</i>
[] [\BEOWULF\]]	BEOWULF
Hw+at. We Gardena in geardagum, +teodcyninga, +trym gefrunon, hu +da +a+telingas ellen fremedon.	Hwæt. We Gardena in geardagum, þeodcyninga, þrym gefrunon, hu ða æþelingas ellen fremedon.
Oft Scyld Scefing [{scea+tena{ } +treatum, monegum m+ag+tum, meodosetla ofteah, egsode eorlas.	Oft Scyld Scefing <i>sceaþena</i> þreatum, monegum mægþum, meodosetla ofteah, egsode eorlas.
Sy+d+dan +arest [{wear+d{ } feasceaft funden, he +t+as frofre gebad, weox under wolcnum, weor+dmyndum +tah, o+d+t+at him +aghwylc +tara ymsittendra ofer hronrade hyran scolde, gomban gyldan.	Syððan ærest <i>wearð</i> feasceaft funden, he þæs frofre gebad, weox under wolcnum, weorðmyndum þah, oðþæt him æghwylc þara ymsittendra ofer hronrade hyran scolde, gomban gyldan.
+t+at w+as god cuning.	þæt wæs god cuning.
+d+am eafera w+as +after cenned, geong in geardum, +tone god sende folce to frofre; fyren+dearfe ongeat +te hie +ar drugon [{aldorlease{ } lange hwile.	ðam eafera wæs æfter cenned, geong in geardum, þone god sende folce to frofre; fyrenðearfe ongeat þe hie ær drugon <i>aldorlease</i> lange hwile.
Him +t+as liffrea, wuldres wealdend, woroldare forgeaf; Beowulf w+as breme bl+ad wide [{sprang{ } , Scyldes eafera Scedelandum in.	Him þæs liffrea, wuldres wealdend, woroldare forgeaf; Beowulf wæs breme blæd wide <i>sprang</i> , Scyldes eafera Scedelandum in.
Swa sceal [{geong{ } [{guma{ } gode gewyrcean, fromum feohgiftum on f+ader [{bearme{ } , +t+at hine on ylde eft gewunigen wilgesi+tas, +tonne wig cume, leode gel+asten; lofd+adum sceal in m+ag+ta gehw+are man ge+teon.	Swa sceal <i>geong guma</i> gode gewyrcean, fromum feohgiftum on fæder <i>bearme</i> , þæt hine on ylde eft gewunigen wilgesiþas, þonne wig cume, leode gelæsten; lofdædum sceal in mægþa gehwære man geþeon.

As we can see from the above examples, the original version from the Helsinki Corpus is much more difficult to read because we always need to mentally replace each *transliterated* character by the appropriate Old English one. Furthermore, emendations, i.e. corrections to the original text, are indicated in the corpus by surrounding the text with a set of opening and closing square and curly brackets ([{...}]), which again distracts while reading, so that I've rendered the modified text in ***bold and italic formatting*** to make it easier to read.

Another way to store a corpus is to save it into a database. This makes it more difficult to view and process without having the appropriate *database management system* (DBMS) installed or if a web-based online interface isn't working as expected, due to browser issues or download speed restrictions. On the other hand, though, this makes it possible for the basic text to be linked to various types of data-enriching annotations, as well as to perform more complex search operations, or to store intermediate or final results of such searches for different users and for quicker access or export later. We'll experience the advantages of this when we set up/work with accounts for access to some web-based corpus interfaces, such as BNCweb or COCA.

2.3 Synchronic vs. Diachronic Corpora

Corpora can be designed and used for *synchronic* (i.e. 'contemporary') and *diachronic* (i.e. 'historical'/comparative) studies. Different issues may apply to the design of these two types of corpora. For instance, historical corpora may contain old-fashioned or unfamiliar words and spellings or a large number of *spelling variants* (e.g. *yeare*, *hee*, *generalitie*, *it selfe*, etc.), as well as possible even characters (letters) that no longer exist in a modern alphabet, such as the Old English *thorn* (*þ*), which we've already encountered in the above *Beowulf* extract.

Historical corpora are also by nature restricted to written materials because there just are no recordings of native speakers of Old or Middle English in existence. Furthermore, the restriction does not only apply to the types of material available but also to the amount of data we can draw on because, in former times, there simply wasn't such a wealth of documents available, and from as many different sources as we have them today.

2.3.1 'Early' synchronic corpora

Another major distinction between different types of corpora is whether they comprise spoken or written data. This is an extremely important distinction because written language generally tends to be far easier to process than spoken language, as it does not contain *fillers*, *hesitations*, *false starts* or any ungrammatical constructs. When creating a spoken corpus, one also needs to think about whether an orthographic representation of the text will be sufficient, whether the corpus

should be represented in phonetic transcription, or whether it should support *annotation* on various different levels (see Chapter 11).

Initially, computerised language corpora tended to contain only written language, which was easier to obtain, and presumably also deemed to be more important than spoken language, a notion that unfortunately still seems to be all-too-prevalent in our often 'literature-focussed' society and education.

2.3.1.1 Written corpora Let's start our investigation into the nature of corpora with a look at some of the earliest written ones, accompanied by a short exercise to sensitise you towards certain issues. At the time these first corpora were created, one million words still seemed like a huge amount of data, partly because computers in those days had a hard time handling even this amount, and partly because no-one had ever had such easy access to so much language data in electronic form before.

Table 2.2 provides a brief overview of some of these early corpora. The web addresses in the first column of this table generally link to the online versions of the respective manuals.

Table 2.2 Early written corpora

<i>Corpus</i>	<i>Description</i>	<i>Size (words, ca.)</i>
Brown http://clu.uni.no/icame/manuals/BROWN/INDEX.HTM ; (available from https://archive.org/details/BrownCorpus)	first-ever computerised corpus, published in 1964: written American English	1 million
LOB (Lancaster-Oslo-Bergen) http://clu.uni.no/icame/manuals/LOB/INDEX.HTM	published in 1978; British counterpart to Brown	1 million
Frown http://clu.uni.no/icame/manuals/FROWN/INDEX.HTM	published in 1999; 90s counterpart to Brown	1 million
FLOB http://clu.uni.no/icame/manuals/FLOB/INDEX.HTM	published in 1998; 90s counterpart to LOB	1 million
Kolhapur http://clu.uni.no/icame/manuals/KOLHAPUR/INDEX.HTM	published in 1978; written Indian English	1 million
ACE (Australian Corpus of English) http://clu.uni.no/icame/manuals/ACE/INDEX.HTM	compiled from 1986; also known as the 'Macquarie Corpus'	1 million
Wellington Corpus of Written New Zealand English http://clu.uni.no/icame/manuals/WELLMAN/INDEX.HTM	published in 1993	1 million

A complete set of all manuals of corpora distributed by the ICAME (*International Computer Archive of Modern and Medieval English*) can also be downloaded from <http://clu.uni.no/icame/icamemanuals.html>.

Exercise 1

Table 2.3 illustrates the composition of the Brown Corpus. Take a look at this and try to see whether you can gain some insights into the nature of the corpus in terms of its categories and number of texts it comprises.

What kind of language would you expect inside the different categories, and can you identify anything particularly interesting regarding them?

If you're already planning a research project, do you think data from these will fit your particular needs and help you to answer your research questions?

Once you've done this, also open some of the links to other manuals given in Table 2.2 and compare the composition of these corpora to the Brown Corpus.

Table 2.3 Composition of the Brown Corpus

<i>Label</i>	<i>Text Category/Genre</i>	<i>No. of Texts</i>
A	Press: Reportage	44
B	Press: Editorial	27
C	Press: Reviews	17
D	Religion	17
E	Skills & Hobbies	36
F	Popular Lore	48
G	Belles Lettres, Biography, Essays	75
H	Miscellaneous: Government Documents, Foundation Reports, Industry Reports, College Catalogue, Industry House Organ	30
J	Learned	80
K	General Fiction	29
L	Mystery & Detective Fiction	24
M	Science Fiction	6
N	Adventure & Western Fiction	29
P	Romance & Love Story	29
R	Humour	9

Compare the categories in the Brown Corpus to those of the Australian Corpus of English directly. Can you pinpoint the slight cultural difference?

You may have noticed that some letters are missing from the categorisation scheme, notably I, O, and Q. This is probably because the uppercase letter I can

easily be confused with the number 1 or lowercase l, and uppercase O with the number 0. I have no logical explanation why Q is not used, unless the assumption is that Q is similar to O, and hence the same options for confusion may arise.

2.3.1.2 Spoken corpora Next, let's take a look at a selection of important spoken corpora to develop a better understanding of where the differences between written and spoken corpora are.

Table 2.4 Some examples of (earlier) spoken corpora

<i>Corpus</i>	<i>Description</i>	<i>Size (words, ca.)</i>
SEU (Survey of English Usage)	non-computerised	1 million
LLC (London-Lund Corpus) http://clu.uni.no/icame/manuals/LONDLUND/INDEX.HTM (available from http://ota.ahds.ac.uk/desc/0168)	published in 1990 (?); spoken	500,000
SEC (Spoken English Corpus) http://clu.uni.no/icame/manuals/SEC/INDEX.HTM	published in 1988; spoken	52,000
MapTask Corpus http://groups.inf.ed.ac.uk/maptask/	corpus of cooperative spoken interaction	
HKCSE (Hong Kong Corpus of Spoken English) http://rcpce.engl.polyu.edu.hk/HKCSE/	compilation of academic & business English, conversations, and public discourse	907,657

Exercise 2

Compare the categories of the SEC to one of the written corpora in Table 2.2 and try to see why/whether those different categories may be important for/representative of written and spoken language, respectively.

Look through the online page for the LLC and try to understand what types of additional information may need to be represented (encoded) in spoken corpora.

2.3.2 Mixed corpora

Mixed corpora try to establish some kind of balance between spoken and written language in order to be more representative of language in general. Earlier mixed corpora were still comparatively small in size, but this has changed with the advent of improved data collection methods and the ensuing creation of mega corpora that now run into hundreds of millions of words.

2.3.2.1 *Early mixed corpora*

Table 2.5 Early mixed corpora

<i>Corpus</i>	<i>Description</i>	<i>Size (words, ca.)</i>
International Corpus of English (ICE) http://www.ucl.ac.uk/english-usage/ice/index.htm	growing collection of corpora of English around the world; comprises written & spoken materials	1 million per corpus

Exercise 3

Open the ICE website and navigate to the ‘Corpus Design’ page. In light of the information you’ve already come across for individual written and spoken corpora, try to evaluate how similar/different the composition of the ICE corpora is to/from these ‘traditional’ corpora.

2.3.2.2 *Modern mega corpora and national corpora* With the use of corpora becoming more popular, and techniques for data analysis improving, researchers soon realised that corpora of one million words were not nearly large enough for observing all interesting linguistic phenomena, especially not those that involve idiomatic structures or collocations (see Chapter 10). Especially for investigating rarer features of the language, the basic notion thus seems to be ‘the bigger, the better’, and thus researchers, often supported by publishing houses who wanted to create better dictionaries or textbooks, began to compile corpora of 100 million words or more.

Such corpora, due to their rather large size, are of course more difficult to process on our own computers, and may not even be easy or affordable enough to obtain for individual research on a smaller scale. However, the latter issue is often not such a big problem after all because most openly accessible mega corpora these days provide online interfaces that users can sign up for free of charge. These interfaces will be covered more extensively in later chapters.

As we already have a basic understanding concerning the composition of the earlier, much smaller, corpora, we can now try and develop a basic understanding of how large-scale mega corpora may differ in that respect, and what the advantages in this may be, apart from simply having more text from different genres. This will also help to prepare you for working with them later. In order to do so, let’s take the BNC as an example and find out where some of the more significant differences may lie, and why using it may represent an advantage over simply working with the smaller earlier corpora that may be much easier to process on our own computer.

Table 2.6 Modern mega corpora

<i>Corpus</i>	<i>Description</i>	<i>Size (ca.)</i>
BNC (British National Corpus) http://www.natcorp.ox.ac.uk/corpus/ (also accessible through various online interfaces)	published in 1995; British reference corpus; 90% written – 10% spoken	100 million
ANC (American National Corpus) http://americannationalcorpus.org/	American counterpart to the BNC; still incomplete	22 million so far; approximately 15 million freely available
The Corpus of Contemporary American English (COCA) http://corpus.byu.edu/coca/	American corpus containing a balanced collection of spoken materials, fiction, popular magazines, newspapers, and academic texts, containing 20% of each major category	more than 450 million words of text, available through a freely accessible online interface
Corpus of Global Web-Based English (GloWbE) http://corpus.byu.edu/glowbe/	a collection of web pages in (native and non-native) English from 20 countries	1.9 billion

Exercise 4

Open the BNC website and read through the descriptions.

As before, try to understand in which way it may be similar to/different from other corpora, and where the advantages in this may lie.

As should be obvious from Table 2.6, there's still quite an imbalance between written and spoken materials, partly due to the fact that spoken materials are much more difficult to obtain and handle than written ones, as well as more costly to process.

The creation of the BNC also set an example for other countries to pursue the development of their own national corpora, some of which are directly modelled on the BNC. Such corpora exist for example for Polish (PELCRA), Czech (CNC), Chinese (Modern Chinese Language Corpus: MCLC), and Korean (Sejong/Korean National Corpus: KNC), to list but a few. For a more extensive overview of these, see Xiao (2008).

2.3.3 Examples of diachronic corpora

Since we have already covered some of the more important aspects of diachronic/historical corpora in conjunction with representing texts, and are mainly concerned with applied rather than historical aspects of corpora here, we will not

discuss this type of corpora further at this point. Thus, Table 2.7 is mainly intended to be a source of reference for the sake of completeness.

Table 2.7 Examples of diachronic corpora

<i>Corpus</i>	<i>Description</i>	<i>Size (words, ca.)</i>
Helsinki Corpus http://clu.uni.no/icame/manuals/HC/INDEX.HTM (also available through the Oxford Text Archive at: http://ota.ahds.ac.uk/headers/1477.xml)	covers historical materials from ca. 750–1700 + dialect transcripts of British rural dialects from the 1970's	1.6 million
The Lampeter Corpus of Early Modern English Tracts http://clu.uni.no/icame/manuals/LAMPETER/LAMPHOME.HTM (also freely available through the Oxford Text Archive after application for licence at http://ota.ahds.ac.uk/headers/2400.xml)	tracts and pamphlets published in the century between 1640 & 1740	1.1 million
ARCHER (A Representative Corpus of Historical English Registers) http://www.alc.manchester.ac.uk/subjects/lcl/research/projects/archer/	British & American texts of different registers from 1650 not freely accessible	1.7 million
Corpus of Historical American English (COHA) http://corpus.byu.edu/coha/	texts from 1810-2009 free online access	400 million

A much more exhaustive and detailed list for all sorts of different purposes is provided by David Lee on his highly useful – but sometimes overwhelming – corpora website at <http://tiny.cc/corpora>.

2.4 General vs. Specific Corpora

Apart from the distinctions noted above, we can also draw another one, namely to distinguish between corpora that are compiled for general purpose research and such that are highly domain specific. The former are deemed representative of the whole language and generally to be used for a wide variety of different research objectives. The latter, in contrast, are often only of limited use to the general public, but may also sometimes be useful because they can highlight particular differences between standard language and specific registers, etc. An example of a domain-specific literary corpus would be the collected works of an author, which can be used to investigate the style of this particular author, or even to verify

disputes about the authorship of a piece of literature where this may be contentious. Related to this are the types of specific corpora of witness and police officers' statements in *forensic linguistics* discussed in Cotterill (2008: 581–2), which may help in identifying whether witness statements have been manipulated before being admitted to court.

2.4.1 Examples of specific corpora

To some extent, we've already seen examples of specific corpora when we looked at some of the spoken corpora currently available. In recent years, there's also been a growing interest in many other types of specific corpora. Among these, we'll only discuss two specific types here, academic and learner corpora.

2.4.1.1 Academic corpora Academic corpora deal exclusively with language produced in academic contexts, i.e. *English for Academic Purposes* (EAP), a special type of *English for Specific Purposes* (ESP). They may consist of transcripts of academic lectures and seminars, various types of writing produced in a university context, but also sometimes include other academic activities, such as meetings or advisory/supervision sessions. They thus tend to reflect the speech of both experts and non-experts in the field of academia. Table 2.8 lists some of the better-known and more or less openly available academic corpora.

The basic idea behind creating and exploiting academic corpora is to be able to extract samples of expert and non-expert language use in academic settings in order to be able to investigate the nature of academic speech and writing, and make suggestions for teaching or best practice in academia.

Table 2.8 Examples of academic corpora

<i>Corpus</i>	<i>Description</i>	<i>Size</i> (<i>words, ca.</i>)
BASE (British Academic Spoken English) http://www.coventry.ac.uk/research-bank/research-archive/art-design/british-academic-spoken-english-corpus-base/ (also available, with restrictions, through the Oxford Text Archive at: http://ota.ahds.ac.uk/headers/2525.xml)	transcripts of 160 academic lectures and 39 seminars	1,644,942
BAWE (British Academic Written English) http://www.coventry.ac.uk/research-bank/research-archive/art-design/british-academic-written-english-corpus-bawe/ (also available, with restrictions, through the Oxford Text Archive at: http://ota.ahds.ac.uk/headers/2539.xml)	assignments from 3 UK universities	6,506,995

Table 2.8 Examples of academic corpora (*Continued*)

<i>Corpus</i>	<i>Description</i>	<i>Size</i> (<i>words, ca.</i>)
MICASE (Michigan Corpus of Academic Spoken English) http://quod.lib.umich.edu/m/micase/ ¹ (access exclusively through dedicated search interface at http://quod.lib.umich.edu/cgi/c/corpus/corpus?c=micase;page=simple or to browse at http://quod.lib.umich.edu/cgi/c/corpus/corpus?c=micase;page=mbrowse)	transcripts of various types of lectures & interactive exchanges in academic settings at the University of Michigan	1.8 million
MICUSP (Michigan Corpus of Upper-Level Student Papers) access exclusively through dedicated search interface at http://micase.elicorpora.info/	various types of writing by senior undergraduate (4th year) & graduate students	1.6 million

2.4.1.2 Learner corpora In contrast to the academic corpora introduced in Section 2.4.1.1, learner corpora, as their name implies, do not contain materials produced by experts in a field, but instead by students at different levels and stages of language acquisition, often restricted to non-native speakers, i.e. *L2* learners, of a language. Occasionally, though, we can also find corpora of *L1* learners, i.e. native speakers of a language. These are often created and used for comparison purposes to investigate differences between *L1* and *L2* learners, but may also be employed to explore different stages of development in the native language. Table 2.9 lists a number of recent learner corpora.

Table 2.9 Examples of learner corpora

<i>Corpus</i>	<i>Description</i>	<i>Size</i> (<i>words, ca.</i>)
ICLE (International Corpus of Learner English) http://www.uclouvain.be/en-cecl-icle.html	essays from higher intermediate to advanced learners from 16 non-native <i>L1</i> backgrounds	3.7 million
LINDSEI	spoken counterpart to ICLE	1 million +

(Continued)

Table 2.9 Examples of learner corpora (*Continued*)

<i>Corpus</i>	<i>Description</i>	<i>Size</i> (<i>words, ca.</i>)
LOCNESS http://www.uclouvain.be/en-cccl-locness.html (obtainable from http://www.learnercorpusassociation.org/resources/corpora/locness-corpus/)	essays produced by British school pupils and university students, and American university students	324,304
LOCNEC USE (Uppsala Student English Corpus) http://www.engelska.uu.se/Forskning/engelsk_spraketenskap/Forskningso mraden/Electronic_Resource_Projects/USE-Corpus/ (obtainable from http://www.ota.ahds.ac.uk/desc/2457)	spoken counterpart to LINDSEI 1,489 essays by 440 Swedish university students of English of three different levels	162,000 1,221,265
ICNALE (International Corpus Network of Asian Learners of English) info & downloads from http://language.sakura.ne.jp/icnale/index.html	spoken and written data from Asian learners of English from 10 countries, plus comparable native speaker data	Over 1 million words of written essays; 1,900 1-minute sound files

2.4.1.3 Pragmatically annotated corpora Pragmatically annotated corpora, i.e. corpora that are annotated for *speech acts* or other pragmatically relevant information, are still relatively rare. Many of them are also *task-oriented*, i.e. deal with relatively limited topics and domains, so that they're often not very representative of general spoken interaction. However, despite this limitation, they can already provide us with valuable insights into some of the general mechanisms involved in natural spoken language exchanges. Table 2.10 lists a small selection of the pragmatically annotated dialogue corpora available.

Table 2.10 Selection of pragmatically annotated corpora

<i>Corpus</i>	<i>Description</i>	<i>Size</i> (<i>words, ca.</i>)
Switchboard Dialog Act Corpus http://www.stanford.edu/~jurafsky/swb1_dialogact_annot.tar.gz	approximately 2,400 spontaneous telephone conversations by native speakers of American English, revolving around 52 topics	3 million
SPAAC (Speech Act Annotated Corpus of Dialogues)	1,200+ transactional dialogues; majority calls to BT operators, smaller part Trainline timetable information and bookings	166,114

Table 2.10 Selection of pragmatically annotated corpora (*Continued*)

<i>Corpus</i>	<i>Description</i>	<i>Size</i> (<i>words, ca.</i>)
SPAADIA Corpus http://martinweisser.org/spaadia_release.zip	Trainline data from the SPAAC Corpus, comprising 35 files	26,653 words
OASIS http://groups.inf.ed.ac.uk/oasis/ available on request from oasis-bt@inf.ed.ac.uk	calls to BT operators in two versions, one annotated at Edinburgh University and the other part of the SPAAC Corpus (see above)	280,000

The one major exception in Table 2.10 is the *Switchboard Dialog Act Corpus*, as it actually consists of spontaneous telephone conversations, rather than *transactional* (see Leech et al., 2000) dialogues.

2.5 Static Versus Dynamic Corpora

One further distinction we can make between different types of corpora is between those that are fixed in size and finalised in that they're never intend to change (i.e. static) and more dynamic types of corpora, which are explicitly designed to change over time and to keep on reflecting the ever-changing nature of language. We can refer to the former type as *snapshot corpora*, whereas the common term for the latter is *monitor corpora*. By this definition, in fact, almost all the corpora discussed above are snapshot corpora. Even the diachronic ones are, because they've not been designed to be added to later, even if, for example, at some point in time a further Old English epic may somehow be unearthed and could thus theoretically be included in a new edition of the Helsinki Corpus. To date, there are only two real monitor corpora in existence, the COCA and the Bank of English. The latter has not been covered earlier, as it's not directly accessibility to outsiders, although it forms part of the Collins WordBanks Online (WordBanks, 2009), which unfortunately is accessible only by subscription. Davies (2010), however, argues that the Bank of English, which was originally designed as a resource for creating the Collins COBUILD dictionaries, cannot be seen as a true monitor corpus, due to imbalance in its categories.

A genuine monitor corpus, as pointed out above, makes it possible to continually compare a language in its different stages of development, including the most recent changes, in and through one single corpus. An alternative solution to tracking change diachronically is to use static corpora, such as the LOB and the FLOB, which were designed to reflect the same categories of language, but using samples whose production dates were 30 years apart.

2.6 Other Sources for Corpora

Apart from some corpora that are made freely available on the web, there are also a few institutions that distribute corpora at a certain cost. The cost usually depends on whether or not someone wants to use the corpora commercially. Obviously, licences for commercial use tend to be a lot more expensive than for private or educational use, but unfortunately some corpora may still seem prohibitively expensive for the individual. Some major sources for corpora are:

- International Computer Archive of Modern and Medieval English (ICAME; <http://clu.uni.no/icame/>)
- Linguistic Data Consortium (LDC; <http://www ldc.upenn.edu/>)
- European Language Resources Association (ELRA)/ELDA – Evaluations and Language Resources Distribution Agency (<http://www.elda.org/>)
- Open Language Archives Community (OLAC; <http://www.language-archives.org/>)
- Oxford Text Archives (OTA, <http://ota.oucs.ox.ac.uk/>)

Solutions to/Comments on the Exercises

Exercise 1

First of all, you'll probably note that there may be some unfamiliar or almost 'archaic-sounding' labels among the categories. For instance, without looking up the terms or taking a look at the detailed list of texts, would you have known immediately what, for example, such category labels as 'Popular Lore' or 'Belle Lettres' refer to? Religion, with 17 texts, also seems to play a fairly dominant role. Do you think this reflects modern-day practices/interests, and illustrates contemporary language in a suitable manner?

When comparing the information in the manuals, you'll hopefully spot that the composition of the different corpora is roughly modelled on that of the Brown Corpus, with only small variations in categories and numbers of sample texts.

The slight cultural difference here is that what is originally labelled "Adventure and Western Fiction" in Brown has been adapted to "bush fiction".

Exercise 2

If you, for instance, compare the categories of the SEC to those of the Brown Corpus, you should be able to see that there are certain parallels in that, for example, both corpora try to cover press materials, such as reportage and commentaries, religion, literature, and learned materials, to some extent, although of course the 'angle' is different because of the differences in the medium. For instance, the treatment of the topic of 'Religion' (D) in the Brown Corpus is generally of a more scholarly or esoteric nature, whereas the category 'Religious Broadcast' (E)

in the SEC purely consists of religious services, rather than scholarly discussions of religious issues, and category K (General Fiction) in the Brown Corpus consists of fiction texts treated as texts to be read, whereas 'Fiction' (G) in the SEC is perhaps unusual in the sense that it covers written materials that are simply presented as read aloud, rather like modern-day audio books. There are, however, also some categories that are exclusively reserved to one type of medium, such as, for example, the category 'Dialogue' (J) or the two 'Lecture' categories (C & D) in the SEC, where there's no direct written counterpart in the Brown Corpus, although they both roughly correspond to the 'Learned' category of the Brown. We can also see that the lectures in the SEC are further subdivided into a more popular/populist type (C), as well as a more specialist/scientific one, whereas the 'Learned' category in the Brown Corpus seems to be aimed almost exclusively at a more specialist audience. 'Poetry' (H) in the SEC presents an interesting case because this really covers the way poetry should be experienced, that is, as a literary art form that only really works if one can hear it, although, of course, we don't have to go to a poetry reading to be able to enjoy poetry, but instead can also 'read it aloud' in our heads. Overall, and based on more modern ideas of what a spoken corpus, such as the spoken part of the BNC, should contain in order to make it representative, the composition of the SEC seems to be distinctly lacking. However, if we consider that the circumstances for obtaining and processing spoken materials around the time when this corpus was created were much more difficult than they are these days, and also that one of the main aims was to obtain suitable materials that would allow research into the kind of grapheme-phoneme correspondences relevant for improving *speech synthesis*, then we can perhaps evaluate its composition and suitability for general research into spoken language better.

To faithfully represent spoken language in most of its facets, it's minimally important to include prosodic information, such as pauses, and pitch movements, along with information about who's speaking at which time, whether there's overlap, etc. Although, as in the case of the LLC, spoken corpora are generally represented in orthographic form, all aspects of verbal behaviour, including 'non-words', such as fillers or even laughter, etc., need to be represented as accurately as possible, as they may be relevant to the communication. Representing the data in this way also makes it possible to investigate vocabulary, (morpho-)syntax, as well as pragmatic or semantic features of spoken language in combination with more phonological or interactional ones.

Exercise 3

First of all, it should be immediately clear that the ICE corpora, despite following the '1 million-word model', are very different in their composition from the corpora we've discussed before in that they contain both written and spoken language, with an unusual (positive) emphasis on the latter, as 300 out of the 500 texts are spoken. The other noticeable feature is that there's stronger balance in

the materials in that the spoken parts distinguish between public vs. private or scripted vs. unscripted speech, and that the written parts are differentiated into different levels/abilities and types of writing.

Exercise 4

Essentially, the fact that the BNC is a mega corpus can easily be seen in the sheer number of words it contains. Although there are still 9 times as many words in the written materials, the fact that it already contains 10 million words of spoken language makes it far more representative of the spoken language actually produced in Britain at the end of last millennium, covering a wide variety of public and private contexts. Furthermore, the design principles try to distinguish clearly between which types of language are produced by a majority of speakers of the language, and which ones are predominantly received and are therefore highly influential, despite the fact that few speakers of the language would ever produce them, such as, for example, novels or other pieces of more elaborate writing. One additional highly important attribute of the corpus is the fact that it contains a very large amount of *meta-information*, i.e. information about who produced which type of document(s) and in which contexts.

Note

1 At the time of writing, unfortunately the link to the manual was broken.

Sources and Further Reading

- Davies, Mark. (2010). The Corpus of Contemporary American English as the First Reliable Monitor Corpus of English. *Literary and Linguistic Computing*, 25(4).
- Kennedy, Graeme. (1998). *An Introduction to Corpus Linguistics*. London: Longman.
- Leech, Geoffrey, Myers, Greg, & Thomas, Jenny. (Eds.). (1995). *Spoken English on Computer*. London: Longman.
- Meyer, Charles. (2002). *English Corpus Linguistics: An Introduction*. Cambridge: CUP.
- WordBanks. (2009). Accessible at http://wordbanks.harpercollins.co.uk/Docs/WBO/WordBanksOnline_English.html [last accessed 10-Nov-2013].
- Xiao, Richard. (2008). Well-known and Influential Corpora. In Lüdeling, A. & Kytö, M. (Eds.). *Corpus Linguistics: An International Handbook*. Berlin: DeGruyter.

3

Understanding Corpus Design

3.1 Food for Thought – General Issues in Corpus Design

In this chapter, we'll raise some issues that are often heavily debated by experts in corpus linguistics, but have as yet unfortunately not been solved to any degree of satisfaction. Most of these tend to be related to the construction of large general corpora, though, something a short textbook like this cannot really teach you, so that we'll focus mainly on creating your own, specialised, and generally much smaller corpora. The main reason why we nevertheless need to discuss these points here is that you ought to be aware of the possible shortcomings of corpora in order for you to be able to anticipate any potential problems for your analysis or classroom use. Apart from this, though, we also want to develop an understanding of what (electronic) texts really are and how they can best be stored on the computer to facilitate analysis and exchange of data.

As we've seen before, the very first electronic corpus, the Brown Corpus of written American English, contained 500 (written) text samples of 2,000+ words, and its size and composition set a standard for the compilation of new corpora for many years. For a long time after the publication of the corpus in 1964, many people thought that 1 million words of text represented a general-enough sample to provide sufficient information about the makeup of the English language. However, over the years, researchers have realised that even this type of size and stratification may not be sufficient for performing certain tasks – such as research in lexicography or collocations (see Chapter 10) – efficiently, and hence started devising ways of creating a variety of different types of corpora, representative of

both spoken and written language, and of varying sizes, ranging from very small specialised corpora to some that comprise many millions of words.

3.1.1 Sampling

One very important issue in putting together (*compiling*) a corpus is to determine what exactly should go into it. This may depend on a variety of factors, such as availability, the potential for obtaining permission for copyrighted data, how many people are actually working on creating the corpus, etc. For building corpora of older forms of language – such as Old or Middle English –, natural limitations exist in that there are limited numbers of texts available, as well as the fact that these obviously can only exist in written form. For modern corpora, however, and especially with the advent of recording technology, the choice of materials has become much more difficult. Here, we need to consider how we want to obtain our data in the first place. Do we still want only written language, such as for the first generation corpora, or do we want to include spoken language in phonetically or orthographically transcribed form, too? Or do we want a fully-fledged spoken corpus that will never be published in any general written form?

In terms of what should be included in a corpus, Atkins et al. (1992: 5) make a highly useful distinction between the notions of *production* vs. *reception*, in other words what language users *say* or *write*, as opposed to what they *hear* or *read*, and also argue that “if the corpus is to be a true reflection of native speaker usage, then every effort must be made to include as much production material as possible”.

The BNC, as an example of a modern mega corpus, attempts to strike at least some kind of balance between spoken and written materials, although the percentage of spoken materials (10%) is still rather low, something which possibly exemplifies an unfortunate continuing dominance of written language in linguistics. On the other hand, maybe keeping the amount of spoken data in the BNC relatively low was actually not too bad an idea, since transcribing spoken language is an expensive and time-consuming business, and one where corpus compilers often take too many ‘shortcuts’. In the case of the BNC, this can unfortunately be seen rather clearly in the quality of some of the transcriptions, where, for example, many apostrophes have ended up in the wrong places or gone missing, so that some plural markers are turned into a genitive {s}, or the contraction *we’re* turned into *were* at least 6 times within a single dialogue (<bncDoc id=D96>).

Within the written section of the BNC, there’s a 75:25 balance between ‘informative’ and ‘imaginative’ prose, where the latter also includes a certain amount of ‘written-to-be-spoken’ materials, i.e. speeches, plays, etc. The spoken part consists of a ‘context-governed’ section, sampled from public recordings, etc., and a ‘demographic’ one, comprising recordings made by private individuals who carried tape recorders with them for a period of two days, respectively.

The sampling distribution of the COCA (Davies, 2009) is different from that of the BNC in that it includes language samples from every year, starting in 1990, and, as it’s a monitor corpus, these keep on getting added to continually. The

number of words added per year is 20 million, “evenly divided between spoken, fiction, popular magazines, newspapers, and academic journals” (Davies, 2009: 160). In other words, the COCA contains 20% spoken and 80% written language, which, theoretically, already represents an improvement over the 10% spoken in the BNC. However, a cautionary note is in order here, as the spoken materials in the COCA actually consist of “[t]ranscripts of unscripted conversation from more than 150 different TV and radio programs” (Davies, 2009: 161), which may be more artificial in nature than many of the demographic materials in the BNC, due to the public setting in which these exchanges occur. Thus, although they’ll contain most of the ‘words’ uttered by the speakers involved, being transcripts that were not created with explicit linguistic purposes in mind, they may not be completely representative of spoken language because they possibly contain corrections, or omissions of typical spoken language features, such as fillers. For instance, the spoken section at the time of writing this book (comprising data from the years 1990–2012) only contained 1,352 instances of the (potential) fillers *uh*, *uhm*, and *erm*, where at least some of the instances marked as *uh* were also followed by *hm*, which usually indicates consent, rather than being a filler, and is often spelt *aha* instead. Now, considering that the data for these 22 years should contain 88 million words, it’s hard to imagine that in fact so few fillers, which are a very normal part of everyday language, should be present. Apart from this, there’s unfortunately still a 20:80 imbalance in the corpus data, which clearly ‘over-values’ the written part.

3.1.2 Size

The size a corpus ought to, or can in fact, have depends on a few different factors. First of all, as we’ve already said, there may be limitations in terms of the amount of material that is available, in which case it may be necessary to be content with whatever data one can obtain. Apart from ‘natural limitations’ – such as for corpora of older variants of language – some limitations may be imposed by funding. This often raises the issue of *quantity* vs. *quality*, for which there seems to be an unfortunate tendency towards the former at the cost of the latter, especially for the larger corpora of 100 million words, such as the BNC or the ANC. On the other hand, though, if a corpus is too small, it may not be very useful for general purpose research because the amount of data needed to conduct research into, for example, *collocations* (the habitual co-occurrence of words; see Section 10.5) apparently increases exponentially (c.f. Ooi, 1998: 55–56) with the length of the n-gram to be collocated (see Chapter 10 for more details on these concepts). The more domain-specific the research interest is, the smaller the corpus can be because often it is only necessary to extract specialised vocabulary or constructions from it in such cases. Sinclair (2005) presents some interesting rough approximations on how large corpora need to be to investigate different phenomena, such as multi-word combinations.

3.1.3 Balance and representativeness

Two further issues in the compilation of corpora are those of balance and representativeness. In principle, the former only applies to corpora for general use, though, as it's often assumed that these ought to provide an equal amount of materials from many different genres or areas of relevance that allow us to investigate a representative and realistic sample of the language and draw more or less universally applicable conclusions. Obviously, this is an aim that's very hard – if not impossible – to achieve. In order to do so, in the past corpus compilers have frequently resorted to using samples of approximately 2,000 words from different texts and genres to achieve balance, but both that number and 'cutting out' part of a text appear fairly arbitrary, and may in fact make such samples somewhat unrepresentative of the texts they're supposed to represent as a whole. For instance, while the beginning (Introduction) of a research article may be very similar to its end (Conclusion) in that both discuss the background and aims of the article, the 'middle parts' that describe the actual research are normally very different, and therefore arbitrarily picking either text from the beginning/end or the middle will potentially provide a very skewed picture of the language of research articles.

Sinclair (2005) even argues explicitly against the idea that it's worth attempting to be too precise in selecting such equal-sized samples:

There is no virtue from a linguistic point of view in selecting samples all of the same size. True, this was the convention in some of the early corpora, and it has been perpetuated in later corpora with a view to simplifying aspects of contrastive research. Apart from this very specialised consideration, it is difficult to justify the continuation of the practice. The integrity and representativeness of complete artefacts is far more important than the difficulty of reconciling texts of different dimensions.

Representativeness, albeit also difficult to measure, may be more easily achievable, especially for domain-specific corpora or limited fields of investigation, because often there are relatively clearly definable criteria for what represents a certain genre of text or domain. As we've seen earlier, though, the composition of the first 1-million word corpora roughly consisted of samples of the 16 genres listed in Table 2.3, and it's quite debatable whether those could be seen to be representative even of written language only, let alone give us any indication of what their spoken 'counterparts' may be like. Exercise 1 will hopefully have enhanced your awareness of this issue.

3.1.4 Legal issues

Deciding what exactly you'd like or need to include to make it useful isn't the only thing you need to bear in mind when constructing your own corpus. There are also a number of legal points you ought to consider when making decisions about which data to incorporate. The most important one for written or multimedia data (e.g. transcripts of televised materials or radio broadcasts, etc.) is that

of *copyright*, which may well differ from country to country, so that it's difficult to provide definite rules. In the US and EU, the general rule for written published works is that the copyright expires 70 years after the death of an author, unless it has been transferred to someone else (e.g. the author's heirs). In the US, works published before 1923 also no longer enjoy copyright protection. In other countries around the world, the situation may either be handled in a more relaxed or, in contrast, even harsher way, so it's always advisable to enquire about the exact copyright situation of the country in question, especially if you later want to make your corpus available to other researchers around the world. Some countries also recognise the concept of *fair use*, which allows relatively insubstantial parts of copyrighted materials to be used for purposes such as research, education, review, etc. (Wikipedia: Fair Use), although, in practice, this will probably not allow you to include sufficient amounts of text or other materials in your corpus.

Although it's become common practice in recent years to use materials from the internet (see Section 4.2), according to Sinclair (2005), this is in theory also a problematic issue, as at least “[...] under UK law, publication on the internet confers the rights on the author whether or not there is an explicit copyright statement”. In general, however, people do publish on the internet in order to make their materials easily available for others, so we can probably assume that most authors would possibly not object to having their writings used in a corpus, apart from the fact that we could also argue that web pages tend only to be used in *derived form* for language analysis purposes; in other words, when using web pages as (part of) a corpus, we're not really interested in the original HTML code that may also contain other multimedia content, but in fact only the language that's been made publicly available. Nevertheless, when using other people's materials in this way, we should, as far as possible, try to obtain permission from the authors, or, at the very least, be prepared to remove certain parts from our corpus upon their request if they happen to come across their data in the corpus and object to our using it.

When collecting spoken materials that have not been pre-recorded by someone else, it's also important to note that, in many countries around the world, it's in fact illegal to record someone *surreptitiously*, i.e. without first obtaining their consent. It's therefore advisable to, whenever possible, let everyone you're planning to record sign a consent form in order to avoid any legal complications later, as well as to give them a chance to refuse in the first place.

Having covered the basic theoretical and legal aspects of creating corpora, we'll now turn to the structural nature of texts and other associated properties that may exist in the form of *meta-data* (e.g. additional information about the text, etc.).

3.2 What's in a Text? – Understanding Document Structure

When we read ordinary printed documents, such as books, we mainly concentrate on their text content, and often – though by no means always – tend to ignore

that they may in fact contain a number of additional pieces of information apart from the main text. These occur in other parts of the document that precede or follow the main section, and are generally referred to as *front and back matter*, at least for longer documents.

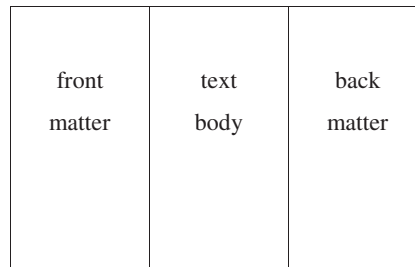


Figure 3.1 Illustration of basic document structure

The content of these sections represents *meta-data*, i.e. additional data about the text, but does not form part of the original text. An example of the meta-data that you encounter in everyday life would be the imprint page inside the front matter of a book, where you find the title of the book, the author, the publisher, edition, year of publication, its ISBN, the typeface and size it has been set in, as well as many other types of information that are mainly independent of the content per se. Another type of meta-information is represented by a table of contents (in the front matter) or an index (in the back matter) of a scholarly book, where the meta-information serves as a kind of navigational aid in accessing individual parts of the book, and where the information is clearly linked to the content – or organisation thereof – itself.

Although all this is interesting information which is conventionally represented inside the document, as well as affecting its *pagination/layout*, it generally does not form part of the *meaning potential* of the text itself, which is what we're usually most interested in when we analyse texts from a linguistic point of view. Thus researching or making use of meta-information for instructional purposes normally doesn't make much sense because it represents language data (in the widest sense) from highly limited/restricted domains. What it may, however, allow us to do is to make a conscious choice on how to restrict the language samples we may wish to analyse to, for example, a specific period in the historical development of the language, or the style of a particular author, etc. We'll see examples of how this may be achieved later on when we'll for example create *subcorpora* from the BNC in BNCweb (see Section 9.3.2).

3.2.1 Headers, 'footers' and meta-data

In 'linguistic' data, such meta-information is often stored either externally in a different file or database, or inside the document itself. In the latter case, it's usually

contained in something called a *header* (cf. Leech et al., 2000: 13). The text itself can then be found in the *body* of the document. Some document formats clearly separate the header and the body, whereas others don't. For example, in *HTML*, the language web pages are written in, all the meta-information is contained in the appropriately named `<head>` tag, while some of the literary texts we'll download and work with later contain a header at the beginning of the document that's not clearly marked off from the rest of the text. Thus, in order to skip/remove it, you'll need to search for the first chapter heading or some similar indication of where the actual text body starts. We'll practise doing this in Section 4.4.3 of the book.

In some files, there may also be some additional information that appears after the main body of the text (which we could correspondingly refer to as a '*footer*'), so that it's best to check the beginning *and* the end of a text for information that's not part of the main text of the book.

Let's get a first impression of this separation of texts into meta-data and content as an exercise by looking at the source of a web page.

Exercise 5

First open http://martinweisser.org/pract_cl/online_materials.html, the page containing the online course materials, in your favourite web browser.

Then, if you're using Firefox or Google Chrome, or Konqueror on Linux, press 'Ctrl + u' to display the HTML source of the document. In Safari for Windows, you need to press 'Ctrl + Alt + u', while on the Mac, you can press 'alt/option + ⌘ + u'. In other browsers, such as Internet Explorer, you may need to find an item on the 'View' menu that allows you to see the page source, as there may not be a shortcut defined.

Another option for accessing the HTML source code in all browsers is generally to trigger the context menu by right-clicking (Windows/Linux) or tapping with two fingers in a blank area of the page (Mac), and then selecting 'Show Page Source', 'View Document Source', or any other relevant option.

For the moment, simply try to identify the 'head' and 'body' sections of the page and see whether you can possibly also understand which type of meta-information the different parts of the header may relate to.

Don't be alarmed if you see a lot of coding inside *angle brackets* (`<...>`) that you don't understand yet. We'll learn more about this a little later.

3.2.2 The structure of the (text) body

Just like we frequently ignore front and back matter to some extent, we may also ‘overlook’ – i.e. not consciously pay attention to – the fact that texts have a specific organisational structure and are generally – or at least may be – made up of chapters, sections, sub-sections, and paragraphs. Among the paragraphs, we also encounter one special type, that of the *heading*, which acts as a kind of guide to the particular chapter or (sub-)section the individual paragraphs occur in.

Headings fulfil multiple functions in a text. The first of these is that they act as a means to reflect the hierarchical structure and logic of the text. In other words, they illustrate how the author has chosen to (best) organise the material under discussion. In the simplest case, such as in a novel, this may just involve breaking the text into chapters and labelling them *Chapter 1*, *Chapter 2*, *Chapter 3*, etc., with a slightly ‘more advanced’ situation being that the individual chapters may also be given a title. The function of the title is in some way to act as a summary, or some other form of indication, of what the individual chapter or section contains. For any kind of scholarly publication, having titles for chapters, sections, sub-sections, etc., is in fact the norm, and here the title not only acts as the ‘summary’ but its content is often also repeated somewhere in the immediately following paragraph – usually even the very beginning, if we have a so-called ‘topic-sentence’ starting the paragraph.

This of course introduces certain redundancies in terms of an analysis of the language used because some of the vocabulary may be repeated without really contributing any additional meaning, and we might therefore sometimes want to get rid of headings for some types of vocabulary analysis. On the other hand, this very redundancy may help us to classify – or even identify the exact genre of – a text better. This is because, due to their occurrence both in headings and the text, some of the key words that we find in the headings may occur with a higher overall frequency in the text, and thus help us to ‘summarise’ the content. We’ll explore ways of investigating this phenomenon in Chapter 9.

Therefore, it really depends on the exact purpose of our analysis, and on our own awareness of these features, whether or not we might want to keep or remove headings. Incidentally, the level of redundancy would increase even further if we analysed the whole text, including the front matter, because there the headings might show up once more inside the table of contents (TOC) of the document, where their meaning is ‘purely’ to serve as a navigational aid by listing them side by side with their respective page numbers. On yet a different level, e.g. when analysing student or professional writings, the presence or absence of a suitable number of headings – however that could be defined – may well present an evaluation criterion as to the proficiency of writers and their ability to deal with a topic efficiently enough to present it in a logically structured manner, so it may well be worth investigating.

Now that you already have a pretty good idea of what types of features you might encounter in a text and how this relates to aspects of its organisation, as

well as issues that may affect its analysis, let's move on to finding and processing some data in electronic form as a first step in laying the foundation for creating your own corpus and analysing it.

3.2.3 What's (in) an electronic text? – understanding file formats and their properties

Although we all encounter a number of different file formats containing text on a daily basis while using the computer, many people generally tend not to be aware of the fact that the text contained in these files may not always be easy to process. The reason for this is that it's often stored in a particular *proprietary format* that's only 'understood' by programs designed for dealing with this particular type of file. It's thus often only if we happen *not* to have such particular programs installed on our computer that we actually realise that there's anything special to these types of files. Why we might need such special programs for displaying the text is because we often want to be able to render it with special types of *formatting*, such as *italics*, **boldface**, etc., use a particular *layout*, or that we want to be able to generate a table of contents automatically in a word-processing application, where this is based on the headings inside the document. Any document that purely contains text (plus potentially some form of *markup*, see Chapter 11), is generally referred to as a *plain-text* document and should be readable (and writable) with any basic text editor (see Section 4.2.1 for a more in-depth discussion of these).

Some of the most common file formats containing text which may be of interest to us here are listed in Table 3.1, together with common extensions and properties. For most of these types, examples are also provided of what these formats would look like when viewed as plain text outside of the programs that are normally used to generate or read them in the online materials at http://martinweisser.org/pract_cl/file_formats.html.

Table 3.1 Common file formats and their properties

<i>Format</i>	<i>Extension</i>	<i>Properties</i>
plain text	<i>.txt</i> or no extension	no formatting, just text, possibly with line breaks
web formats	<i>.htm(l)</i> (HTML), <i>.xml</i> (XML)	plain text with additional information contained in tags; rendering not based on page layout, but on instructions contained in the <i>tags</i> , possibly in connection with a specific <i>style sheet</i>
'graphical'/page description formats	<i>.pdf</i> (Portable Document Format), <i>.ps(.gz)</i> (Postscript)	the position of all elements on a page is described in a specific format that can (only) be rendered by an appropriate reader
proprietary formats	<i>.doc</i> , <i>.docx</i> (MS Word), <i>.wpd</i> (WordPerfect), etc.	text export is possible, but may contain additional line breaks, based on the page layout
		usually stored in a specific binary format; export through optional filters

To develop a better understanding of how suitable different types of documents in their original formats may or may not be for corpus analysis, let's do another exercise.

Exercise 6

Use the links in the online materials on the page 'Understanding File Formats & Their Properties' to go through each of the examples of different types of 'text' documents above and see how easy/difficult it is to identify where the actual text is.

If you do have the relevant programs installed on your computer, you can also try to create more complex versions, containing more text and formatting, of the different document types yourself and then investigate them.

Exercise 6 has hopefully already alerted you to the fact that it isn't easily possible to just use any document that we can read in some way on the computer equally well as a source for our corpus-linguistic analysis, simply because it contains text. Instead, what we've seen is that it's often also necessary to understand the nature of how that text has been produced to some extent, or which particular features the program that was used to produce the text may offer. In addition to this, if we want to be able to exchange documents efficiently, we also need to develop a basic understanding of how exactly text may be represented on the computer, which is what we'll do next.

3.3 Understanding Encoding: Character Sets, File Size, etc.

In our discussion of file types, and in Section 2.3 when we discussed issues of encoding for diachronic/historical corpora briefly, we've already seen that not all forms of textual representation are equally useful for corpus analysis. For instance, we saw that if we want to treat a Word or PDF document as a corpus file, we first have to extract its textual contents to a plain-text file in order to be left with any amount of usable text. The reason for this was that the formatting and layout options contained in documents of these types are usually not stored as plain text, but instead are often binary coded, and therefore we have no (easy) way of dealing with these.

3.3.1 ASCII and legacy encodings

The problem we have with encodings is one of a similar nature, only that in this case it doesn't have anything to do with proprietary formats, but rather the way

that individual characters are physically represented inside the computer, which is as a special type of number. For a long time, most of the basic Latin(-derived) characters used to be represented on the computer as sequences of bits up to one single byte (= 8 bits) only. This made it possible to represent and store up to either 128 (7 bits; 2^7) or 256 (8 bits; 2^8) characters, where each character was assigned a specific number in the original *ASCII* (*American Standard Code for Information Interchange*; 7-bit) or Latin1 (8-bit) character sets or their derived forms. Within these character sets, the ‘ordinary’ letters of the Latin alphabet are encoded by a single number each, but with a distinction between upper and lowercase ones, where the uppercase ones range from position 65 to 90 and the lowercase ones from position 97 to 122. Punctuation marks, numbers, and mathematical operators occupy the intermediate ones between 33 and 63.

However, there used to be major differences between the representations of the upper ranges of characters and the individual requirements for them for the different, simpler, Western character sets, and it proved impossible to even attempt to store the huge numbers of characters necessary for writing in non-Western languages, such as Chinese. This is also why, even if we’re producing a document that only contains characters of the English alphabet, plus a few special characters, such as special types of (typographic) quotation marks, etc., and save this in one of the so-called *legacy encodings* (e.g. ANSI/Latin1 for European languages, GB5 for traditional Chinese, GB2312 for simplified Chinese, EUC-KR for Korean, to name but a few), someone reading this on a computer that uses a different character set by default may see some, or sometimes even all, characters misrepresented.

Exercise 7

To understand the issues caused by using different character sets, and viewing them with the wrong encoding, better, let’s take a look at the page on encodings in the online materials at http://martinweisser.org/pract_cl/encoding.html.

3.3.2 Unicode

In order to solve the discrepancy between character sets, first additional double-byte character sets were introduced, but eventually the notion of one single character set for all processing needs was put forth and implemented. This ‘umbrella’ character set is referred to as *Unicode*, which, despite the unifying attempts, still exists in a number of different flavours that use fixed or variable byte length to encode thousands of characters. The particular flavour we’ll want to use for this course is a variable byte encoding called *UTF-8*, which stores characters in up to six bytes, but generally has the advantage that the basic Latin characters appear at exactly the same positions as in the simple encoding discussed above, so that

opening an English text in UTF-8 would essentially be the same as opening it in ASCII or Latin1. This character set is also well supported by many different programs, including browsers, so that it makes information exchange including characters from many different languages, as well as special characters, such as phonetic symbols, etc., much easier, thus also facilitating the creation of multi-language corpora.

3.3.3 File sizes

Plain-text files in general tend to be much smaller than other files because representing characters, even if some of them may take up six bytes in UTF-8 in some cases, does not require much space. As pointed out in Section 3.3.1, for most characters appearing in an English text, one single byte will be enough. Shorter plain-text files are therefore generally very small, sometimes even less than a kilobyte (kB; 1kB = 1024 bytes). Based on a small selection of four literary files we'll download and analyse later, I tested the approximate ratio of words per kilobyte, which appears to be around 180, so that per 1,000 words we may want to collect for our own corpora, we'd probably require about 5.5 kB of text. In order to collect this much – or rather, little – text, we'd probably need the equivalent of about 2.4 to 3 pages of scanned text, as the number of words in texts roughly varies between 250 per page for double-spaced texts of average font size (i.e. 12 pt) in printed books such as novels, and maximally about 600, which I found in a mono-spaced article from a scientific journal where the print size was around 10 pt.

This text size doesn't increase dramatically, even if we add other types of enriching information – generally referred to as annotations (see Chapter 11) – to our files, provided that these annotations are also stored in plain-text form, and aren't excessive. To verify this in a very crude manner, I ran another test by comparing the raw and annotated files for my home page (in HTML), one dialogue annotated on a number of linguistic levels by one of my own programs, and one dialogue from the BNC, which contains a rather large amount of meta-information in its header and extensive word-level annotation. It turned out that the web page was only 1.2 times as large as the original raw text contained in it (6 kB: 7 kB), the first annotated dialogue containing minimal meta-information and my own annotations was only approximately 3 times as large (2 kB: 6 kB), but the BNC file was more than 11 times the size of the original source text (30 kB: 340 kB). However, even though the latter two values may already seem relatively high, this is still fairly small compared to the file sizes of some of the proprietary document formats we encountered in Section 3.2.3. We'll explore plain-text-based file formats, such as HTML and XML, that may contain such markup further in later sections, as well as discussing their use(fulness) for linguistic annotation/analysis.

Now that you have a basic idea regarding the formats and encodings a text might come in, and the issues involved in being able to work with them, we can move on to finding out how we can obtain our own texts for analysis purposes. As we go about investigating various sources, we'll also learn more about making the

texts contained in the different types of documents more ‘amenable’ to analysis. Perhaps one more thing is worth mentioning before we move on, though, which is that sometimes, if you use special programs or corpus data that other people have collected, you may occasionally encounter files (or file types) with uncommon extensions that are not recognised by any other programs. Due to this fact, if you simply try to open these files by (double-)clicking on them, you’ll usually get some form of message saying that your operating system doesn’t recognise this particular file type, and asking you to identify a program to open the file with. In most cases, this should not be a problem, though, because, as pointed out earlier, most corpus data is stored in some form of plain text, so you can always try opening these files in your editor first. In case this fails, you then either have the choice of trying to find the program used to create the files or, alternatively, simply not to use these files.

Solutions to/Comments on the Exercises

Exercise 5

Looking at an HTML page for the first time may be somewhat bewildering, due to the strange bits of code that are generally contained within angle brackets (such as e.g. `<p>`), which seem to have nothing whatsoever to do with what the text is all about. Furthermore, most HTML page sources do contain a fair bit of information before the actual start of the text, which is signalled through the `<body>` tag, so most of what precedes it should be considered ‘non-text’. Don’t worry, though, if this all still looks like a foreign language to you – you’ll soon learn to understand this better, at least as far as you need to in order to be able to make use of the text contained inside an HTML document.

You may well notice that the header (`<head>` in HTML) generally doesn’t really contain any part of the text itself, but simply stores meta-information, i.e. information about the text or relevant to how the browser is supposed to handle the page, for example, in which way to display it. Header elements may for instance be the page title (contained in the `<title>...</title>` tag, where the forward slash indicates the end of the tag in the closing part), which then appears in the title bar of your browser, or possibly meta-information (`<meta>...</meta>`), such as the text (content) type or encoding/character set (*charset*), as well as style (sheet) information/references (`<style>...</style>` or `<link rel="stylesheet" href="/corpus.css" type="text/css" />`, in our case) responsible for some of the page formatting.

Exercise 6

When looking through the examples, you’ll hopefully realise that there are some document formats that allow you to see the text contained in them quite easily,

while others contain too much additional coding describing the layout or formatting of the content to be able to easily detect/identify the textual content. As almost no programs for corpus analysis can deal with documents in ‘graphical’ formats, such as PDF, or proprietary formats, such as MS Word, the only logical choice for working with corpora is to use either plain text or other types of documents that contain minimal or easily recognisable annotations, such as HTML or XML documents.

Exercise 7

This brief exercise should have demonstrated to you how tricky it may be to work with the wrong encoding, especially when our data may contain characters from a number of different languages. However, what you’ve just seen on the exercise page is not only an issue in corpus linguistics, but actually represents a much more prevalent problem on the internet. This is because many people still only produce web pages, mainly through easy-to-use, but badly configured programs, in their own local encodings, which is absolutely fine as long as they only use these pages to ‘communicate’ with other web users in their own country, who are likely to have their computers set to the same *code page*. When those pages, however, are then opened on a computer in another country, and which is set to a different code page, the result may look very similar to, or even worse than, the result we get when we set the exercise to any encoding that is different from UTF-8.

Sources and Further Reading

- Atkins, Sue, Clear, Jeremy, & Ostler, Nicholas. (1992). Corpus Design Criteria. *Literary and Linguistic Computing*, 7(1).
- Biber, Douglas. (1993). Representativeness in Corpus Design. *Literary and Linguistic Computing*, 8(4).
- Biber, Douglas, Conrad, Susan, & Reppen, Randi. (1998). *Corpus Linguistics: Investigating Language Structure and Use*. Cambridge: CUP.
- Kennedy, Graeme. (1998). *An Introduction to Corpus Linguistics*. London: Longman.
- Leech, Geoffrey, Weisser, Martin, Wilson, Andrew, & Grice, Martine. (1998). Survey and Guidelines for the Representation and Annotation of Dialogue. In Gibbon, Mertins, & Moore. (Eds.). (2000). *Handbook of Multimodal and Spoken Language Systems*. Dordrecht: Kluwer Academic Publishers.
- Legal Information Institute. (n.d.). U.S.C.: Title 17 – COPYRIGHTS. <http://www.law.cornell.edu/uscode/text/17>. [last accessed: 21-Nov-2013]
- Ooi, Vincent. (1998). *Computer Corpus Lexicography*. Edinburgh: EUP.
- Sinclair, John. (2005). Corpus and Text – Basic Principles. In Wynne, M. (Ed.). *Developing Linguistic Corpora: A Guide to Good Practice*. Oxford: Oxbow Books, pp. 1–16. Available online from <http://www.ahds.ac.uk/creating/guides/linguistic-corpora/chapter1.htm>.

4

Finding and Preparing Your Data

Having the right kind of data to work with is essential in doing any kind of linguistic analysis. However, even the best and most appropriate data may not always fulfil all your needs, or may contain bits of information or formatting that make it difficult to work with. Apart from this, the programs we can use for such an endeavour are not always able to handle all the data you may think suitable, due to some of the above issues. This is why, in this chapter, we'll first explore where to find some basic materials for language research, and then go on to discuss which types of formats may be suitable for language analysis, as well as how we can try, to the best of our abilities, to bring our data into such a format.

The data we'll be working with here may or may not be similar to the kind of data you'll be interested in working with yourself, but mainly serves for illustrative purposes, so that you can learn the right preparatory and analysis techniques. Once you start compiling your own data later, you'll obviously need to make your own decisions regarding which data exactly suits your research purposes, and also how much to collect in order to get a representative sample that may reflect all or a specific sub-part/genre/text type of the language you're trying to describe. This obviously requires careful consideration and also to develop at least some initial hypotheses about what you'll encounter in which type(s) of data, which techniques to apply, etc. Therefore, this preparatory process should not be taken lightly, especially because corpus compilation and preparation, if done well, is a very time-consuming effort. And thus, the more time you waste on collecting data that's unsuitable for your purposes, the more time you lose for actually interpreting this data in a linguistically meaningful way, which is, after all, still the most important part of the analysis.

4.1 Finding Suitable Materials for Analysis

4.1.1 Retrieving data from text archives

Text archives are internet-based repositories of literary and non-literary texts that have usually been scanned from the original sources and can be retrieved in a variety of different formats or encodings. Many – if not even most – of the texts available through such archives are free of copyright or available under academic licences, but, as pointed out in Section 3.1.4, before you publish anything based on such a text, it's usually advisable to inform yourself about any potential issues, such as how to acknowledge the source of your materials or whether there are any restrictions concerning re-distribution, etc.

Perhaps the two most important text archives for linguists or literary scholars are the Project Gutenberg website (<http://www.gutenberg.org/>), and the Oxford Text Archive (OTA; <http://ota.ahds.ac.uk/>), so we'll here concentrate on finding suitable texts or corpora and downloading them from there. Within those archives, you can find a wealth of language materials from different languages and periods, suitable for a variety of language analysis purposes, ranging from diachronic studies of different texts through the ages to synchronic analyses of individual language periods, although, to some extent, contemporary data may not be available due to copyright issues. Perhaps the main difference between the two repositories is that Project Gutenberg hosts primarily literary texts, while the OTA contains both literary texts and linguistic corpora, where the latter may also comprise contemporary language data.

4.1.2 Obtaining materials from Project Gutenberg

Project Gutenberg is a large repository of texts in many different languages. These texts were/are essentially created by volunteers who scan or type in the materials, thus converting them into an easily readable electronic format, without any special formatting or layout. What this format may look like, and which advantages and issues this form of presentation may have in store for us in terms of learning about language, and its representation in electronic form, is something we'll be exploring throughout this section. For some initial practice, let's start by downloading a text from the Project Gutenberg website and taking a look at it. We'll later do some further exercises using these files as well.

Exercise 8

Open the Project Gutenberg site at <http://www.gutenberg.org/> in your web browser.

Click the 'Browse catalog' link towards the top of the page. The online book catalogue should now open.

Take a look at the different options available for finding books, then restrict the language to English, and finally click on the letter A under ‘Authors’.

Scroll down the list until you find Jane Austen or press ‘Ctrl + f’ on Windows/Linux, ‘⌘ + f’ on the Mac, on your keyboard in order to use the browser’s find functionality to search for the name. Once you’ve found the entry for her, click on the link for *Emma*, making sure that you select the one with the book symbol next to it, rather than the loudspeaker symbol. You’ll now be redirected to a new page with a listing for that book.

Take some time to look at the information provided on this page, especially with regard to copyright on the ‘Bibrec’ (bibliographical record) tab, and then look at the table at the bottom of the ‘Download’ tab listing all the different formats available. You’ll notice that there may be a variety of formats available for different purposes, but the most useful for ours will usually be ‘Plain Text UTF-8’.

Press the right mouse button on the corresponding link. From the *context menu* that will open, select ‘Save Link As ...’ and save the file to your computer or memory stick, possibly changing the name to something more telling than the original file name suggested. Tip: If the name you’ve chosen would ordinarily contain spaces, I’d suggest you replace these by underscores (_) because computers are generally better at handling file names that do not have spaces in them.

Repeat the above process with *Sense and Sensibility*.

Finally, also download the PDF version of the same file. We’ll use this later on to see how we can extract text from a PDF file.

Open either one of the text files and scroll through it to see whether you may be able to recognise anything special about the formatting, layout, etc. If not, don’t worry, we’ll discuss these matters in more detail soon.

4.1.3 Obtaining materials from the Oxford Text Archive

Now that we’ve already downloaded some basic literary texts, let’s explore another archive, the Oxford Text Archive, to see what’s available in general there, and to download a whole corpus for later use.

Exercise 9

Open the Oxford Text Archive page at <http://ota.ahds.ac.uk/> and click on the link to the ‘Catalogue’.

Browse through the tabs in the catalogue a little, and then explore the options for sorting it, for instance listing all free, unrestricted resources first, or by author or language.

Switch the display option to display 100 entries, just to make it easier to see more corpora at a glance.

Use your browser's find functionality to look for the *Uppsala Student English corpus (USE)* under the 'Corpora' tab, then click on the id (2457) on the left.

Look at the information presented on the page, especially about availability, and then download the .zip archive of the corpus from the link provided there.

Once you've downloaded the zip file, unpack it and start exploring the contents.

If you're less interested in literary language, but may be more interested in exploring 'real' linguistic corpora, this type of data may already be more useful for you, especially if your main interest is in learner language.

4.2 Collecting Written Materials Yourself ('Web as Corpus')

Although you can collect other types of written data in various ways, including scanning or retyping, these days it's becoming more and more fashionable to retrieve written materials in the form of web pages (or other formats) from the WWW, so this is what we want to explore next. Obviously there are some problems with this approach due to the format of the data that will be retrieved, which often includes other types of information apart from the main text, but we'll deal with these issues only partially now, and explore the remaining options a little later.

4.2.1 A brief note on plain-text editors

In order to work with corpus data outside of the specialised corpus analysis programs we'll be discussing later, for instance to prepare it for processing or distribution, it's usually best to use a simple plain-text editor, rather than a word processor like MS Word or OpenOffice Writer, etc. Plain-text editors usually save their files as pure plain text, often using the *extension* .txt by default, and the more useful ones also allow you to specify a default encoding (which should generally be UTF-8 these days to ensure exchangeability of data), run sophisticated search-and-replace operations based on regular expressions (see Chapter 6), do syntax highlighting for special annotation formats (see Chapter 11), display line numbers, allow the user to run word counts, or even set up *macros*, i.e. little programs that automate certain tasks, etc. Don't worry if some of these features are still unfamiliar to you. You'll learn to appreciate them more and more once you encounter them in later sections of the book or simply gain more experience in working with such editors.

As most operating systems recognise the extension `.txt` and will automatically open an appropriate built-in editor when a plain-text file with this extension is clicked, I'd strongly recommend you to use this for your own corpus data, at least for data that doesn't contain any special annotations, even if some operating systems, such as Linux or Mac OS X, may not require it, and default installations of Windows will unfortunately also hide known extensions from the user. The latter, however, can easily be fixed by going to 'Tools→Folder Options...', selecting the 'View' tab and unchecking the option to 'Hide extensions for known file types'.

Because many of the built-in editors, such as Windows Notepad or TextEdit on the Mac, are either not powerful enough (the former), or first need to be configured in special ways to handle plain text by default (the latter), I will make some recommendations for editors I consider suitable for corpus processing for Windows, Mac OS X, and Linux below, and also try to explain some of their advantages for basic corpus processing.

On Windows, one of the most easy-to-use, fast, and small editors is Notepad++ (<http://notepad-plus-plus.org/>), which also comes with a number of useful *plugins* that allow it to be extended if necessary. The basic setup already allows the user to set the default encoding to UTF-8, run simple and complex search-and-replace operations including line breaks, display different annotation formats using syntax highlighting, record macros, etc. Another highly useful program which has rather similar functionality, but a different look-and-feel, as well as taking up more space on your disk and being slower to start up, is KomodoEdit (<http://komodoide.com/komodo-edit/>), which in fact is available for all three operating systems. The built-in Windows Notepad lacks many of the features that would be desirable for corpus processing and may also store text in the default encoding for your language setting, which may make the data incompatible for exchange, so I'd definitely not recommend using it.

On Macs, a highly useful and powerful editor that can handle many different plain-text formats is TextWrangler (<http://www.barebones.com/products/textwrangler/>). As far as I can tell, its default encoding is already set to UTF-8, so that it requires no further configuration in that respect. While most of the editors on Windows and Linux have similar menu items, TextWrangler's menus may take more getting used to. Therefore, for example, despite the fact that there's a separate 'replace' menu item, replacing text (at least for the first time) requires you to use the 'find' dialog first, etc. Mac OS X's default editor, TextEdit, first needs to be set up as your text editor as described at <http://support.apple.com/kb/ta20406>, but overall isn't as powerful as TextWrangler, so I'd suggest you only use it if you really cannot manage to install the latter for some reason.

On Linux, if you're running KDE, you can safely rely on the built-in default editor, Kwrite, as it fulfils all needs of the budding corpus linguist, while unfortunately the Gnome default editor, gedit, neither supports regex replacements nor setting encoding. To replace this on Gnome, I'd therefore suggest you install the Bluefish HTML editor, which provides all these options.

The following is a brief summary of desiderata for suitable plain-text editors. They should:

- allow the (default) encoding to be set to UTF-8; ideally also to convert between encodings
- support regular expressions in search-and-replace operations
- be HTML/XML-aware, i.e. allow matching elements and colour coding; ideally allow syntax checking
- support setting line endings to Windows or Linux/Mac OS X format

The more you'll work with text editors, the better you'll hopefully come to understand these features, and will ultimately also be able to make your own decisions as to which editor you prefer, at least if there are multiple options available.

4.2.2 Browser text export

Perhaps the easiest option to obtain texts from web pages is to use your browser's text export function, if one is available. Often browsers will not only allow you to save a web page in its HTML-form(s), but also as a plain-text variant. The quality of this type of export may vary from browser to browser, though. Some browsers will remove the HTML and only leave plain-text content, while others may retain some bits of HTML, such as, for example, email addresses contained in *mailto* links (i.e. those that allow you to fire up an email client when you click on them), etc. As far as the layout is concerned, some browsers will remove almost all layout or structure information, apart from maybe some line breaks, whereas others may indent headings, etc., or insert some empty lines in order to try and preserve at least some of the original layout. It is therefore best to test whatever capabilities your browser has in this respect and download a sample page, which you can then compare with the original. As Chrome and Safari have no support for downloading only the text, the instructions below are split into two sections:

Exercise 10

Open the page on file formats we looked at earlier in Exercise 6.

Firefox and IE:

Click on the 'File' menu.

In Firefox, choose 'Save Page As...', in IE, 'Save As...'.

Under 'Save as type:' (or whichever entry is equivalent in the dialogue box on your operating system), select 'Text Files (*.txt ; *.text)' in Firefox, 'Text Files (*.txt)' in IE for the type. You may optionally also be able to specify an encoding, and if this is available, you should definitely select 'UTF-8' here.

Specify a file name or accept the one provided by the browser.

Chrome and Safari:

Select all the text on the page and copy and paste it into an editor to save it from there. To do so, press ‘Ctrl + a’ on Windows/Linux, and ‘⌘ + a’ on the Mac, followed by ‘Ctrl + c’/‘⌘ + c’, respectively.

This should copy your text to the clipboard.

Open your favourite text editor and paste in the contents by pressing ‘Ctrl + v’ on Windows/Linux, ‘⌘ + v’ on the Mac.

Open the file in your favourite text editor (unless it’s already open) and compare it to the original web page.

One thing you may have noticed when reading through the instructions above is that, as far as using keyboard shortcut keys is concerned, whenever Windows/Linux use the ‘Ctrl’ (Control) key, the same feature is usually triggered by replacing ‘Ctrl’ by ‘⌘’ (command) on the Mac, although the Mac actually does have a key labelled ‘control’. This keyboard mapping for shortcuts seems to be a general principle, so if you need to move between platforms for some reason, you can bear this in mind.

4.2.3 Browser HTML export

Although using the text export function will provide you with most of the relevant data you might want to extract from a web page, as we’ve seen through Exercise 10, it’ll also potentially remove certain types of information, such as the indications for headings (as opposed to proper running text), or even add line breaks that weren’t part of the original text. In some cases, we may actually be interested in particular types of ‘layouting’ or formatting information, which is why we might want to download the whole web page, including all of its HTML markup. Luckily, this time, all browsers provide this functionality directly. In order to do this, you just need to follow the instructions below:

Exercise 11

Click on the ‘File’ menu as before or the  icon in Chrome.

In Firefox, choose ‘Save Page As...’, in IE and Safari ‘Save As...’, and in Chrome ‘Save page as...’ again. For other browsers, find the equivalent function, which will be named in a similar way.

Under ‘Save as type:’, select ‘Web Page, HTML only’ or ‘Webpage, HTML only’ for the type. In Safari (v. 5) for Windows, select the ‘HTML

Files' option, on Mac OS X, select 'Page Source' under the 'Format' option. Also make sure that you can see the extension by unchecking 'Hide extension'.

Specify a file name or accept the one provided by the browser.

Click on the 'Save' button or press the 'Enter' key.

Open the file in your text editor and examine its format. Pay particular attention to whether you might be able to recognise basic textual divisions, headings, formatting instructions, links, etc.

We'll take a closer look at HTML and its conventions a little later. For now, it should just be enough if you become aware of roughly what the original source format and formatting options for web pages look like.

4.2.4 Getting web data using ICEweb

ICEweb is a small application that I've written in order to retrieve a number of web pages for a given domain, and to process them automatically in order to be able to use the raw text for analysis purposes or to create different types of frequency lists, something we'll discuss later in this textbook. The original idea behind its name is that it was to be used to retrieve corpora of web pages that could form a web-based counterpart to the corpora contained in the International Corpus of English (ICE) we discussed in Chapter 2. You can see a screenshot below.

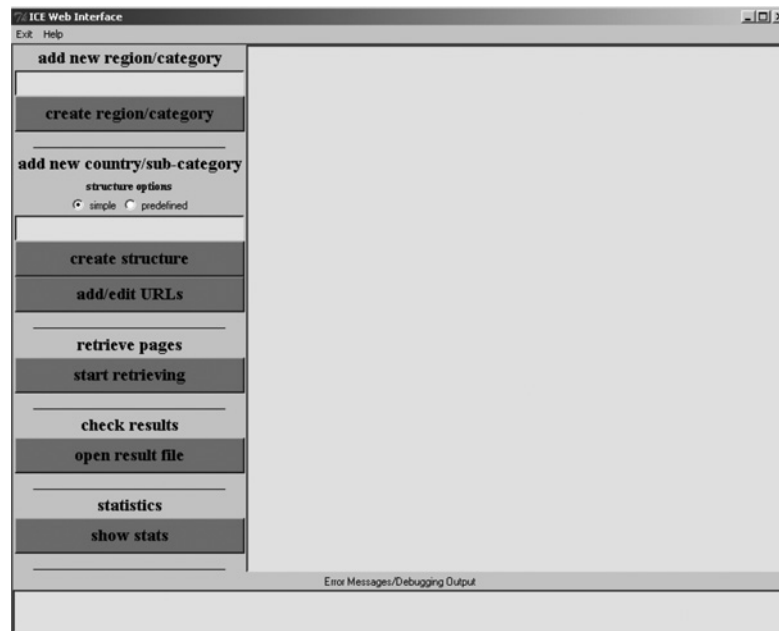


Figure 4.1 The ICEweb interface

To download a copy of the program, go to http://martinweisser.org/ling_soft.html#iceweb, get the zip archive, and unpack it to a folder where you have write access, preferably to your user area, a memory stick, or any folder that's not a system folder.

The program itself is relatively intuitive to use once you've launched the executable, which may be named 'ICEweb_win32.exe', or simply 'ICEweb.exe', depending on which version you're using. Once it's running, you normally start by defining – i.e. typing the name of – a new region (generally a continent) in the textbox in the top left-hand corner and creating a directory for this by clicking the button below it. If you're not collecting data from a specific region of the world, you can always set up a different main category instead, though. Next, you add one or more countries/sub-categories to this region/category and create either a fixed genre/directory structure, or a simple one, by clicking the 'create structure' button for each country/sub-category.

You can then use a web browser to identify and collect the web addresses of pages that you want to download. These need to be specified in a file called 'urls.txt' in the appropriate directory, which you can create by clicking on the 'add/edit URLs' button and selecting the directory. This action will automatically open the built-in editor where you can paste in the web addresses, one per line.

Once you've added enough URLs, use the 'start retrieving' button and you'll be able to select a directory/domain for which you want to collect the pages. The download and processing progress will be reported in the text window on the right, and a number of sub-folders will be created once the pages have been downloaded and processed successfully. You can later explore these folders/files using Windows Explorer, Finder on the Mac, or your favourite file manager on Linux. If you want to see frequency statistics for a given directory, you can also use the 'show stats' button and a new text window will open, presenting you with detailed descriptive statistics. Let's practise some of this with a few web addresses I'll provide below:

Exercise 12

If you haven't already done so, download ICEweb and set it up as described above.

Start the program and create a new category 'linguistics'.

Create a sub-category called 'corpus_linguistics' in the 'linguistics' category folder.

Start the URL editor, and type in the following web addresses: "http://www.euppublishing.com/userimages/ContentEditor/1257173917699/html_download_test.html", 'http://static.wikipedia.org/new/wikipedia/en/articles/c/o/r/Corpus_linguistics.html', 'http://www.ict4lt.org/en/en_mod2-4.htm'.

Save the file and close the editor.

Retrieve the pages. If there are any error messages in the window at the bottom, you can usually safely ignore them 😊, but if any appear in the window on the right, this generally means that you've either got a misspelt URL or that the relevant web page is simply not downloadable for reasons related to the server configuration.

Open Explorer or whichever file manager is appropriate for your system to view the output of the program. You should normally see 3 files ('urls.txt', 'dirIndex.html', and 'index.csv'), plus 4 folders ('html', 'raw', 'tok', and 'frq') if any files have successfully been downloaded. The first of the files is the one you created for storing the web addresses, the second one an HTML page that lists the HTML files that have been downloaded and numbered sequentially in the form of hyperlinks to the original pages, while the last file contains an index of the pages that can be viewed as a spreadsheet. If ICEweb was unable to download one or more files, it'll also create a file called 'download.log'. The individual folders contain the original HTML files, a raw text version, a word *token* file, and a *frequency list* for each downloaded file, respectively.

You can look through the folders to see whether/how many files they contain, but it's generally best to look at each file, apart from maybe 'dirIndex.html' and 'index.csv', through the built-in editor by clicking on the 'open result file' button and selecting an appropriate file. Try this with at least one of the downloaded HTML files and its corresponding text version.

Please note that the extensions for the files containing the word tokens (*.tok*) and frequency lists (*.frq*) are not extensions that are generally recognised by any programs, such as editors, but you'll be able to view them with any text editor if you use the usual file-opening mechanisms. However, if you're working in ICEweb, anyway, it's usually easiest to use the built-in editor. We'll discuss the significance of word tokens and frequency lists similar to the ones generated by ICEweb in Chapter 9, and once you've worked through this chapter, you'll be able to interpret these lists as well, although they, just like the statistics you can view from inside the program, will probably not really tell you much at the moment.

4.2.5 Downloading other types of files

Of course, a corpus downloaded from the web can also consist of other types of text-containing documents, such as those discussed earlier that are in a non-plain-text format, e.g. PDFs, MS Word documents, etc. For those, you'd essentially have to download them manually, but at least we can still discuss an efficient way

of finding the data you might want using a little Google or Baidu (if you're in China) trick.

Exercise 13

Open your browser and navigate to the Google/Baidu page, or use a built-in Google/Baidu search bar if available. Type in 'corpus linguistics filetype:doc', and hit 'Enter'.

Look through the list of results and download one or two that seem interesting to you. Hint: don't forget about the context menu...

Repeat the same thing, only changing 'filetype:doc' to 'filetype:pdf'. We'll soon investigate ways of extracting the text parts from these documents.

4.3 Collecting Spoken Data

As already hinted at through our earlier discussions, collecting spoken data is far more complex than compiling a corpus of written materials (see, for example, also Exercise 2). This is why I can only provide you with some fairly general guidelines here on how to achieve this, apart from pointing out which issues you may encounter when embarking on such an endeavour, and this section won't be accompanied by any exercises, either.

The first thing you obviously need to consider is what type of spoken data you may want to analyse. In general, it won't make sense, apart from actually being illegal (see Section 3.1.4 above for legal points), to simply go out into the street and make random recordings of people. As before, you'll therefore need to consider your data requirements carefully in the light of your particular research questions. Thus, for instance, you may want to record and analyse the speech of native or non-native speakers of a language only, or perhaps make recordings of them in interaction. The first form would then hopefully allow you to investigate one particular population, in whichever recording context you've chosen, while the second should make it possible to compare the two groups (see e.g. Weisser 2001) to see whether their language behaviour differs in any obvious way(s). In the latter case, you may then ultimately be able to come up with fairly objective recommendations for teaching non-native speakers to improve their proficiency. Of course, recordings don't actually need to consist of spoken interaction at all, but may also comprise monologues, such as public speeches, lectures, etc.

Once you've established exactly what type of spoken data you want to collect, perhaps the first important point to observe is that the recordings need to be of sufficiently high quality to make them useful for linguistics research in the first place. Therefore, the recording conditions must be suitable in order to ensure that

there isn't too much background noise, that all speakers can actually be picked up by the recording equipment, that the input signal is strong enough, etc. This is especially important if you're not only planning to analyse *what* your informants say, but also *how* exactly they say it. In other words, if your aim is to conduct any phonetic/prosodic analyses, then you'll also need to use relatively high-quality recording equipment that will at the very least allow you make recordings that comprise the necessary frequency range that is significant in producing different nuances of human speech, i.e. between 80 Hz and 11 kHz (Ladefoged 2003: 19). This may require the use of a high-quality recording device and microphone, but if you're really not interested in this type of close analysis, a simple (and relatively cheap) MP3 recorder may work for you, provided that the above-named recording conditions hold. These days, it actually makes very little sense to use any analogue recording devices because you'd then only end up digitising the data, anyway, to be able to store it on the computer and ideally also distribute it as part of your corpus, and this extra step may not only involve additional time, but also a potential loss of quality.

In comparison to written data, the digital audio files produced through such recordings tend to be relatively large. For instance, one minute of uncompressed speech that allows us to analyse samples of up to 11 kHz, with 16bit-quantisation and recorded in mono (i.e. one single channel) in .wav format, will take up about 2.6 Mb of hard-disk space. This can be reduced to maximally a tenth (around 270 kB) if we use the common .mp3 compression format that MP3 players/recorders employ, still retaining suitable quality for phonetic analysis.

Once you've stored your sound files on the computer, you can start the highly time-consuming process of transcribing them. To do so, it's advisable to use an *audio editor* like Wavesurfer (<http://sourceforge.net/projects/wavesurfer/>) or Praat (<http://www.fon.hum.uva.nl/praat/>) because these programs will not only allow you to carry out phonetic analyses later, but, in the first instance, make it possible to select individual stretches of speech and play them back repeatedly, which makes transcription a lot easier than when you're listening to a continuous stream of sound. My personal preference here is the former, as it's easier to use, but the latter provides you with more phonetic analysis options for advanced phonetics work. Please note that, although you can actually create transcriptions inside the programs and then export them, it's perhaps easier to simply transcribe the data in a text editor, at least if you're only aiming for an orthographic representation (see Chapter 11), so that you'll then need to place your programs side by side and switch from one to the other. If you do in fact want to create a corpus for research into phonetics/phonology, then learning how to transcribe inside these programs and achieving time-alignment for your sound data and transcriptions is something you'll need to learn independently, as space considerations don't allow me to illustrate this here.

We'll discuss some of the specific formats for better representing and annotating orthographically transcribed spoken data later on in Chapter 11, and ways of

identifying such features in Chapter 9, but for now, I just want to mention some specific points you need to bear in mind when trying to render speech more or less accurately in orthographic form. The first of these is that you should always represent what's been said as faithfully as possible; in other words, for example, if someone uses a contracted form, such as *won't*, then you should always represent it as a contraction, and only transcribe *will not* if this has actually been said. As similar point applies to other typically spoken features that are often wrongly assumed to be non-standard, such as the pronunciations of /wɒnə/ or /gɒnə/, which should be transcribed as *wanna* or *gonna*, even if this may be frowned upon by prescriptive teachers or grammarians, because they're really absolutely normal features of genuine spoken language, and 'correcting' them to what is supposed to be 'standard usage/representation' would simply constitute an act of falsifying your data. Further issues may arise in representing other features of spoken language, such as *minimal responses* like *ah/uh/uh-huh* or *mbmlmm-hm* (for *yes* or *yes-like backchannels*), *huh-uh/uh-uh* (for *no*), etc. As the alternative representations listed in the previous sentence show, there may be multiple ways of representing the same thing, and you should not only find a consistent way of representing these features, but also document their meaning, so that other potential users of your corpus will be able to understand exactly what they represent. The same thing goes for common abbreviations of conjunctions, such as *because*, which may variably be represented as *cos* (BrE) or *cuz* (AmE). In some special cases, it may also be necessary to represent other *vocalisations*, such as *hesitation markers* (e.g. *emlermlum*), or specific realisations phonetically, possibly in order to convey emphasis appropriately, e.g. /ði:/ (without following vowel) instead of /ðə/. Other features that may be highly relevant to some types of analyses are pauses, which are frequently indicated in round brackets, e.g. (.3) meaning .3 seconds, or incomplete words, which can be abbreviated by using an ellipsis, e.g. Mon... for *Monday* or *Monica*, etc.

In spoken interaction, usually speaker *turns*, i.e. periods where one speaker talks before the next one takes over, are also indicated, generally prefaced by the speaker name or an identifier in anonymised data. When speakers exhibit *overlap*, which is quite frequent in *multi-party talk* or dialogues, this is often indicated via opening and closing square brackets to indicate start and end, respectively. In some cases, where one speaker simply tries to indicate to the other that they're still 'following', we may also get backchannels that should in fact not be marked as separate turns, but nevertheless indicated, which can be achieved by inserting them inside some form of bracketing, similar to the format for pauses above. As pointed out earlier, we'll later discuss more sophisticated ways of rendering such information, and there far too many different possible conventions to list them all here, so I can only refer you to a few sources, such as Edwards and Lampert (1993), Leech et al. (2000), and Jenks (2011), for further reading here.

Before we conclude this section, a further brief note about data handling and meta-information is in order. As pointed out earlier, not only is it important when

recording informants to obtain their full consent for using the recordings for research, as well as possible distribution, but it is equally necessary to protect their privacy through a suitable anonymisation of the data, for instance using codes to represent personal data, such as names, addresses, telephone numbers, etc., in the orthographic transcript, but also masking these in the audio data if it is to be distributed alongside the transcripts, which it generally makes sense to do if possible. On the other hand, it's usually important to make some personal information, such as the informant's age, sex, provenance, level of education, etc., available to users of the corpus, in order to allow them to conduct research of a more socio-linguistic nature. And, of course, as the compiler of the corpus, you still need to be able to identify and possibly contact your informants later, should follow-up questions arise, so you need to keep a separate file that allows you to look up this information, based on the user codes in your data.

4.4 Preparing Written Data for Analysis

In this section, we'll discuss how to prepare our data for analysis. We'll first explore ways of efficiently cleaning up or *normalising* parts of our data by means of basic *search-and-replace operations*, using a small number of illustrative examples. Later on, we'll move on to learning about ways of *extracting* text data from files that contain formatted text, where of course the same, or at least similar, clean-up operations might be necessary after the main data extraction has been performed.

4.4.1 'Cleaning up' your data

As we've already seen, some of the electronic data we can obtain off the web (or elsewhere) can contain unwanted formatting or meta-information, so we always need to scrutinise whatever materials we download carefully. In some cases, it may just be a question of removing a header or some other types of information, or converting some special symbols into a format that's more suitable for our analyses. Sometimes you may also have to correct artificial line breaks, for example, when you've extracted text from graphical formats – such as .ps or .pdf –, or those inserted by some browsers, as we've seen in Section 4.2.2. We need to do this because these line breaks, as indeed anything that doesn't really form part of our text, may in fact interfere with the processing of the text later and even create a number of problems that could affect the meaningfulness of at least part of your data for linguistic analyses.

What exactly you may need to remove from your data depends on the specific formatting or encoding conventions used for the document. Doing this can sometimes involve a substantial amount of manual work, but might also be as easy as opening the document in your editor and using the search-and-replace functionality in order to replace certain codes from the data.

Exercise 14

Download and open the web page ‘Cleaning Written Data’ (cleanup.html) from the online materials in your browser and also in your text editor.

Call up the *search-and-replace* function, either from the ‘Edit’ menu or by pressing ‘Ctrl + h’, which is the shortcut available in most general text editors these days, even in the non-Windows world. If you’re using TextWrangler on the Mac, you need to trigger the replace operation through the ‘Find’ functionality (⌘ + f) and then fill in the replacement term.

In the ‘search box’, type `&` (including the semi-colon) and in the ‘replace box’ the word `and`.

Click on the button.

Next, change the search term to `–` and the replacement to `-`, and replace all instances again.

You should now have replaced all ampersands and n-dashes in the HTML document. Save it and refresh your browser to see the effect.

Searching and replacing text along the lines of what we just practised above is a very useful semi-automatic means of preparing your data efficiently, but you nevertheless constantly need to be aware of potential errors that might be introduced by replacing *the wrong things* or replacing them in *the wrong order*. For example, you might be tempted to remove all single quotation marks in a text altogether because they ‘interfere with’ the creation of word frequency lists (see Chapter 9), etc., but of course if this is done carelessly and improperly, you may end up taking out all apostrophes, too, in which case you might, for example, end up with a single neuter possessive pronoun form *its* instead of the contraction *it’s* which actually represents two separate word forms! In order to prevent problems like this, it may sometimes be necessary to use the more time-consuming option of replacing all occurrences individually by clicking on and only replacing the item found by clicking on if you’re really sure that you want to replace it.

No matter how much of an unnecessary effort it may seem to you to clean up our data in this way, you should always remember that if you don’t do this, then the data you’re going to use for your analysis later could in fact contain many errors that will most likely skew your results, potentially making them (highly) unreliable.

An additional step you should also take in preparing your data in this way is to keep track of the way in which you adjust the data to your needs. This should ideally be done in the form of a text file that lists all the separate editing steps, and can later provide the basis for part of a manual of information to be distributed with your corpus if you ever plan to release or distribute it. Such information will then help other users of your data to understand better what to expect from your

corpus, or allow yourself to refresh your memory if you should use the corpus data once more after an extended period of time of not working with it.

4.4.2 Extracting text from proprietary document formats

Essentially, the mechanism for extracting text from proprietary formats, such as MS Word or PDF, is always more or less the same, although the exact output and changes you'll need to make to the data later will vary depending on the specific features of the program we're extracting from. The most important thing to remember/look for in such a program is a menu item that either provides us with a 'Save as...' or 'Export' option from the 'File' menu to save the text as plain text, as we've seen for web pages earlier, or some item on a different menu that will probably contain the word 'extract', such as in older versions of Adobe Acrobat or GSview.

Exercise 15

Open the Word and PDF documents you downloaded earlier one at a time and try to extract the text contained in them using one of the mechanisms described above. If you don't have a copy of Word, you can try OpenOffice Writer, which will usually do a very good job of dealing with Word documents, unless they're really complex.

Open the resulting text files and see whether you can identify any other clean-up operations you may need to carry out.

4.4.3 Removing unnecessary header and 'footer' information

Many types of files that you can download, such as the text files we retrieved from the Project Gutenberg website in Exercise 8, do contain some form of meta-information in the form of headers (at the beginning) and/or 'footers' (at the end), concerning copyright, etc. In HTML(-like) files, as we've seen earlier, such sections containing meta-information are clearly delimited by HTML tags, but in other documents, they may be more 'free-form', without any clear indication as to their beginning or end. This is for instance the case with the two literary documents by Jane Austen we downloaded, which contain both header, as well as 'footer', information. As mentioned earlier, this type of meta-information definitely doesn't represent any textual content that we want to keep and analyse, so let's practise identifying and removing this.

The keyboard shortcuts given below for Windows/Linux and the Mac should work in most editors these days; otherwise, if you're using a non-standard editor, you'll need to try and find their equivalents. To make it a little easier to understand

what the shortcuts do, just try to remember that the basic keyboard combinations without pressing the ‘Shift’ key – the one that switches between small and capital letters – will only help you to navigate through the document more efficiently, while combining them with ‘Shift’ will also select the text spans covered by the shortcuts.

Exercise 16

Open your copy of *Emma* in the editor and see whether you can identify the beginning of the text body. Also take a note of the contents of the header.

Once you’ve done so, place the cursor at the very beginning of it, i.e. just in front of the first letter (character).

Press ‘Shift + Ctrl + Home’ on Windows/Linux, ‘Shift + fn + ⌘ + ←’ on the Mac. In most editors, this will select everything from the current cursor position to the beginning of the document.

Press the ‘Delete’ key on your keyboard. The header should now have disappeared.

Next, press ‘Ctrl + End’ on Windows/Linux, ‘fn + ⌘ + →’ on the Mac. Your cursor should jump to the very end of the document.

Scrolling up through the document, find the end of the text body and place the cursor there, keeping note of the ‘footer’ contents.

Press ‘Shift + Ctrl + End’ on Windows/Linux, ‘Shift + fn + ⌘ + →’, which should highlight everything to the end of the document, then press ‘Delete’ again.

Save your document. It should now be in a state where you’ve at least removed all redundant meta-information.

Look through the cleaned copy and see whether you notice any additional formatting-related problems, and think about whether these could possibly be solved via search-and-replace operations. If you identify any, then try them out, each time carefully checking the results before saving the document. If something goes wrong, you can always undo the last step (using ‘Ctrl + z’/‘⌘ + z’), as long as you haven’t saved the document, so there’s no need to panic ☺.

Repeat the process for *Sense and Sensibility*.

4.4.4 Documenting what you’ve collected

As already pointed out above, it’s generally at least advisable to document all the steps you’ve seen necessary in editing your data to make it fit your research

questions and purposes. However, ideally, describing the editing process shouldn't be the only thing you do in this respect; you may also want to retain a certain amount of meta-information about the compilation of your corpus, especially if your plan is to share the data with other people.

Therefore, in addition to the file describing the editing process, you'll probably want to keep at least one extra file that lists the contents of the corpus file-by-file if your data contains materials from different genres, text types or domains, or, as with our web page data, that lists information about where the file was retrieved from, when, what the original file name was if you've changed it), etc. ICEweb already caters for some of that information by keeping a little text database that you can open with Excel or OpenOffice Calc.

As I cannot possibly list, let alone even imagine all the particular pieces of information that may be required for your own research projects/questions, I'd suggest that when you embark on creating your own substantial corpora, you go back to the manuals for similar corpora we investigated earlier on, and try to evaluate whether the documentation for these corpora satisfies all your needs, and, if necessary, adapt this accordingly.

4.4.5 Preparing your data for distribution or archiving

Sometimes, in order to be able to exchange your data with other people, or simply to archive it in some way for backup purposes, etc., it's useful to compress this data into an archive. This not only saves storage space, because texts can be compressed quite easily, but also makes it easier to email files, as one relatively small archive file can easily be sent as an attachment instead of sending each file individually and in its original size. There are numerous compression formats, but the most common one is *.zip*, for which most operating systems not only provide direct support, but also generally have options to create and manage this type of archive from within whichever graphical *file manager* they offer. The general mechanisms employed in such a file manager are in fact very similar for Windows, Linux, and Mac OS X. On the latter two systems, you can also use the command line to perhaps achieve the archiving task even more efficiently, but in order to be able to do that you obviously need to be familiar with such procedures, which are a) more complicated to learn, and b) too extensive to be discussed within the confines of this book.

Essentially, it's generally either possible to select a number of files or even a whole folder and then instruct your file manager to compress these into an archive, or to create the archive first, then open it again, and keep on adding files to it, often per *drag-and-drop*. Of course, the latter mechanism, which is only available on Windows in a simple form, can also be used to add, change, or update files at a later point in time. Let's practise both ways by adding the two texts by Jane Austen you downloaded and cleaned up in Exercise 8 to an archive.

Exercise 17

Open Windows Explorer, Finder on Mac OS X, or whichever file manager you're using on Linux, and navigate to wherever you've stored the two files. Tip: The easiest way to open Explorer in Windows is to hold down the windows flag and press 'E' on the keyboard once, while Finder will automatically be running on the Mac and you just need to switch to it. On Linux, start your favourite file manager, for instance Dolphin on KDE, which is the one I tested the following actions with.

Highlight the two files you want to add to the archive. If the files appear immediately below one another, the easiest way to do this is to click on the first one, then hold down the 'Shift' key, then click on the second one. If your computer is configured to open files via a single click, as is probably the case on Linux, hold down the 'Ctrl' key to select the first one.

If the files are not in consecutive order, Ctrl + click on either one of them, then hold down the 'Ctrl' key (⌘ on the Mac), and click on the other. Tip: These two mechanisms should work on more than two files, too.

Next, use the right mouse button to trigger the context menu or use two fingers to tap if you're using a touchpad on the Mac.

In Windows, move the mouse down to the item 'Send to', which will open another sub-menu, where the first entry should read 'Compressed (zipped) folder'. Clicking on this (using the left mouse button) will create the archive, add copies of the selected files to it, and also allow you to (re)name the archive itself. In Linux (KDE), find the menu item 'Compress', then select 'As ZIP archive' from the sub-menu. On the Mac, choose 'Compress *n* items', where *n* here refers to however many items you've selected to add to the archive. Finder will automatically create a zip archive for you, which, by default, is simply called 'Archive.zip', and which you can then rename to something more appropriate by first selecting the archive and then clicking on the file name once again (avoiding a double-click, which will extract from the archive instead). On Linux, the archive name will probably be based on the name of the first file you selected, but you can easily change that by using the context menu again and choosing 'Rename' from there.

Test this by clicking on the archive itself, once it's been named. All operating systems will just treat it like any other folder that you can copy files (or other folders) to or delete them. Tip: The easiest way to copy additional files into the archive is to simply open another copy of your file manager and then drag-and-drop from there.

Let's also try the other way, if you're running Windows, creating a new archive from scratch. This basically works in a rather similar way, only that, instead of pre-selecting files, you use the right mouse button somewhere


inside a blank space in the folder where you want to create the archive. This will again trigger a context menu, where this time you need to move the mouse cursor down to the ‘New’ option, which will again present you with the option for creating the archive, only that it should appear as the second-to-last option here.

Once you’ve created the new archive, give it any (half-way sensible) name you like and copy the two files in there, either via drag-and-drop or copy-and-paste.

Now that we’ve discussed most of the preliminary issues in corpus design, and seen how we can actually collect and prepare our own data for analysis, we can soon move on to learning how to analyse linguistic data in various forms. However, before we do so, perhaps a final note on record-keeping is again in order. As I stated before, because many of the steps you may need to take in order to produce your corpus may frequently involve making changes to the original data, it’s advisable to document the steps you’ve taken in your preparation as much as possible, to allow both yourself and any other potential users of your corpus to understand the exact nature of the data. Such documentation should obviously also be included in any distribution, provided that there are in fact substantial changes to the original data. In addition, you should also add any meta-information, and finally, perhaps describe both of the above, at least to some extent, as part of a complete *manual of information*. This corpus manual will usually be in PDF format, and from here you can always refer to any additional files for reference if necessary.

Solutions to/Comments on the Exercises

Exercise 8

This exercise should not present much difficulty to you. Perhaps the only real difficulty if you’re not really well-versed in using a computer may be to learn to make use of a few important program features. The first is that you can employ the right mouse button (two-finger tap on touchpads on the Mac) to activate the context menu inside the browser in order to be able to save the material, as otherwise clicking on the link will simply get the text displayed inside a browser window, rather than saved to your computer. Even if this happens, though, it’s not a problem, as you can then use the main browser menu and the ‘File→Save Page As...’ menu in Firefox, ‘File→Save As...’ in Internet Explorer and Safari. In Chrome, you unfortunately have to use the  icon to access the ‘Save page as...’ option, as no menu bar at all exists in this browser. An added complication in Firefox and Safari is that you may need to switch on your menu bar first before being able to do so, but I’d recommend you do this for consistency, anyway. Activating the

menu bar in both browsers can be achieved by using the context menu in the grey area below the title bar of the browser and choosing the appropriate option.

Exercise 9

Again, this exercise should be relatively straightforward to accomplish, although you may need to learn to handle .zip archives if you're not familiar with them yet. If you're using Windows, Explorer has built-in functionality for extracting the data from such an archive, and on Linux and the Mac, (double-)clicking on the file inside the file manager will usually extract the data automatically.

Another thing that you may have been wondering about while exploring the OTA is what some of the abbreviations in the column for 'Availability' stand for. All those that start with 'CC' refer to Creative Commons licences, the nature of which you can explore in more detail at <http://creativecommons.org/licenses/>.

When you explore the contents of the archive, you'll hopefully also realise that additional information along the lines of what I recommended earlier is in fact included in the form of a plain-text file called 'ReadMe.txt' that describes the contents of the corpus distribution, a manual (in Word format), and an Excel file that contains a database of important details related to the corpus files, speakers, etc.

Exercise 10

It's difficult to describe what exactly you're going to see once you've opened the text version in your editor because the output from different browsers varies so greatly, so I'll just provide a few examples here to raise your awareness of different issues.

Firefox (ver. 25 and above), for instance, will try to preserve at least some of the layout and formatting for the page, so the main heading, which is centred on the page in the HTML and has some spacing around it, will be surrounded by two empty lines and indented somewhat. In general, paragraphs or other objects, such tables, will have spacing around them simulated through empty lines. A longer paragraph will be broken into a number of lines, each ending in a *line break*, where the maximum line length seems to be about 80 characters. The table structure is mimicked through tabs, which will simply look like spaces, unless you can get your editor to display them as tabs. Hyperlinks are preserved and rendered in angle brackets (<...>), italicised text surrounded by forward slashes (/.../), and underlined text surrounded by underscores (_..._). The label and number for the exercise at the bottom of the page, which are generated automatically in my HTML, are deleted, and the horizontal rule below this paragraph is represented as a series of hyphens.

IE (ver. 11) will strip out almost all of the formatting, even removing the spaces between the cells in the table, and thus joining some words that should not be joined, just like in our example of the 'fake compounds' in Section 1.1.2. Strangely,

though, it simply ‘lists’ all the row headers (*Format*, *Extension*, and *Properties*) as separate paragraphs before the remaining cells of the table, which are then presented as individual paragraphs from left to right and then top to bottom, but without any additional spacing between them. It therefore treats the header row conceptually very differently from the rest of the table. It also removes all the hyperlinks and the horizontal rule, as well as the generated content, completely. Unlike Firefox, IE also takes the page title from the <head> section of the HTML page and lists it, with a slight indent of one space, before the first heading. As I generally use (more or less) the same text in the <title> tag of my pages, this effectively duplicates the text of the heading inside the saved text version. Just like Firefox, it also breaks longer paragraphs into shorter lines, thus adding extra line breaks that do not form part of the original text.

The copy-and-paste versions retrieved from Chrome (ver. 31 and above) and Safari (ver. 5 on Windows, ver. 8 on the Mac) seem to be almost identical in that they generally strip all the formatting and hyperlinks from the document and simply leave the text with a minimal degree of formatting. In the case of our test page, all regular paragraphs, including the heading, were rendered with two line breaks following them, apart from the last one in the document. The table here is presented in a different way from the paragraphs, with no extra line breaks following it, but with each cell inside a row separated from the next by a tab and the rows themselves separated by a single line break. On the plus side, neither browser produced extra line breaks.

If you have a number of different browsers installed, you can also use them to download the same page in text format and compare the versions they produce.

Exercise 11

By now, you’ll hopefully be able to download/save files to your computer quite easily, so the only difficulty in this exercise could be in identifying what some of the HTML codes mean in terms of creating a certain layout or formatting. What you’ll hopefully have observed already is that headings in HTML start with <h, followed by a number. In our document, we only have one heading, and the number here is 1, which means that it’s heading level 1, that is, a main heading, similar to a chapter heading in a book. You’ll probably also have noticed a number of tags that start with <p, followed by additional *class* information, but once only as <p>. What all instances of these have in common is that they contain some text, and that this text had </p> at the end, to close the tag. The <p>...</p> tag is perhaps the most common tag in HTML because it encloses paragraphs, which account for most of the textual divisions inside web pages.

Furthermore, you’ve hopefully observed that the table is contained inside a <table>...</table> tag, with table rows (<tr>...</tr>) inside them, which, in turn, contain cells marked as <td>...</td> for table data or, in the header row, as <th>...</th>. Some other things you could have noticed are that materials that are highlighted, i.e. emphasised, appear in ... tags, and that

hyperlinks to external pages (or my email address) are enclosed in `<a>...` tags that contain a so-called href *attribute* which specifies their location/target.

Exercise 12

This exercise should be pretty straightforward, provided that you have permission to install the program on the computer and follow the instructions as described. The only problem could be that the files you choose to get are not downloadable via a program. Some web servers can detect such things and treat automated downloads as an intrusion. In this case, unfortunately, you don't have any other choice but to download and process the files manually.

The main purpose of the exercise was actually to demonstrate to you that there are ways of downloading web pages automatically, as well as store and process them in a principled manner, so as to be able to conduct different types of analyses on them later.

Exercise 13

The main point of this exercise was to teach you how you can narrow down a web search to be able to find only particular types of files, rather than potentially having to click through many results pages just to find a few of these documents among a multitude of other – mainly HTML – documents. A by-product of this exercise is, of course, that you should now also have a number of Word and PDF documents that you can extract some text from later.

Exercise 14

This exercise should have made you aware of how (deceptively) easy it may be to convert some copied data into a more appropriate form. The search-and-replace functionality built into many editors or word-processors these days is a very powerful tool to make changes comprehensively and quickly, but, as you've hopefully gathered from my comments, there may well be some risks involved if you're not sure about the exact nature of the data and therefore unable to anticipate any potential issues. Thus, the semi-automated way to search and replace, where you first verify whether you want to replace what's been found, may often be a safer option. If you're too pressed for time, or simply impatient to use this method, then you should at the very least test the results of your clean-up operations on a sufficiently large amount of data.

Exercise 15

Again, the basic operations for extracting the text should be relatively straightforward, especially for Word documents, provided that you either have Word itself or a reasonably good word-processor that can handle Word documents, such as OpenOffice Writer, installed. Of course, more complex constructs, such as tables

or lists, will never quite look the same in plain-text format, but the essential thing is that we can somehow get a handle on the text content. PDFs, on the other hand, may present more of a problem as, due to their graphical nature, it's unavoidable that we'll end up with unwanted line breaks within paragraphs which probably need to be replaced by spaces manually as and when appropriate to avoid complications when trying to create frequency lists or otherwise processing the text documents later.

Exercise 16

In doing this exercise, you should not only have learnt how to remove unwanted parts of a document, but also gained more experience in working with editors. Ideally, you'll also have memorised the keyboard shortcuts by now because if you have to process a lot of data, the more efficient you are in moving around a document, the more time you'll save that you can later use for actually analysing and interpreting the data, which is also a very time-consuming process, although it may sometimes seem as if the data preparation is the most extensive part of a project.

Exercise 17

As before, this exercise should be relatively straightforward if you follow the instructions carefully and are working on a Windows system. As pointed out in Section 4.4.5, compressing files in this way is not only useful for archiving purposes, but especially if you want to make your data available to other people online or via email because an archive constitutes a convenient and relatively small-sized package (depending on how much data you have in it) that can easily be copied or downloaded and later extracted again, so it's definitely worthwhile practising how to do this.

Sources and Further Reading

- Edwards, Jane & Lampert, Martin. (Eds.). (1993). *Talking Data: Transcription and Coding in Discourse Research*. Hillsdale, New Jersey: Lawrence Erlbaum Associates.
- Jenks, Christopher. (2011). *Transcribing Talk and Interaction*. Amsterdam: John Benjamins.
- Leech, Geoffrey, Weisser, Martin, Wilson, Andrew, & Grice, Martine. (2000). Survey and Guidelines for the Representation and Annotation of Dialogue. In Gibbon, Mertins, & Moore. (Eds.). (2000). *Handbook of Multimodal and Spoken Language Systems*. Dordrecht: Kluwer Academic Publishers.

5

Concordancing

5.1 What's Concordancing?

Concordancing is an analysis technique that allows linguists to investigate the occurrences and behaviour of different word forms in real-life contexts, that is, in situations where they have actually been used by native or non-native speakers. This is quite different from more 'traditional approaches' in linguistics that simply rely on the intuition of native speakers in order to determine 'correct' usage, and especially provides a mechanism for non-native speaker researchers to justify their research claims, or learners to improve their awareness of many different features of language by investigating different word forms in their 'natural' contexts, either with or without the guidance of a teacher. Learners can thus achieve a more realistic learning experience that is at least a little closer to language acquisition, rather than simply learning specific structures and rules. On a more 'commercial' level, concordancing is heavily employed in lexicography in order to select the most frequent, suitable, and representative examples for a particular lexicon entry, as well as to help disambiguate between its different senses. Based on concordances and frequency lists, modern dictionaries often now contain information regarding the frequencies of words in different domains, their typical collocations, register appropriateness, etc. In textbook preparation, concordancing can also help to identify the most important and salient features of vocabulary items and idiomatic structures, as you'll hopefully soon realise through the exercises in this chapter.

Essentially, a concordance is a listing of individual word forms in a given specific context, where the exact nature of the context depends on the requirements of the analysis and which particular program one may be using. The word context here refers to something different from what we discussed above, that is, not the situational usage in a particular place and time, but instead the immediately surrounding text, something we can also refer to as *co-text* in case of ambiguity.

Exercise 18

As a preliminary awareness-raising exercise, go to the online resource pages and open the page for ‘Concordancing’, which contains a short sample paragraph, constructed for illustrative purposes.

Try out the built-in concordancing facility by searching for the words ‘this’, ‘very’, and ‘in’ in the paragraph, and observe what happens. Do the results always correspond to what you would expect to happen?

Exercise 18 will hopefully already have alerted you to the fact that, in analysing language, especially by computer, we may sometimes get rather unexpected results. This is one of the ‘beauties’ (albeit also one of the pitfalls) of doing corpus linguistic analysis because it allows us to identify language features that we may never have expected to find, thus providing inspiration for further and deeper research into the regularities and irregularities of language (structure), which generally go hand-in-hand. This is a phenomenon we’ll frequently encounter in our analyses from now on, and it requires us to always have an open mind and the willingness to let the data ‘drive’ our interpretation, rather than trying to force the data to fit the theory, which is something I’ve frequently observed, for example, when colleagues who were doing literary analysis were not working closely enough to the actual text, but always somehow tried very hard to make the text fit the theory they were using.

The context (or co-text), for a concordance, as in traditional, printed concordances, may be a whole sentence, a paragraph, or simply a given number of characters to the left and/or right of the search term. The latter form is the one most frequently encountered in the results produced by modern concordance programs (concordancers), and is known as the *keyword-in-context* (KWIC) format (see Figure 5.1).

In this format, the *search term* – which may be either a single word, different forms of a word or even a series of words in a row – is usually displayed in the centre of the display window and may additionally be highlighted in a different colour or format. The results of these searches can usually also be saved to disk, together with additional information, such as the *line number* or *file name* where the occurrence was found.

```

KWIC
repeated party opposition to the internal market in the National Health Service and said there had b
ng to prospective Labour parliamentary candidates in London, Mr Cook said his party <quote>will bring
have opted out</quote>. "If there is an election in November and we win office we will stop any hosp
nber and we win office we will stop any hospital in the pipeline." </p> <p> He and his colleagues are
ted States as an example, he argued that markets in health care are flawed because they stimulate dem
rs give regime a shock </p> <p> From Kevin Hamlin in Singapore </p></head> <p> SINGAPORE's ruling Peop
le's Action Party (PAP) suffered stunning losses in Saturday's general election, opening a new polit
eneral election, opening a new political chapter in the island republic and raising questions over G
won a landslide victory, having secured 77 seats in the 81-seat parliament, Mr Goh was visibly shake
of vote dipped to 61 per cent from 63.2 per cent in 1988. The Singapore Democratic Party (SDP), whic
l now be that of a conventional governing party (in al partisan situation. This is a new situation.
san situation. This is a new situation. Politics in Singapore cannot go on as before. Certain things
ote>a little deaf</quote> to the needs of people in opposition seats. He also accused opposition can
ion seats. He also accused opposition candidates in one constituency of using racial politics to win
eeply concerned for the future shape of politics in multi-racial Singapore</quote>. Malays account fo
show</quote>, he said, adding that Lee junior was in charge of campaign strategy. Mr Goh acknowledged
re the victors and that yesterday was a landmark in Singapore's political development. <quote>The PA
nstructive attitude towards industrial relations in Britain which is reciprocated by many managers. I

```

Figure 5.1 Example of a KWIC concordance output

Although all concordancers usually provide a minimum of core functionality, such as the ability to produce KWIC concordances, they also often offer many additional options, so that it's difficult to generalise about which types of functionality to expect from any particular program.

Most concordancers are also *stream-based*, which means that they 'suppress' line breaks by replacing them with spaces, so that the text is essentially read as a continuous stream of words. Thus, words that actually occur on different lines in the text may still be presented as part of the context. In contrast, a *line-based* concordancer, such as the one built into my Simple Corpus Tool (downloadable from martinweisser.org), will only extract and display the immediate context found on the same line, plus a number of surrounding lines specified.

5.2 Concordancing with AntConc

AntConc is a free and highly versatile concordancer that provides support for many advanced concordancing features, including support for non-Latin character sets, as well as additional functionality that we'll discuss in the relevant later sections. The program itself, which is available in versions for Windows, Mac, and Linux, can be downloaded from http://www.laurenceanthony.net/antconc_index.html. On Windows, it doesn't even require installation, so you can simply save it to a folder on your computer or memory stick (I suggest you call it 'AntConc') or anywhere on your computer that you have write-access to, and then start it by (double-)clicking the executable. On the Mac, you'll first need to extract the .app file and drag it into the Applications folder to install it. On Linux, you might be able to run the executable file directly once you've extracted and set the permissions to make it executable, but if you're running a 64bit Linux, you probably first need to install 32bit support libraries. This may vary from version to version,

and is therefore too complex to discuss here. In case you encounter this problem, you should either consult an expert or search through forums for your particular version of Linux to identify ways of enabling 32bit support.

Exercise 19

As our first exercise, download and start AntConc now, and familiarise yourself with the elements of the program window a little.

Also have a quick browse through the menus to get a first impression of the options, even though you may not understand what they all mean yet.

When you first start up the program, you'll be presented with an initial screen like the one shown below:

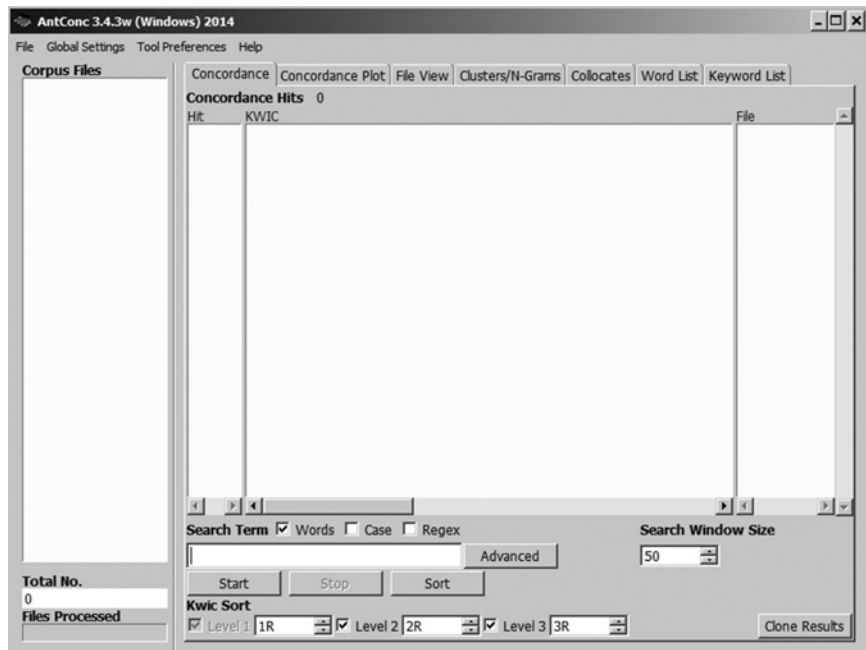


Figure 5.2 The AntConc startup screen

You can see that the tab for ‘Concordance’ has automatically been activated. The first thing we need to do now is to select either a single file or a number of different files to concordance on. As you might have expected, we do this by choosing the appropriate option from the ‘File’ menu. Here, AntConc provides us with two different options, one for selecting one or more individual files, or for choosing a whole directory at once, as depicted in the Figure 5.3.

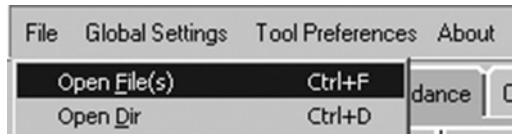


Figure 5.3 AntConc file opening options

If you find yourself regularly using AntConc, you might also want to learn the appropriate shortcut keys for opening files ('Ctrl + f') or directories ('Ctrl + d'), which are listed next to the menu entries. For using a whole directory, there are further options regarding specific file types and file information that you can choose from the 'Global Settings' menu/dialogue as shown in Figure 5.4.

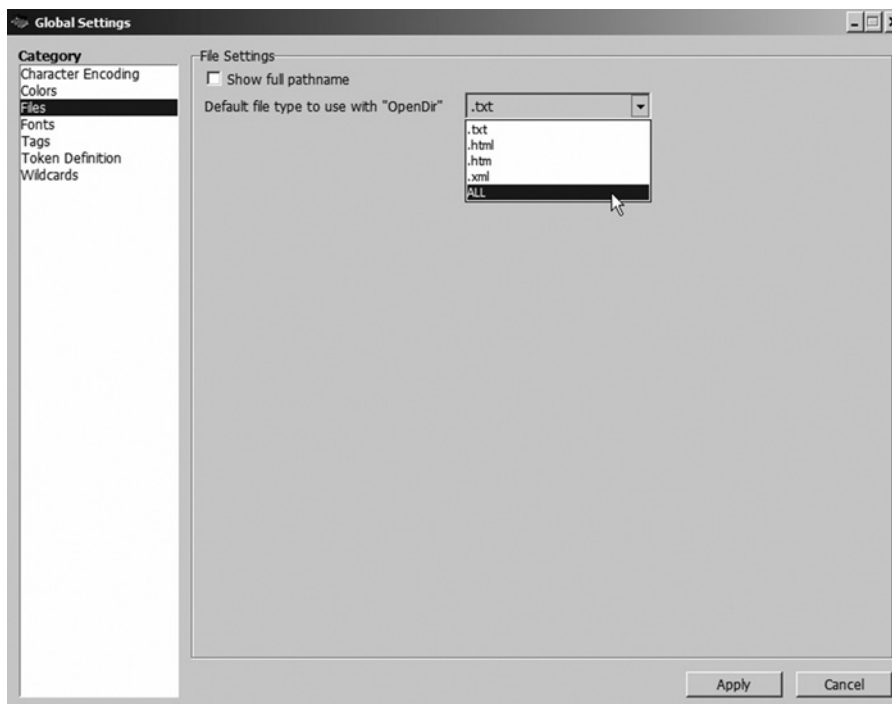


Figure 5.4 AntConc file settings

When you work with directories, you should probably set the file type option to 'ALL', as many of the files you may be analysing might contain additional markup and have a different extension, and therefore not automatically be made available by AntConc otherwise. If you then want to have this configuration option available permanently, you can overwrite AntConc's default configuration file by choosing 'File→Export Settings To File...' and simply replacing the file 'antconc_settings.ant'. To create additional configuration options – which also store information about the files you had open and your last searches, etc. – later, you can create separate configuration files and then load them via 'File→Import Settings from File...' as and when required for a particular research purpose.

Don't worry if you open a directory and there may be files listed that you don't really want to include in your analysis – you can always remove them from the analysis corpus later. For now, however, we're going to start by selecting only two files for concordancing as part of Exercise 20, which is designed both to give you some initial practice on running concordances, and to illustrate one of the most important linguistic issues you'll encounter in concordancing, that of *polysemy*, that is, the ability for word forms to either represent multiple meanings or belong to different word classes.

Exercise 20

Search for the folder in which you've saved the Jane Austen texts *Emma* and *Sense and Sensibility*.

Open the two files in AntConc. The result should look similar to this.

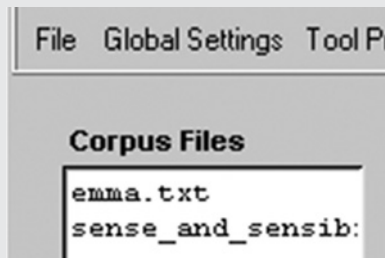


Figure 5.5 AntConc 'Corpus Files' window (two files loaded)

In the search box, type in the word, *round* as indicated in the following graphic.

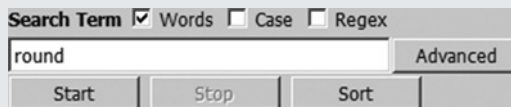


Figure 5.6 AntConc 'Search Term' and search options

Note that the option to search for 'Words' is automatically pre-selected by default and that 'Case' is not selected, which means that the search will be *case-insensitive*. In other words, no distinction is made between capital and non-capital letters, so that you could actually have typed in *Round* or even *rOuNd*, too, even if the latter doesn't really make much sense. We'll discuss the third option, 'Regex', in more detail in Chapter 6.

Think about what kind of results you would expect to get from the search (in terms of word classes, etc.), and then click on **Start**. You should get a result that looks like Figure 5.7.

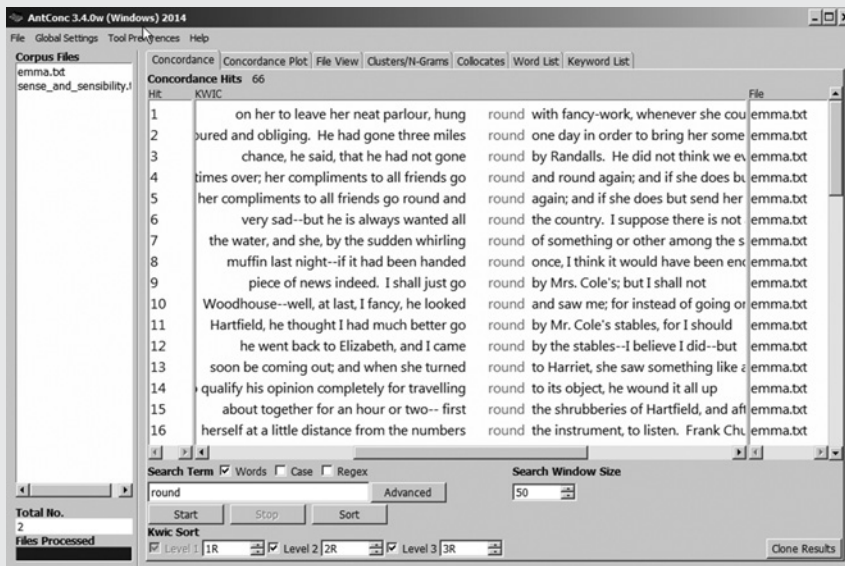


Figure 5.7 AntConc results for *round* in two novels by Jane Austen

In the window on the left, you see the number of the occurrence ('Hit') found, in the middle the results in KWIC format, with the search term highlighted in blue on your screen, and on the right, the file name where the hit was found. Above the KWIC window, you can see the overall number of hits displayed. You can adjust the size of all the (sub-)windows to suit your needs. If you need more or less context for your KWIC display, then you can also adjust the options by the 'Search Window Size' as shown here.



Figure 5.8 AntConc 'Search Window Size' options

If you click on the search term, you'll be taken to the display in the file 'File View' window where you can see the current hit in its complete context in the file where it was found.

Click anywhere in the KWIC window, apart from on the search term, and scroll down the list of occurrences to check the results and see whether they match your expectations or whether there's anything unusual. Think about the potential of word forms for representing different meanings or word classes (polysemy) and usage!

5.2.1 Sorting results

If you think about how long it would have taken you to find all these occurrences manually in both texts, you can see how useful it is to be able to create concordances like this within a few seconds. However, as useful as an ordinary KWIC concordance may be, AntConc also offers us the functionality to create much better views of our search results by providing options for sorting the results based on their immediate left or right contexts. We'll have a quick look at how this is done, and then you'll hopefully understand the usefulness of this option without much further explanation, although, as usual, some will be provided in the solutions section.

Exercise 21

Take a look at the sort options immediately to the right of the search options.

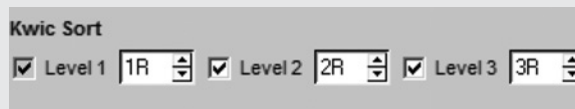


Figure 5.9 AntConc 'Kwic Sort' options

You'll see that for 'Level 1', which is already pre-selected, there's '1R' in the box by default, for 'Level 2', '2R', and for 'Level 3', '3R'. This means that, once you click on **[Sort]**, the primary sort will be conducted alphabetically on the word immediately to the right of the search term, and in descending order of frequency. However, the sorting is then not yet complete, as it will be continued, based on the next word to the right, and finally, also on the 3rd word to the right.

For our current purposes, we want to sort according to only the word immediately to the left of the search term, so change Level 1 to '1L' and untick the check boxes next to the other two levels.

Before you try this, first change the search term to *mind*, then click on **[Start]** and finally on **[Sort]**.

Check the results and try to understand why this feature may be so useful...

Tip: If you have problems in getting upper- and lowercase characters sorted separately, open the 'Tool Preferences' for 'Concordance' and check the option for 'Treat case in sort'.

5.2.2 Saving, pruning and reusing your results

5.2.2.1 Saving There are essentially two different types of ‘Save’ option in AntConc, one that allows you to save your results as a regular plain-text file, and one that provides the option to only save the results displayed inside the analysis window temporarily, so that you can compare them to the results of another search side-by-side. The first type is accessible through the ‘File→Save Output to Text File...’ menu option, which saves the hit number, the concordance line, plus the file name for each hit. Saving our results not only provides a means of keeping a record of what we’ve found, but also to analyse and/or manipulate the data further, perhaps also using different programs, so let’s see how we can produce such output in Exercise 22.

Exercise 22

Save the results of the previous search to a text file. I’d suggest you save all such files to a folder called ‘results’ inside your ‘AntConc’ folder, and always give each file a suitably descriptive file name that will later allow you to identify what its contents are. In this particular case, maybe something like ‘mind_in_jane_austen.txt’ would be appropriate. Once you’ve saved the file, open it in your editor to view the output.

Unfortunately the search term is not highlighted in a very useful form in the output, although you can achieve some degree of ‘highlighting’ by changing the delimiters between the different fields of the output. In order to get around this slight limitation of AntConc, you can of course always use your favourite plain-text editor and do a search-and-replace operation where you replace the search term by something like >>> [search term] <<< (leaving out the square brackets). Let’s try this:

Exercise 23

Open the file in your editor and run the search-and-replace operation to add the highlighting.

The second ‘save’ option can be triggered by clicking the **Clone Results** button on the bottom right-hand side of the AntConc window and will open a separate window containing the results of the current analysis, so that you can then run the other analysis and compare it to the original one side-by-side. The search terms in the saved window will still be displayed in colour, but are no longer clickable,

so direct access to the larger file context, as it's available in the main window, is unfortunately no longer possible.

5.2.2.2 Pruning Sometimes, not all the results of a search may be relevant to the problem or topic you're investigating, or you may simply end up with too many examples of a similar nature, so that you need to make some decisions as to which ones of the results you want to keep or treat as relevant to your purpose. If necessary, you should also document the decisions, maybe if you need to explain your choices to other people. For instance, in the grammatically polysemous results for *mind* in Exercise 21, you might want to only retain the noun usages for your research because you're only interested in nouns, but to discard all verbs, or vice versa.

In such cases, you have two choices for removing unwanted hits. The first one is to save the results to a file, as described above, and then delete all irrelevant material from this file. However, AntConc also allows you to remove some hits before you save the results. You can best do this by holding down the 'Ctrl' key, clicking on the hit number, and then pressing the Delete key. If you want to delete more than one non-consecutive hit, then simply keep the 'Ctrl' key pressed down and select any other hits until you've highlighted all the ones you want to remove, and then press Delete. If you're lucky enough to have all the data you want to delete in consecutive order, then you can also use 'Ctrl' + click on the first one, and then, holding down the 'Shift' key, click on the final one. This technique for making multi-selections is something you should already be familiar with from our discussion in Section 4.4.5 on how to select files for archiving in a file manager.

In case you're now worried that this may destroy your corpus data, there's no need, because whatever AntConc displays as a list of results in its KWIC window is in fact copied from the original file, rather than showing the original data itself.

Exercise 24

Try pruning a few hits in the current result set, perhaps deleting all non-noun occurrences. Don't worry, you'll always be able to re-run the concordance to get the original results back 😊

Save the results of the search that you've just pruned under a suitable file name.

5.2.2.3 Reusing Once you've extracted some data from a corpus, or a number of different corpora, there are different options regarding what you can do with this material. In some cases, you may already have identified all the relevant examples, and there's no need for any further analysis if your examples already illustrate all the points you might want to make in your 'research output', whether it's an assignment, an academic paper, a dissertation or thesis, or some teaching materials. This, however, is only the simplest case, and rarely do all those extracted

examples simply speak for themselves, but instead may still require a substantial amount of analysis before they really tell the whole story.

Although you can always look through your data and keep notes on any salient features you've identified, as well as perhaps periodically go back to revise these notes if you encounter some new relevant details that may confirm or contradict your earlier observations, this may not be the best strategy for understanding your data completely. Perhaps a better way to analyse some data is not just to work through it, but, once you have all the relevant results in a file, to edit this file and mark up your samples using some simple codes in order to categorise them. This type of basic initial coding needn't be very complex, but can consist of a few easily identifiable markers that you insert somewhere in a text. The key to this is in the two words "easily identifiable" because whatever codes you may insert in your file, those shouldn't easily be confused/confusable with any regular text. Therefore, maybe simply inserting your codes in round brackets may look too much like regular text to be really useful, and even replacing the round ones by other types of brackets may be problematic, for instance, if you happen to be analysing a linguistics text where different types of brackets are used to indicate different levels of analysis, such as, for example, curly brackets to enclose morphemes, or square ones to mark phonetic representation. Thus, you need to find something that distinguishes your code from any ordinary text, ideally by using characters that do not form a part of any normal text. For example, if you're investigating grammatically polysemous words like the ones we searched for above, you can add an underscore and a simple word category identifier – such as N for 'noun', V for 'verb', etc. (we'll soon learn more about this when we discuss morpho-syntactic annotation in Section 7.1) – to each occurrence of the word (form). Once you're finished, you can just load the resulting file in AntConc instead of the original corpus, and filter out or sort the relevant entries in another concordance. And once you've reached a more advanced level in your research, you might also want to consider using a 'proper' markup language, such as HTML or XML, for annotating and categorising your data. Such options will be explored in some more detail in Chapter 11. For now, we only want to practise adding simple word class codes to our results file.

Exercise 25

Open the results file from Exercise 22, and categorise at least the first 30–40 search results, but ideally all, according to their grammatical categories as described above.

Save the file and load it in AntConc.

Run a search for the search term again, this time only searching for those occurrences where the word *mind* occurs as a noun, then as a verb.

Solutions to/Comments on the Exercises

Exercise 18

As you'll hopefully observe while doing the online exercise (and can verify again in the exercise paragraph replicated below), the computer search for our fairly common sample words managed to find *this*, but not *This*, *in* as a word, but also many other occurrences of the character (letter) sequence *it+n* (where both are lowercase characters) that you would probably not have expected to find. This is because the computer does not understand what a word is/may be, and can thus only 'blindly' match characters. The only 'word' that was apparently not an issue in this exercise is *very*, but I said "apparently" above because of course the same issue as for *this* also applies to *very*, only that you may not have noticed it because no form of *very* with an initial capital letter occurs in the text.

This is a short test paragraph to illustrate *very* basic concordanc*ing*. As you can see, *in this* basic form, concordanc*ing* is *very* similar to a Google search, only that it shows you the results *in* one or more pre-selected pieces of text, rather than try*ing* to find them *online*. The other th*ing* is that our expectations concern*ing* the results may differ from the actual output of the concordancer, as you will hopefully have noticed while concordanc*ing* on the words *this* and *in*. For the former, you might have expected to find both occurrences of the word with and without an *initial* capital letter, but the *very* first word of the paragraph did not *in* fact get highlighted because *in* general concordances tend to be *case-sensitive*, so that they will only find exact matches. This unfortunately resulted *in* you find*ing* fewer occurrences than you would have expected, whereas the search for *in* gave you many more *hits* (i.e. search results) than you would probably have expected because the grapheme comb*ination* <*in*> occurs as part of a number of other words *in this* paragraph. Both of these issues are related to differences *in* the way that human readers and the computer process texts and we'll explore options for handl*ing* them later.

Exercise 19

This exercise was designed to allow you to explore (very roughly) the different types of functionality a concordance program, such as AntConc may provide in addition to pure concordancing. Furthermore, if you've paid close attention to the configuration options, you may already have noticed that the program not only allows you to work with single, or a number of different, files at the same time, but that you also have some degree of control over the particular (plain text) input format and its encodings, something you should by now be able to understand better through the exercises we did in previous sections. As far as the basic concordancing interface is concerned, you'll hopefully already have spotted

that you can in fact adjust the context displayed by the concordancer for showing the result, as well as that there are various options for sorting our results, which is something we'll explore in more detail in Section 5.2.1.

Exercise 20

Essentially, the main aim of this exercise was – apart from giving you some initial concordancing practice in AntConc – to further raise your awareness of the fact that words – or rather word forms – in text are basically just sequences of characters for the computer/concordancer. Thus, generally the concordancer will not be able to ‘understand’ that you may be looking for a word with a particular meaning if there are homographs or polysemous forms, as in the case of *round*, which may be classified as an adjective, an adverb, a noun, a verb, or preposition. And, although you would most likely have expected to mainly get hits for the adjectival meaning, which is perhaps the most prototypical meaning associated with it these days, you will have seen from the results from the two novels by Jane Austen that the adverb or preposition meaning appear to have been much more common in her day, or perhaps only in her writing, something which only an in-depth comparison with the works of other authors of her period can establish for sure.

Another thing you could potentially have learnt from this exercise is that, because we tend to have particular expectations regarding any data we may be analysing, these expectations may sometimes initially make us blind to alternative options. However, as the examples from the exercise will hopefully have shown you, if we use a concordancer to investigate enough data, we're bound to find examples that may run counter to our imagination, and also help us deepen our knowledge/understanding of how words are used, which is one of the reasons why concordances are not only useful for research in lexicology or lexicography, but also as a means for native and non-native speakers to develop their language skills.

Now, let's briefly return to the issue of case-sensitive vs. case-insensitive searches. The computer stores what we commonly refer to as small and capital letters in a different way, so that it becomes possible to search for them separately, for instance if we only want to find words that occur at the beginning of a syntactic unit ('sentence') or proper names. In many cases, however, we just want to find all instances of the word to get an overview of its different functions, and, for this, the position may simply not always be relevant. To test and see what happens if you make the search *case-sensitive*, go back to your search in AntConc, briefly change the search term to `Round` with an initial capital letter, and run the search again. This time, you should in fact get no hits at all because none of the occurrences of the word *round* are in such an initial position in our texts.

Exercise 21

If you've observed the results of our search for the word form *mind* closely, you'll probably have noticed a number of things. First of all, that – just as in our previous

exercise – the word form *mind* is again grammatically polysemous, i.e. may represent different word classes, in this case either the singular form of the noun or the base/infinitive form of the verb. Secondly, that sorting the output in this way makes it far easier to see which word forms may precede the hit most frequently, and last, but not least, also which word classes/parts of speech may occur most frequently/typically with a given word form. For example, by examining the first 121 hits, you'll notice that more than half of all 225 hits are in fact nouns (apart from number 29, *don't mind*) because they're preceded by determiners, possessive pronouns, genitive nouns (e.g. *Emma's*), qualifying attributive adjectives, and quantifiers, while the rest of the examples do contain a few more instances of verbs, characterised by preceding adverbs, such as *never*, or the negation operator *not*.

However, as examples 212 – “the Highbury people, but if you call to mind exactly the present line of the path.” – & 213 – “bake or boil. William did not seem to mind it himself, he was so pleased to think” – illustrate clearly, disambiguating the word class simply based on the preceding word may not be straightforward, either, as the first of them has the word form *to* occurring as a position marker, and the second as an infinitive marker. Such issues may also cause problems for approaches to the automated processing of language where such disambiguation is of course also important, but naïve algorithms based on probability-based assumptions regarding the word class of only a single word preceding a grammatically polysemous item would potentially fail, as such probabilities would, in our case, clearly favour the more frequent use of *to* as an infinitive marker.

Exercise 22

As this is more or less a ‘mechanical’ task, there isn’t really much you can do wrong here, apart from maybe making one or two minor mistakes. The first could be that of not choosing an appropriate output folder to store your results in, which could mean that you may end up spending a considerable time searching for your output files later. The second might be that you don’t label your output file sensibly, which will have a similar effect in that you may find yourself searching for the right file for longer than necessary if you have a number of different output files that are not clearly distinguished from one another. Both of these mistakes basically may cause you to lose valuable time that could be spent on actually *analysing* your data, thus ‘throwing away’ one of the most important advantages of using corpus linguistics as a methodology, which is that it allows you to save a significant amount of time finding a large amount of potentially relevant and interesting data quickly. In the past, having long file names wasn’t even possible, but these days, having a maximally explicit file name that is up to maybe 20 characters long is no longer an issue, although I have occasionally experienced some issues with exceedingly long folder or file names when trying to back up files on even more recent versions of Windows.

Exercise 23

This exercise is very similar to the ones we did in Chapter 4 in order to clean up our data, so it shouldn't really be too difficult to do. The only thing you may need to be careful with is not to accidentally highlight hits that you don't want because the sequence of characters that represent your hit might also be part of another word. In order to avoid this problem, it may be best not to do all replacements fully automatically, but instead use the search-and-replace functionality step-by-step to identify and 'OK' each replacement.

Exercise 24

Again, this exercise is quite straightforward. The only thing that could happen is that you accidentally either delete an entry you hadn't intended to delete, in which case you'll need to re-run the concordance and delete more carefully, or that you may accidentally select too many hits before pressing Delete. In the latter case, you won't need to re-run the concordance, but can simply click anywhere in the hits to remove all selections, although you'll still need to select the ones you want again. In order to avoid any issues with this, it's probably best to only select a few hits each time before deleting, and then start selecting again...

Exercise 25

This exercise was really just meant to demonstrate a) how easy it is to categorise your own data in a simple way, and b) how efficiently you can then work with such pre-categorised data. Of course, adding this type of information can be quite time-consuming, so, in later sections, we'll explore ways of adding similar types of information automatically, as well as looking into ways in which we can exploit such annotated data even more extensively in order to search for and identify more complex linguistic patterns.

Sources and Further Reading

- Hunston, Susan. (2008). How Can a Corpus be Used to Explore Patterns? In McCarthy, M. & O'Keefe, A. (Eds.) (2010). *The Routledge Handbook of Corpus Linguistics*. London: Routledge.
- Sinclair, John. (1991). *Corpus, Concordance, Collocation*. Oxford: Oxford University Press.
- Sinclair, John. (2003). *Reading Concordances*. Harlow: Pearson Education Ltd.
- Tribble, Christopher. (2008). What Are Concordances and How Are They Used? In McCarthy, M. & O'Keefe, A. (Eds.) (2010). *The Routledge Handbook of Corpus Linguistics*. London: Routledge.

6

Regular Expressions

Imagine yourself frequently having to look for very similar patterns that differ only in a few minor details, such as in the different forms of a *verb paradigm*, for example, *(to) walk, walks, walking, walked, singular* and *plural* forms of nouns, or words that may start with either a capital or small letter. This is a very common and important task in linguistics, even for languages that only exhibit a moderate degree of inflection, such as English, but much more so for more strongly inflected ones like German, etc. Now, even if you are aware of all the relevant forms you may need to identify, and search for each of these forms separately in a row in a concordance program, you can only save the results, maybe even print them out, and then compare them afterwards. However, this makes your job much, much more difficult and time-consuming than it needs be, and may also lead to your overlooking some details or missing out on important generalisations. Thus, to be able to work more efficiently with a concordancer or similar search tool, it would be very useful to be able to specify more complex patterns that we could then look for all at once, and also make use of the other useful options we've explored before, such as sorting, etc., to help us simplify our analysis procedures.

Regular Expressions (or *regexes*, for short) are an important and very powerful means of specifying such complex search terms for concordances or computer programs for language processing. Most concordance packages support at least some basic forms of regexes, although they're not necessarily as advanced as the options offered by command-line search utilities, such as (e)grep (*global regular*

expression printer), or programming languages, such as *Perl*, *Python*, or *Java*. Here, we'll discuss the most important general regex concepts in order to demonstrate their usefulness for linguistic purposes, but may introduce further, more complex, options later, as and when required for advanced analyses. We'll start by discussing some of the abstract notions behind regexes, each time accompanied by appropriate exercises that should allow you to observe and test the individual features we're discussing. At this point, perhaps a cautionary note is in order, as the individual features related to regexes that we need to introduce individually will really only make sense fully from a linguistic analysis perspective once we can put a number of them together in order to construct efficient searches. Therefore, the very first exercises, even though I've always tried my best to make them as relevant as possible to linguistics, may not yet appear very useful to you, and you might well be tempted to give up if things appear too abstract. However, if you persist, I can promise you that soon you'll not only be able to understand the value of regexes for achieving even highly complex tasks in linguistic analysis very efficiently, but will also learn to hone your analysis skills related to morphology and morpho-syntax, something you'll definitely be able to profit from in your work in corpus linguistics, as these two areas often form the basis for further linguistic analyses.

Tip: When using regexes in any other program, always bear in mind that there may be fairly large differences in their implementation, depending on the purposes or design decisions made by the author! This is especially the case for using them in search-and-replace operations, where many editors will only offer a reduced set of functionality, or even implement a special syntax for some purposes.

Below, you can see a short test passage. This passage will mainly be used in the online exercises (http://martinweisser.org/pract_cl/regExes.html) for displaying the results of all the basic exercises that follow in this chapter because it's easiest to understand the results of our searches when they get highlighted. However, you should also try and train yourself to identify these patterns within the printed text here, so that you'll learn more efficiently how to use and specify regexes, which is why I'd suggest that before you look at the results highlighted in the online version, you first scan the sample paragraph below visually to see whether you can form any hypotheses as to what the results would be, and also why a particular feature may be useful for linguistic analysis. Please note that, in a real-life scenario, that is, other than for illustrative purposes, we may never search for all of the features illustrated below individually, but usually employ them to expand or constrain a certain search term in a manageable fashion, that is, only to focus in on a linguistic pattern. Otherwise, we may end up with too many, and potentially also meaningless results, which is why I'm deliberately restricting the size of the input text to a short paragraph here. Another feature of the online exercise is that all occurrences of a regex feature get highlighted at the same time, while a normal concordancer would find and list them individually.

This is a short test paragraph. It will allow us to explore and test different regular expression features and concepts, such as character classes, quantification, and grouping/alternation, by displaying them in a separate colour. For good measure, and to be ‘well-rounded’, this also contains some numbers, punctuation, and special characters here (1, 2, 3, 10, ! ;), as well as some unusual words like ‘theme’, ‘rtheme’, ‘phatic’, ‘thistle’, ‘chisel’, or ‘phishing’, and a bit of fəˈnætɪk.tɪənˈskɪpʃn̩. Can you understand all the results, based on your own intuitions concerning different types of characters and how they make up words?

Figure 6.1 Sample paragraph for practising and understanding regex patterns

6.1 Character Classes

Character classes allow you to specify groups/ranges of characters in order to efficiently search for such characters that have specific shared properties, maybe all being *vowel* or *consonant* letters (as opposed to the actual vowel or consonant sounds, i.e. phonemes), etc. These classes contain individual characters or *ranges* of characters, where a range is indicated by a hyphen in-between characters that should be consecutive. These patterns are enclosed in the same kind of square brackets ([]) that we usually use for (narrow) phonetic transcription. Here are some basic examples:

- [a-z] (all English lowercase letters); useful, for example, if you want to exclude all *proper names/nouns* or *sentence/paragraph-initial* words from a search
- [A-Z] (all English uppercase letters); useful for finding proper nouns or initial words
- [0-9] (all digits, also often abbreviated \d); useful for finding and possibly removing numbers from word lists
- [aeiouy] (all lowercase vowel letters for English); useful, for example, for finding words that begin or end in vowel letters
- [Tt] (either <T> or <t>); this will allow us to specify words that start with an upper- or lowercase letter ‘t’, as in *This/this, That/that, The/the*, etc.
- [A-E0-3] (all uppercase letters between A and E, all digits between 0 and 3, and a space); potentially useful for finding specific English postcodes

In order to develop a feeling for how to construct character classes and establish their usefulness for identifying basic linguistic features potentially shared by different word forms, let’s do a two-part exercise, both off- and online.

Exercise 26

Test the character classes shown above on the sample paragraph provided in Figure 6.1 by first visually scanning the text for members of these classes to see whether you could easily pick out what each class would allow you to find.

Next, search for the same character classes in the online exercise version at http://martinweisser.org/pract_cl/regExes.html and verify whether your visual scanning did indeed allow you to identify the same results.

Feel free to make your own changes to the classes, too, to explore some more.

Other classes that are often predefined as shorthands/abbreviations are:

- `\w` for all *word* characters, usually including hyphens, theoretically also matching characters that are not part of the character set used for English, such as, for example, β , \ddot{a} , \acute{e} , ζ , \tilde{n} , ξ , etc.
- `\W` for all *non-word* characters, such as, for example, punctuation
- `\s` for (white)space characters (but sometimes a *whitespace* itself only)
- a single `.` usually stands for *any* arbitrary character, unless it occurs inside a character class, in which case it simply means a dot.

Exercise 27

To experiment a little with the character class shorthands shown above, test these on the sample paragraph above and online, and observe the effects.

Each character class, the way we're defining and using them at the moment, essentially represents options that act as placeholders *for one single character only*. The only reason why all of them will get highlighted in the sample paragraph is because the script that allows you to display them identifies them globally, that is, each and every one of them individually. Just seeing the results may also not yet allow you to see the usefulness of creating/using this type of class, but this will soon become clearer when we introduce quantification.

One special feature to note is that, as hyphens are used to indicate ranges in character classes, if we want to include them as hyphens in a class, we have to place them somewhere where they don't occur between characters, that is, either at the beginning or the end of the class definition.

6.2 Negative Character Classes

As we've already seen for `\W`, sometimes it's useful to simply be able to *negate* character classes if we want to exclude certain characters/constructs. Unless we have such a predefined group as `\W`, we need to define this class ourselves, which is done by creating the class in the same way as a positive class, only that the opening square bracket needs to be followed by a *caret* sign (^), for example, `[^A-Z]` (no uppercase English letters).

Exercise 28

Try to think of some negative character classes and test them on the sample paragraph. If you still have difficulties thinking of any sensible ones yourself, just negate the positive character classes we saw above, and try to understand what's happening.

As you'll hopefully have observed, simply excluding a particular character by negating it may have rather unexpected effects. So, for instance, if we simply exclude a capital `<T>` and expect now only to find instances of words that start with a `<t>`, just because the word form we had in mind may have been `<the>` or `<this>`, this is clearly wrong because we're only excluding one single character option, rather than the whole set of potential options we may have wanted to exclude. Thus, successfully specifying an exact pattern may often require careful thinking and using a mix of different regex options to constrain our pattern.

6.3 Quantification

Now, simply being able to look for single individual characters, or those belonging to a specific class on their own, frequently doesn't make such a lot of sense, so we need a way of being able to specify *how often* they can occur. This can be done by using quantifiers, which allow us to say whether a given character or class may (or has to) occur zero or more times, maybe even infinitely, in a row. This is a bit similar to using quantifiers in natural language, only that there we tend to quantify noun phrases through 'non-specific' (or underspecified) quantifiers like *some*, *many*, or *all*, as well as using numbers or numeral quantifiers, such as *once*, *twice*, *three times*, etc., to say exactly how many subjects or objects we may be talking about. On the other hand, in natural language we also have quantifiers that signify the absence of something, such as *none* or *no one*, which actually have pronoun character. The specific (basic) options for quantification in regexes are:

- a * following a character (class)/group means it may occur from 0 to an *unlimited* number of times; expressed in natural language, this would be from *none* to *infinitely many*.
- a ? following a character (class)/group means it may be *optional* or can occur *at most once*; in natural language: *possibly (not)* to *maximally once*.
- a + following a character (class)/group means it has to occur *at least once* but *up to an unlimited* number of times; in natural language: *minimally once* to *infinitely many*.
- a set of curly brackets {} following a character (class)/group specifies a more exact quantification
 - {5} matches exactly five times; or: *no more and no less than 5 times*.
 - {5, } matches at least 5 times or up to an unlimited number of times: *minimally 5 times to infinitely many*.
 - {5,10} matches between 5 and 10 times.;

At least in this way, we can already specify that we may want to look for something like `\s\w+\s`, that is, only words, although, of course, we need to bear in mind that not all words are actually surrounded (delimited, in technical terms) by two whitespaces. This equally allows us to take into account dialectal differences, such as the differing British and American English spellings of the word `colour`, but still doesn't quite give us the flexibility we may want in looking for specific *word forms* or *paradigms* (i.e. all associated forms of a word), which is why we need to introduce three further concepts below, after doing another exercise.

Exercise 29

Test the two quantification examples shown above using the sample text. The text on the online page is editable, so you can also delete the `u` and see whether the quantification using the question mark still works, or even try some of your own words, if you know any other differences in spelling that relate to one character only. You can also try to find words where a character is repeated multiple times.

For the example of the whitespace-bounded words, also experiment with the curly-bracket type to practise more exact quantification for the number of characters allowed inside a word. Can you already detect any practical use in this?

6.4 Anchoring, Grouping and Alternation

6.4.1 Anchoring

To be able to increase the precision in the search for words, we next need to solve our problem of word boundaries that aren't signalled by whitespace, that

is, if a word occurs at the beginning or end of a line or in a different context. In cases like these, we can *anchor* our search string in different ways. The greatest flexibility in this is provided by marking word boundaries, for example, by writing `\band\b` if we only want to look for the word *and*, where `\b` is the word boundary marker. The backslash preceding the *b* here is just like the one we've seen in the other abbreviation classes above and changes the 'meaning' of the *b* from a literal character to something different. Thus, although our example may look like the word *band* preceded by a backslash and followed by `\b`, it actually represents a combination of boundary marker + *and* + boundary marker. To see exactly how this works, let's do another exercise.

Exercise 30

First, try the example of *and* as described above on the print version of the sample text to see whether you can already spot the odd-one out, then try 'searching' with and without boundary markers in the online version.

Finally, using appropriate quantification, look for words of different length.

If we want to anchor our search term at the beginning or end of the line (or string), we can specify `^` or `$`, respectively. However, please always bear in mind that this may make sense in a program that reads texts *line by line*, such as `grep`, a Perl script, or most of my own programs that allow you to run line-based concordances, but not necessarily in a stream-based concordancer which usually reads and processes all words as a continuous stream of characters/words and may therefore ignore these markers, or may only match at the beginning or end of the whole file!

6.4.2 Grouping and alternation

Enclosing items in round brackets (in the first instance) causes them to be *grouped*. Thus, for example, `(\sapples\s)*` would look for the *exact group/sequence of characters* representing the (white-space-bounded) word *apples*, as well as specify that the word may occur zero or any number of times in a row. Granted, we generally wouldn't really expect this word to occur two or three times in sequence, unless we may have encountered a copy-and-paste error where the word was duplicated accidentally, or maybe in a question like "Why do we call apples apples?", so, in this case, we're probably more interested in specifying that the word *could*, but *need not*, occur. However, in some rare cases, apart from the question just cited, it may indeed be grammatically correct to repeat the same word form twice in a row, albeit with different grammatical functions and meanings. This could, for example, happen in a relative clause that begins with the relative pronoun *that*,

followed by the demonstrative determiner *that*, as in the following example taken from the BNC: “I realize now *that that* is what I want more than anything else” (A08 2983).

Grouping character sequences together, though, isn’t the only thing bracketing allows us to do. It also gives us a way to express *alternatives* within our group, and thus a much greater flexibility in specifying linguistic patterns. If, for instance, we wanted to be able to search for a number of different types of fruit at the same time, we could construct a regex like this: `(apples|bananas|grapes|pears)`, where the pipe symbol (`|`) separates the alternatives from one another. This way, we can also specify alternatives within alternatives because we can nest brackets within one another, as in, for example, `(\b(an?|the)\b (old|young) (wo)?man)`, which would group and find all occurrences of *alan/the old/young man/woman*, where I’ve indicated the alternatives options using slashes, allowing for either an indefinite or definite article, followed by either one of the two adjectives, and followed by either *man* or *woman*.

Exercise 31

Try something similar by specifying a regex that will find all the verbs in the sample text above, then do the same thing for all nouns. First, draft the regex on paper and then try it in the online interface. Are there any unexpected results or difficulties in specifying the patterns?

Another ‘side effect’ of using brackets in many programs/programming languages that use regexes is that whatever is enclosed in a group is captured and can usually be ‘referred to’ again via a so-called *backreference*, where each group of brackets, starting with the outermost, can be referred to by a number, generally preceded by a backslash. Thus, if we had only one pair of brackets, as in the fruit example above, we’d be able to retrieve whichever one of the alternatives was found by `\1`. In the more complex example with the nesting, `\1` would contain the whole expression found, `\2` whichever article was found, `\3` either one of the adjectives, and `\4` the initial morpheme of *woman*, provided that woman was matched in the first place, but nothing if only *man* was matched.

As concordancers generally only match one search term, though, they are very unlikely to let you do much with backreferences. However, in programming or in converting data from one particular format to another, this is extremely useful for extracting parts of complex matches, and ‘re-using’ them. In some editors that allow regex search-and-replace operations, you can even do things like automatically swap `\1` and `\2`, for instance to move adverbs after auxiliaries, as in for instance turning the phrase *I (rarely) (have) seen* automatically into *I have rarely seen*, or even turn some declaratives into interrogatives (or exclamatives), provided that you ignore punctuation and sentence case, as in, for example, *I*

have seen it to *have I seen it* (?!). If you want to, and your editor, like the ones I recommended above, supports regex replacements, you can try this on the above example for swapping the adverb with the auxiliary. To do this, simply type the initial phrase into an empty document and then search for (rarely) (have) and replace this by \2 \1, making sure that the option for regex replacements is switched on. You'll probably learn to appreciate backreferences much more once you've learnt about how to construct linguistic annotations in Chapter 11, where you'll be able to convert some original data into a special coding format using techniques such as the one above to carry out the conversion more efficiently.

6.4.3 Quoting and using special characters

Now we've discussed almost all the essential details to begin using regexes properly inside a concordancer, apart from one little detail. If certain characters, such as the dot/full stop (.), question mark, or caret, etc., have a special meaning for the program interpreting the regex, how do we actually specify those characters if we explicitly want to look for them? Well, this is done by *escaping* them, which is generally achieved by preceding them by a backslash (\), that gives them back their literal meaning. Therefore, in order to search for a full stop, you simply write \. (but note that this is not necessary inside a character class, where all punctuation marks retain their 'punctuation meaning').

For linguistic purposes, when we quantify character classes or shorthands, we'll mainly search for and use constructs that represent word characters, for example, something like [a-z]+ or \w+, which is somewhat safer than using a . to simply indicate any character. This is because quantifying the latter will literally find any character at all, including spaces, line breaks, as well as even special control characters that only the computer may understand, or even word boundaries themselves. Thus, if we, for example, want to look for words with particular beginnings or ends and think that, instead of using appropriate constructs representing word characters, we can be lazy and just write the dot followed by an appropriate quantifier, we may end up with rather unexpected results. For instance, if we wanted to look for all pronouns that start with the character sequence <the>, i.e. *them, these, their*, etc., and just specified \bthe.+ \b, assuming that what this would find starts with said combination and ends at the next word boundary, we'd be in for a surprise. This is because what it would in fact generally match is the very first occurrence of the character sequence, followed by the rest of the text, that is, a combination of word characters, whitespaces, punctuation marks, etc., until we reach the end of the text itself, which is – perhaps not so obviously – also a kind of word boundary. This happens because most regex engines assume that quantification is by default *greedy*, that is, supposed to match as many characters as possible. In order to avoid this problem, and still use our 'lazy' version of a shorthand, we have to make the quantification *non-greedy*, which is done by adding another question mark after the first quantifier. In our case, we'd then have to write \bthe.+? \b, which would

stop the match as soon as the regex finds the first word boundary following the quantified shorthand expression, also finding non-pronouns like *theme*, *thematic*, *thesis*, etc., though.

6.4.4 Constraining the context further

In some cases, rather than just specifying everything we want to find, we also need to be able to indicate that something either should or should not occur in an environment. To some extent, we've already done this when we used word boundaries above, because whenever we inserted a boundary marker in our regex constructs, we effectively said "don't allow another word character to occur here", thereby constraining the options for a match. To return to our example of *and* from Section 5.4.1, where we wrote `\band\b`, we stated that there should be no word character preceding the `<a>`, and no word character following the `<d>`. This is a very powerful construct, but only allows us to constrain one particular environment, that is, the beginning or end of a word we're looking for. Sometimes, however, we also need to be able to constrain certain patterns inside or around parts of a word or a grouping, in which case we need other ways of being able to indicate that we either don't want something to precede or follow a specific pattern. This can be achieved by using so-called *lookaround*, which is subdivided into *lookahead* and *lookbehind*. The former allows us to constrain what is (positive lookahead) or is not (negative lookahead) supposed to *follow*, while the latter provides positive and negative options for whatever is or is not supposed to *precede*, our search term. As groupings normally capture, and we don't want whatever is supposed to be constraining our grouping to become part of the match, and thus be highlighted by the concordancer, these constructs actually use a special syntax that excludes them from being captured, which is that the opening bracket of the grouping parentheses is immediately followed by a question mark, i.e. `(?...)`.

To illustrate the usefulness of such a feature, let's assume you want to teach students vocabulary about different types of containers to store things in, in which case you may want to run a search for compounds involving the words *box* and *case* to find all singulars and plurals. From what we already know, we can easily achieve looking for the *stems* of these compounds by writing the following regex: `(boxe?|case) s?`. However, using this expression, we're not only bound to find many occurrences of the stems themselves (which are relatively uninteresting), but at the same time also retrieve instances of the word *case* in its legal meaning or as part of the prepositional phrase *in case*. To avoid this, we can use negative lookbehind like this, `(?<!\b)(boxe?|case) s?`, stating that the stem should not be immediately preceded by a word boundary. The counterpart, positive lookbehind, can for instance be used to find all words at the beginning of a sentence, using the following expression: `(?<=[. !?])\w+\b`. As you'll hopefully already be able to guess, this finds all words that follow a major punctuation

mark and a space, but unfortunately misses all first words in paragraph-initial sentences, as these are not preceded by punctuation. Now, unfortunately, we cannot simply combine these searches because the one limiting feature of lookbehind in most regex implementations currently is that they have to be fixed, so that we have no option but to run both searches and then combine the results, if necessary.

Lookahead, probably because it's easier to implement, doesn't suffer from this limitation, so that we can specify more complex expressions, such as the counterpart of the one we just tried, in one go, i.e. `\b\w+(?=[.!?]|$))`, which finds all words that occur at the end of a sentence and also the end of a paragraph, along with, unfortunately, Roman ordinal numbers and some abbreviations that may occur in the middle of the sentence. Negative lookahead basically allows us to exclude unwanted constructions, such as parts of a paradigm. Thus, if we wanted to find all but the ing-forms of the verb *want*, we could say `\bwant(?!ing)\w*\b`, where we first ensure that `<ing>` cannot follow the stem, and then say that any other word characters may or may not follow before the boundary. Of course, depending on your corpus, you may also find some rather unexpected words that have nothing whatsoever to do with the verb *want*; for instance, because my test corpus for trying out regexes contains more 'archaic' language, I also found the adjective *wanton*, as well as some other constructions, this way. In comparison to the genuine expected results, though, those were extremely small and easy to weed out after sorting.

6.5 Further Exercises

In order to do these exercises, you should have AntConc installed on your system or on a memory stick. The exercises are based on the Project Gutenberg e-text of *Wuthering Heights*, which you can obtain from Project Gutenberg (<http://www.gutenberg.org/>) and clean up as we've practised before.

Exercise 32

Start AntConc and type in `hand` as your search term. View the results to see how many different words or word forms you get. Tip: It's best if you sort your results.

Next, check the tick mark for 'Regex', run the search again, and observe/interpret the results to see what additional useful or not-so-useful information may have been added to the results.

Next, in our first attempt to narrow down our search to forms we know belong to either the noun or verb paradigm, specify `hand(ed|ing|s)?`

instead. Check the results and see how/whether they differ from what you found before.

Change your search term to include a `\b` at the end of your search term and observe the difference.

Now, add `\b` at the beginning of the term, too, and compare the results.

Last, add `[^-]` at the beginning of the term and check the result again.

Some more, diverse, exercises:

Exercise 33

Do a search for all occurrences of the ‘verbs of belief’, *guess*, *suppose*, *think*, *believe* and *assume*.

In the next step, add all possible inflectional endings to the search. What will you fail to catch with this?

Do the same for *see* and *understand*. Do you encounter similar problems here?

For all the above exercises, it obviously doesn’t make sense to only do them purely mechanically, but you should always analyse and evaluate the results very carefully. This generally involves keeping a record of how you achieved the results by setting up and possibly modifying your regex until you’ve managed to find all the expected results, keeping track of interesting, but unexpected, ones as well as saving some suitable examples and discussing them with regard to their (morpho-) syntax, semantics, pragmatics, possible frequency distributions, etc.

Solutions to/Comments on the Exercises

Exercise 26

Character classes clearly represent one of the features referred to above, where we would get too many, and probably also linguistically uninteresting, search results, if we simply look for these in a corpus. However, what you’ll hopefully have learnt from the online demo display is that a character class on its own simply represents alternative options for finding a *single character*, which is why I’ve chosen to surround all instances by a little extra space to make the individual characters stand out more clearly. When using character classes, and also partly when looking at the most salient examples in the list of basic examples above, it’s perhaps easy to be confused into thinking that character classes always represent consecutive

ranges, but of course they can represent any kind of grouping that may be (linguistically) meaningful in any way, as in, for example, `[mnn]`, which would find all nasal consonants in a corpus of phonetic transcriptions.

Exercise 27

In my description of the `\w` word characters definition, I deliberately used the word “theoretically” when stating that this construct should also match other characters that occur in words, in whichever way or character set they’re transcribed. Unfortunately, though, JavaScript, which I used to create the online exercises, only seems to take characters from the traditional Latin (ASCII) character set as being word characters, which is why the phonetic characters are also not selected. However, when you use the ‘any character’ construct, i.e. the single dot (`.`), the phonetic characters are at least also recognised as characters. I also tested the same feature with some Chinese characters, which are clearly word characters, but again these were not recognised, which again proves that we always need to test any regex constructs we use in different programs to see whether they’re going to be recognised properly. In this case, JavaScript has shown that its implementation of regexes is still very limited, but many other languages or programs – including AntConc – understand many more advanced regex options.

One further thing that may have been confusing when you tested for whitespace is that around all the spaces you will probably have noticed strange so-called ‘pipe’ symbols (`|`). These indicate word boundaries and their being displayed here is due to some tricks I had to use in writing the script to make it display everything the way I wanted it displayed, so you can safely ignore them ☺.

Exercise 28

As stated below the exercise itself, it’s quite easy to get confused about what negation does to a character class, partly because we may sometimes get the impression that we’re actually just turning around a definition if we negate a range inside a character class. However, although, for instance, negating the original character class `[A-Z]` (i.e. all English uppercase letters) to `[^A-Z]` will predominantly have the effect of now showing us/selecting all lowercase letters, it will also show/select non-word characters, such as punctuation, spaces, or numbers, simply because they’re obviously also non-uppercase characters. Thus, one thing you may need to do when defining/using a negative character class, apart from thinking about it carefully anyway, is to not think in terms of binary oppositions.

Exercise 29

Essentially, the two basic examples in this exercise have allowed you to explore a number of things:

- a) how to specify one optional single character through the use of a question mark: the difference between *colour* and *color*, or the presence/absence of a 3rd person singular or plural *s*, provided that you add a `\s` after the question mark, which will then find only those instances that are not followed by a punctuation mark,
- b) how to possibly identify words that contain multiple occurrences of the same character, such as double *f*, *s*, or *l*, or even words with duplicated vowel letters such as the word *good*,
- c) how to identify words of different length, but only if they're in fact surrounded by whitespace.

Examples of type a) allow us to work with different dialectal variants, or sometimes also to cover spelling variants in historical texts, where the spelling may not yet have been standardised, so that even one and the same text may contain alternate forms that represent exactly the same word meaning. In addition, we can already model very basic morphological features, such as the plurals referred to above, or vowel alternation in irregular verbs that may signify differences in tense, for example, *give* vs. *gave*, or in a small number of singular–plural distinctions for nouns, such as in *man–men*. Examples of type b) make it possible to identify phonotactic features, such as the presence or absence of reduplication and its effect on pronunciation, while type c) may be useful for selecting or extracting words of different length in order to establish potential correlations between word length and complexity, or, if we assume that shorter words are indeed less complex, to extract simpler vocabulary from texts in order to teach it at less advanced learner levels. Obviously, though, to what extent we'll in fact be able to use these regular expression constructs in the ways described here again depends on which particular program we're using and how regexes are implemented there.

Of course, although we've already made great progress towards identifying words, we still have the issue of words not surrounded by whitespace to consider, but we'll soon find a solution for this, too.

Exercise 30

When scanning the printed sample text visually, you'll hopefully already have spotted that the character sequence `<and>` not only occurs in the conjunction, but also as the final part of the word *understand*. If not, then the online exercise will have revealed this very quickly, although you may still have had to look carefully to see it. Of course, now knowing that this sequence may occur inside another word, we could make it easier to highlight such examples simply by adding a `\w` in front of the character sequence `<and>` in our search, thereby specifying that it has to be preceded by at least another word character. However, this still wouldn't allow us to identify all occurrences where the sequence occurs at the very beginning of the word, so we need to be even more precise and formulate our regex like this: `\b\w*and\w*\b`. If you cannot immediately recognise what this means, then I

suggest you spend a few minutes thinking about this and maybe even test it on a longer text in AntConc, as described in Section 6.5.

As I hope you'll also have seen, by using the boundary markers in combination with the shorthand for word characters, you'll now definitely be able to look for words of fixed or variable length properly, where, for example, `\b\w{2}\b` will find all words that are exactly two characters long, `\b\w{2,5}\b` between two and five characters, and `\b\w{2,}\b` at least 2 characters long.

Exercise 31

Your first attempt at finding all verbs will probably be to simply identify all existing verb forms in the sample paragraph and group them together in one fixed regex that matches everything exactly as it occurs in the text, which would probably look like the following string: `(allow|test|displaying|\bbe\b|contains|Can|understand|based|concerning|make)`, where hopefully you'll already have realised that the sequence `<be>` can also be part of another word and thus we'd at least have to add boundary markers around it to only find the base form of the verb. However, if you think about this more closely from a linguistic perspective, doing this only allows you to find what's really there, but wouldn't be suitable for identifying all the different potential word (paradigm) forms of the verbs we may be interested in if we were actually researching them. In order to properly investigate all the different verb forms, we'd minimally have to try and specify a regex that groups the base forms of the verbs together, and then adds another optional grouping that expresses all the possible verb endings that could be present, and which may then initially end up looking somewhat similar to the following, already considerably more complex, construct: `(allow|test|display|\bbe\b|contain|Can|understand|base|concern|make)(e?d|ing|s)?`. This regex, though, still has a number of problems, not least of all that it would also potentially find the word *shallow*, so that we'd at least need to use a boundary marker before the word *allow*, too. Other things the expression may also find, and which we may or may not actually want it to find, are the 'negative counterparts' of some of the verbs (or nouns) that carry a negative prefix, for example, *disallow*, *misunderstand(ing)*, *debase(d)*, *unconcerned*, and *unmake*. If we want to exclude some of these, but still not the noun forms *understanding*, *base(s)*, *concern(s)*, or *make(s)*, not to forget capitalised *can*, which was yet another issue above, then we'd at least have to add a boundary marker before the whole group, and ideally also after it, thus yielding something along the lines of or at least close to this: `\b(allow|test|display|be|contain|Can|understand|base|concern|make)(e?d|ing|s)?\b`. To make this expression completely flexible to cover almost everything we want it to find, we'd then still need to adapt it a little more by allowing for initial small (lowercase) and capital letters: `\b([Aa]llow|[Tt]est|[Dd]isplay|[Bb]e|[Cc]ontain|[Cc]an|[Uu]nderstand|[Bb]ase?|[Cc]oncern|[Mm]ake)(ed|ing|s)?\b`.

Now, if you have a keen eye for, or at least some substantial training in, morphology, you'll also have spotted that, among all the regular verbs, we have two irregular ones that we haven't dealt with properly. The first one, *make*, is relatively easy to fix because we only need to add a character class that specifies the consonant options, i.e. `[dk]`, as well as make the final `<e>` optional, so that we can also 'create' the present participle form, which, incidentally, we also have to do for the verb *base*. For the verb *be*, though, we have no choice but to list the full paradigm. So, the 'final' version of our regex would then be: `\b([Aa]llow|[Tt]est|[Dd]isplay|[Aa](m|re)|[Bb]e(en)?\b|[Ii]s|[Ww](as|ere)|[Cc]ontain|[Cc]an|[Uu]nderstand|[Bb]ase?|[Cc]oncern|[Mmdk]e?)(e?d|ing|s)?\b`. Now, I put the word *final* above in scare quotes because of course this regex still wouldn't be able to handle the negative contraction forms of the BE paradigm, which, however, I leave up to you to add as another exercise. Please also note that our regex would of course 'overgenerate', that is, match more than just legitimate words, e.g. also errors like *maked*, which could still occur, for example, in child language or learner interlanguage. One other thing we need to be aware of here is that the above regex could still be made more compact, albeit less readable, to the novice, if we grouped all the words that start with the same letter together, which would obviously lead to more brackets and pipe symbols within brackets.

I'll leave the second part of the exercise, to identify all the nouns, to you. Now that you're aware of the main issues, you should probably be able to handle this one relatively 'easily' if you bear in mind that nouns do have singular and plural forms, etc. Unfortunately, though, given our current means, and even using the most sophisticated regexes, we still have no means of distinguishing between grammatically polysemous nouns and verbs. To solve this problem, though, we'll soon look into ways of enriching our data with word-class information.

Furthermore, now that you've gone through a lot of trouble to painstakingly create two sets of fairly complex regexes, you'll probably want to test them on more than just my short paragraph, so I'd suggest that you open a file in AntConc, paste the regexes in the search box, tick the 'Regex' option, and see what the results will be...

Exercise 32

When you do a basic word search for *hand* in AntConc, not using the regex feature, you should get 94 hits altogether. To distinguish between your examples, you can simply sort on the first word to the left (Level 1: 1L). This should allow you to quickly establish that there are no occurrences of *hand* as a verb, but instead a number of different noun uses, ranging from the basic noun (phrase) use (*a hand, her hand, an unformed, childish hand*), via temporal/locative prepositional/circumstantial phrases (*at hand, on either hand, at your left hand, by the hand*), idiomatic/metaphorical expressions (*take him in hand, try my hand at, came to hand; He shall have his share of my hand* [i.e. be beaten], *offering the right*

hand of fellowship), to textual deixis (*on the one hand*). Furthermore, there's also a single compound noun, *minute-hand*.

Once you tick the 'Regex' option, you'll probably be very surprised that you suddenly end up with 199 hits instead of the original 94. Sorting the hits again, this time by the word itself (Level 1: 0), and then by the first word on the left (Level 2: 1L) quickly shows us that now we not only find many more forms of the noun, including plurals, plus a few ED-forms of the verb (commonly referred to as 'past participles'), which we did want to find, but also quite a few words that simply contain the character sequence <hand>, such as, for example, *handsome* or even *chandelier*, which we definitely didn't want to find.

When trying to narrow down our search to noun or verb forms of *hand* by specifying the regex `hand(ed|ing|s)?`, unfortunately we still get all of the 199 hits that the most basic regex string has already given us, so we definitely need to explore better ways of limiting our search via additional regex options. The most basic thing to do here is to 'say' that the verb/noun 'endings' we specified really have to occur at the very end, and 'word end' here means 'word boundary', i.e. `\b`. Please note that, even if you may have assumed that we could simply replace the `?` quantifier in the regex by a `+` to force the endings to occur at least once, this simply would not work because we would then be excluding all uninflected forms.

Adding the boundary marker at the end now already reduces the number of hits to 159, that is, excluding 40 unwanted hits, but still leaves us with some unwanted cases, *beforehand*, *close-handed* or *minute-hand*. Adding another boundary marker at the beginning of our regex now eliminates the 2 occurrences of *beforehand*, reducing the number to 157, but still leaving us with *close-handed* and *minute-hand* because hyphens are generally considered to be parts of words, for instance many compound nouns, so, in order to get exactly the results we want, we also need to exclude the hyphen, thus making the final regex `\b[^\-]hand(ed|ing|s)?\b` and only leaving us with 155 relevant results.

Exercise 33

As in the previous exercise, we'll develop the solution here as far as possible, in a step-by-step manner in order to 'fine-tune' the regex as much as possible. I'd thus suggest that, at every stage, you try out the adjusted regex and observe the results closely to enable you to understand the process better. To begin exploring the options here, you'll probably want to start with the simple pattern `\b(assume|believe|guess|suppose|think)\b`. To make it easier to read, I've already put the words into alphabetically sorted order, which has the added advantage of speeding up the search process a little because regexes actually try to match in alphabetical order. This first attempt will give us 210 hits, but unfortunately miss all the inflected forms.

In the next step, to add inflectional endings, and having noticed that the final <e>s in *assume* and *believe* and *suppose* may actually need to be

made optional in the groupings for the inflections for the sake of creating a more consistent pattern, we can already change the pattern to `\b(assum|believ|guess|suppos|think)(e?[ds]?|ing)?\b`. This will give us 284 hits that are already fairly close to what we want to be able to find, but still of course misses the ED-form of *think*, *thought*.

Adding an option for this will then provide us with an improved solution for the regex, `\b(((assum|believ|guess|suppos|think)(e?[ds]?|ing)?|thought)\b`, increasing the number of hits to 414, and indicating the importance of the past tense form of *think* in the text. However, after sorting the list on the search term and scrolling down to the end of the hits, we can see that our attempt to cover all inflectional verb endings has inadvertently led to our also finding a number of occurrences of the plural of the noun *thought*, which is definitely something we'd like to avoid if possible.

Now, obviously, we don't really want to have to specify all the verb paradigms for each verb separately, but, on the other hand, cannot simply exclude the 3rd person singular present {s} morpheme because it's such an important part of the paradigm for verbs. Luckily, though, the {s} morpheme is not a part of the paradigm for the past tense forms, where we only have *thought* in both the simple past and perfect tense forms. Therefore, what we can do here is use a little negative lookahead to exclude the <s> from our pattern definition for the past tense form, yielding `\b(assum|believ|guess|suppos|th(ink|ought(?!s)))(e?[ds]?|ing)?\b`, and reducing the final number of hits to 400.

In this way, we've now managed to exclude all the plural forms of the noun *thought*, but are still left with a number of occurrences of the singular noun, as in:

- 1 *every thought*
- 2 *an evil thought*
- 3 *Catherine's first thought*
- 4 *there lay immense consolation in that thought*
- 5 *ere they had a thought to spend for any*
- 6 *Hareton seemed a thought bothered*
- 7 *while I can bestow a thought on the matter*
- 8 *for every thought she spends on Linton*
- 9 *Our first thought*
- 10 *my great thought in living*
- 11 *no thought of worldly affairs*
- 12 *the slightest act not prompted by one thought*
- 13 *sometimes shuddered at the thought of.*

Looking at this list, we may assume that, in order to improve our search results, we could adjust to restrict our search to only finding instances of *thought* not preceded by a determiner or quantifier, which, from a linguistic point of view, is obviously correct. However, unfortunately the one construct we know that should

allow us to express such restrictions, that is, negative lookbehind, is (currently still) restricted to a single fixed pattern, so we could maximally exclude one of our options, in this case possibly <a> because it occurs marginally more frequently than any other form, but still only three times. Although, of course, we could easily add the negative lookbehind for <a>, this seems hardly worth it, especially because the effect is negligible, and because sorting the hits according to the first word on the left will relatively easily allow us to identify and delete all noun occurrences of *thought*. For the moment, we therefore simply need to accept that we have no ideal way to exclude the nouns from our search until we learn how to work with morpho-syntactically annotated versions of our data, which will then allow us to only look for verb forms much more easily.

A similar problem to the one above could also exist for the singular and plural forms of the noun *guess* because they look identical to the base form of the verb and the 3rd person singular present, but luckily an investigation of the data reveals that those don't actually occur in our text. If they did, we'd have to resort to the same solution again.

Regarding the search for *see* and *understand*, the first thing to note would be that the past tense forms are both irregular, so that the regex 'suffix' pattern would only need to include the 3rd person singular and ing-form, with one minor addition, the <n> that allows us to create the ED-form of *see*, while all other irregular forms would need to be specified as part of the stem allomorph pattern as in `\b(s(a|ee)|underst(an|oo)d)(ing|[ns])?\b`, which is the most compact and efficient way of writing this expression. This pattern should return 249 hits, out of which, luckily, only a single one represents a case we'd want to exclude, which is *It is right to establish a good **understanding** at the beginning*, where the word *understanding* is, of course, a deverbal noun.

Sources and Further Reading

- Friedl, J. (2006). *Mastering Regular Expressions* (3rd ed.). Sebastopol, CA: O'Reilly.
 Weisser, M. (2009). *Essential Programming for Linguistics*. Edinburgh: EUP.

7

Understanding Part-of-Speech Tagging and Its Uses

This chapter will explore one of the main breakthroughs in corpus linguistics, *morpho-syntactic annotation*, which is also referred to as *Part-of-Speech* (or PoS) *tagging*. This kind of linguistic technology makes it possible to enrich data with information that facilitates extracting and analysing specific word forms or constructions in a way that's more advanced than what we've discussed previously. However, despite this obvious advantage, we still need to be cautious and aware of its limitations when using it. Thus, the following sections will try to provide you with a rough overview of what exactly PoS tagging is and how it can be carried out, where its strengths and weaknesses lie, and how you may be able to use it with your own data.

The following is an extract from the beginning of the Brown Corpus that illustrates what basic PoS-tagged text may look like, although the exact format and conventions for representing the tags may vary for different *annotation schemes*:

The_AT Fulton_NP1 County_NN1 Grand_JJ Jury_NN1 said_VVD Friday_NPD1
an_AT1 investigation_NN1 of_IO Atlanta_NP1's_GE recent_JJ primary_JJ elec-
tion_NN1 produced_VVD no_AT evidence_NN1 that_CST any_DD irregulari-
ties_NN2 took_VVD place_NN1 ._.

For the moment, you don't need to understand what the tags mean, as we'll soon explore which different bits of information may be contained in a PoS tag.

As we've seen in some of our previous exercises and discussions, frequently one and the same word form may have different meanings or functions, that is,

be semantically or grammatically polysemous. Very frequently, this polysemy is in fact of a morpho-syntactic nature because these differences in meaning depend on the word class associated with the word form in a particular context, as well as potentially its inflection. Remember the example of the duplicated word form *that* in Section 6.4.2, where the first occurrence was a relative pronoun and the second a demonstrative determiner? This type of grammatical polysemy is actually far more likely than you may assume, although it rarely occurs in such reduplicated word form contexts, but more frequently in the shape of words that are commonly assumed to be the products of *zero-derivation* or *conversion*. The following table, based on DeRose (1988), lists the number and degree of ambiguity found in the word-class labels – in corpus linguistics circles better known as *PoS tags* – assigned to words in the Brown Corpus.

Table 7.1 Ambiguous PoS tags in the Brown Corpus

<i>Degree of Ambiguity</i>	<i>Number of Words</i>	<i>Percentage</i>
unambiguous (1 tag)	35,340	89.604%
ambiguous (2-7 tags)	4,100	10.396%
2 tags	3,760	9.533%
3 tags	264	0.669%
4 tags	61	0.155%
5 tags	12	0.030%
6 tags	2	0.005%
7 tags	1 (<i>still</i>)	0.003%

Here, we can see that slightly more than 10% of all word forms that occur in the Brown Corpus are in fact ambiguous. This means that, even in a comparatively small corpus like the Brown, one out of ten words may become a potential source of error in our analysis, unless we check our results carefully! For the majority of these, there are only two options, with the likelihood of a word belonging to more than two word classes dropping drastically, but still being a possibility, although the more extreme cases do get very rare indeed.

As stated above, some of the high potential for grammatical polysemy in English stems from what is generally described as zero-derivation in morphology, that is, the ability of certain words to change their word class without any form of inflection or affixation. Most of the instances of this tend to involve noun–verb pairs that need to be disambiguated by their context in writing. Sometimes, in spoken language, we also have the option to recognise them through their differing stress and/or pronunciation patterns, for example, *house* (noun; /haʊs/) vs. *house* (verb; /haʊz/) or *insult* (noun; /'ɪnsʌlt/) vs. *insult* (noun; /ɪn'sʌlt/), etc., but of course, without additional grammatical information, this would only be possible in phonetically transcribed corpora.

For this and other reasons, it's highly useful to have corpora enriched with morpho-syntactic tags. At least some of the existing corpora that we'll be using

here already do contain such information, and we'll also explore ways of getting our own data tagged to some extent later. Being able to make use of such extra information will not only help us to distinguish between word forms that are grammatically polysemous, but also to look for collocational or colligational patterns (see Chapter 10), in other words, to investigate formulaic language or language patterns more easily.

7.1 A Brief Introduction to (Morpho-Syntactic) Tagsets

To be able to understand exactly what type of information we can usually get from many tagsets, let's start by exploring a relatively simple tagset for English, the Penn Treebank Tagset, depicted in Table 7.2, which only contains 48 tags, a number that already exceeds, to a fairly high extent, that of the traditional word classes you'll be familiar with from school or any basic or advanced linguistics courses, though.

Table 7.2 The Penn Treebank tagset (based on Taylor et al. 2003: 8)

<i>Tag</i>	<i>Description</i>
CC	Coordinating conjunction
CD	Cardinal number
DT	Determiner
EX	Existential there
FW	Foreign word
IN	Preposition or subordinating conjunction
JJ	Adjective
JJR	Adjective, comparative
JJS	Adjective, superlative
LS	List item marker
MD	Modal
NN	Noun, singular or mass
NNS	Noun, plural
NP	Proper noun, singular
NPS	Proper noun, plural
PDT	Predeterminer
POS	Possessive ending
PP	Personal pronoun
PP\$	Possessive pronoun
RB	Adverb
RBR	Adverb, comparative
RBS	Adverb, superlative
RP	Particle
SYM	Symbol

Table 7.2 The Penn Treebank tagset (based on Taylor et al. 2003: 8)
(Continued)

<i>Tag</i>	<i>Description</i>
TO	to
UH	Interjection
VB	Verb, base form
VBD	Verb, past tense
VBG	Verb, gerund or present participle
VBN	Verb, past participle
VBP	Verb, non-3rd person singular present
VBZ	Verb, 3rd person singular present
WDT	Wh-determiner
WP	Wh-pronoun
WP\$	Possessive wh-pronoun
WRB	Wh-adverb
#	Pound sign (AmEn), hash mark (BrEn)
\$	Dollar sign
.	Sentence-final punctuation
,	Comma
:	Colon, semi-colon
(Left bracket character
)	Right bracket character
"	Straight double quote
'	Left open single quote
“	Left open double quote
’	Right close single quote
”	Right close double quote

Exercise 34

Look through the list of tags in Table 7.2 and check to see whether you're familiar with all the grammatical (sub-)categories listed in the right-hand 'Description' column, or can at least roughly imagine what the individual tags may stand for.

Do an online search for categories you're unfamiliar with, and try to see whether this will allow you to understand the categories better.

Think about whether there are any tags that may be unusual, either in the sense that they represent independent categories or that they may be 'lumping together' categories you'd normally have seen as being separate.

As you'll hopefully already have observed when going through the tagset, typically a morpho-syntactic tag will consist of one (in the case of punctuation in the

Penn tagset) or more, often three, (capital) letters or special characters, beginning with one that represents the ‘major PoS’, that is, noun, verb, adjective, adverb, etc. Usually, and whenever possible, the designers of tagsets try to choose a mnemonic for this initial letter, based on the first letter of the word class itself (e.g. N for **n**oun, V for **v**erb). But of course, due to the fact that the names of some word classes start with the same letter (e.g. article/adjective/adverb; pronoun/preposition/particle), this is not always possible. In this case, an alternative letter that doesn’t constitute the beginning of any word class will be used, for example, commonly J for ad**j**ectives, R for adve**r**bs, where at least the letter used tends to be part of the word class name. The second character then often represents a form of sub-categorisation – such as N or P for nouns, where the N can be seen as a mnemonic for ge**n**eral and P for **p**roper –, if applicable, whereas the third (or sometimes whichever is the final one) often stands for other properties of the word, such as number or person, etc. Those are just very general rules, though, and depending on the exact number of distinctions made – i.e. how fine-grained it is – a tagset may also contain tags that are much longer. This is especially the case for more strongly inflected languages, as more distinctions can, and need to be, made.

Let’s explore this a little further by looking at another of the currently best-known tagsets, the *CLAWS* (Constituent Likelihood Automatic Word-tagging System) *C7* Tagset, which is already far more detailed at 152 tags, exceeding the 48 tags observed in the Penn tagset by 104 tags.

Table 7.3 The *CLAWS 7 (C7)* tagset

<i>Tag</i>	<i>Description</i>
!	punctuation tag - exclamation mark
" " " "	punctuation tag - quotes
(punctuation tag - left bracket
)	punctuation tag - right bracket
,	punctuation tag - comma
-	punctuation tag - dash
----	new sentence marker
.	punctuation tag - full-stop
...	punctuation tag - ellipsis
:	punctuation tag - colon
;	punctuation tag - semicolon
?	punctuation tag - question mark
APPGE	possessive pronoun, pre-nominal (e.g. <i>my, your, our</i>)
AT	article (e.g. <i>the, no</i>)
AT1	singular article (e.g. <i>a, an, every</i>)
BCL	before - clause marker (e.g. <i>in order (that), in order (to)</i>)
CC	coordinating conjunction (e.g. <i>and, or</i>)
CCB	adversative coordinating conjunction (<i>but</i>)
CS	subordinating conjunction (e.g. <i>if, because, unless, so, for</i>)

(Continued)

Table 7.3 The CLAWS 7 (C7) tagset (*Continued*)

<i>Tag</i>	<i>Description</i>
CSA	<i>as</i> (as conjunction)
CSN	<i>than</i> (as conjunction)
CST	<i>that</i> (as conjunction)
CSW	<i>whether</i> (as conjunction)
DA	after-determiner or post-determiner capable of pronominal function (e.g. <i>such, former, same</i>)
DA1	singular after-determiner (e.g. <i>little, much</i>)
DA2	plural after-determiner (e.g. <i>few, several, many</i>)
DAR	comparative after-determiner (e.g. <i>more, less, fewer</i>)
DAT	superlative after-determiner (e.g. <i>most, least, fewest</i>)
DB	before determiner or pre-determiner capable of pronominal function (<i>all, half</i>)
DB2	plural before-determiner (<i>both</i>)
DD	determiner (capable of pronominal function) (e.g. <i>any, some</i>)
DD1	singular determiner (e.g. <i>this, that, another</i>)
DD2	plural determiner (<i>these, those</i>)
DDQ	wh-determiner (<i>which, what</i>)
DDQGE	wh-determiner, genitive (<i>whose</i>)
DDQV	wh-ever determiner (<i>whichever, whatever</i>)
EX	existential there
FO	formula
FU	unclassified word
FW	foreign word
GE	Germanic genitive marker - (' or 's)
IF	<i>for</i> (as preposition)
II	general preposition
IO	<i>of</i> (as preposition)
IW	<i>with, without</i> (as prepositions)
JJ	general adjective
JJR	general comparative adjective (e.g. <i>older, better, stronger</i>)
JJT	general superlative adjective (e.g. <i>oldest, best, strongest</i>)
JK	catenative adjective (<i>able</i> in <i>be able to, willing</i> in <i>be willing to</i>)
MC	cardinal number, neutral for number (<i>two, three ..</i>)
MC1	singular cardinal number (<i>one</i>)
MC2	plural cardinal number (e.g. <i>sixes, sevens</i>)
MCGE	genitive cardinal number, neutral for number (<i>two's, 100's</i>)
MCMC	hyphenated number (<i>40-50, 1770-1827</i>)
MD	ordinal number (e.g. <i>first, second, next, last</i>)
MF	fraction, neutral for number (e.g. <i>quarters, two-thirds</i>)
ND1	singular noun of direction (e.g. <i>north, southeast</i>)
NN	common noun, neutral for number (e.g. <i>sheep, cod, headquarters</i>)
NN1	singular common noun (e.g. <i>book, girl</i>)
NN2	plural common noun (e.g. <i>books, girls</i>)
NNA	following noun of title (e.g. <i>M.A.</i>)
NNB	preceding noun of title (e.g. <i>Mr., Prof.</i>)
NNJ	organization noun, neutral for number (e.g. <i>council, department</i>)

Table 7.3 The CLAWS 7 (C7) tagset (*Continued*)

<i>Tag</i>	<i>Description</i>
NNJ2	organization noun, plural (e.g. <i>governments, committees</i>)
NNL1	singular locative noun (e.g. <i>island, street</i>)
NNL2	plural locative noun (e.g. <i>islands, streets</i>)
NNO	numeral noun, neutral for number (e.g. <i>dozen, hundred</i>)
NNO2	numeral noun, plural (e.g. <i>hundreds, thousands</i>)
NNT1	temporal noun, singular (e.g. <i>day, week, year</i>)
NNT2	temporal noun, plural (e.g. <i>days, weeks, years</i>)
NNU	unit of measurement, neutral for number (e.g. <i>in, cc</i>)
NNU1	singular unit of measurement (e.g. <i>inch, centimetre</i>)
NNU2	plural unit of measurement (e.g. <i>ins., feet</i>)
NP	proper noun, neutral for number (e.g. <i>IBM, Andes</i>)
NP1	singular proper noun (e.g. <i>London, Jane, Frederick</i>)
NP2	plural proper noun (e.g. <i>Browns, Reagans, Koreas</i>)
NPD1	singular weekday noun (e.g. <i>Sunday</i>)
NPD2	plural weekday noun (e.g. <i>Sundays</i>)
NPM1	singular month noun (e.g. <i>October</i>)
NPM2	plural month noun (e.g. <i>Octobers</i>)
NULL	the null tag, for words which receive no tag
PN	indefinite pronoun, neutral for number (<i>none</i>)
PN1	indefinite pronoun, singular (e.g. <i>anyone, everything, nobody, one</i>)
PNQO	objective wh-pronoun (<i>whom</i>)
PNQS	subjective wh-pronoun (<i>who</i>)
PNQV	wh-ever pronoun (<i>whoever</i>)
PNX1	reflexive indefinite pronoun (<i>oneself</i>)
PPGE	nominal possessive personal pronoun (e.g. <i>mine, yours</i>)
PPH1	3rd person sing. neuter personal pronoun (<i>it</i>)
PPHO1	3rd person sing. objective personal pronoun (<i>him, her</i>)
PPHO2	3rd person plural objective personal pronoun (<i>them</i>)
PPHS1	3rd person sing. subjective personal pronoun (<i>he, she</i>)
PPHS2	3rd person plural subjective personal pronoun (<i>they</i>)
PPIO1	1st person sing. objective personal pronoun (<i>me</i>)
PPIO2	1st person plural objective personal pronoun (<i>us</i>)
PPIS1	1st person sing. subjective personal pronoun (<i>I</i>)
PPIS2	1st person plural subjective personal pronoun (<i>we</i>)
PPX1	singular reflexive personal pronoun (e.g. <i>yourself, itself</i>)
PPX2	plural reflexive personal pronoun (e.g. <i>yourselves, themselves</i>)
PPY	2nd person personal pronoun (<i>you</i>)
RA	adverb, after nominal head (e.g. <i>else, galore</i>)
REX	adverb introducing appositional constructions (<i>namely, e.g.</i>)
RG	degree adverb (<i>very, so, too</i>)
RGQ	wh-degree adverb (<i>how</i>)
RGQV	wh-ever degree adverb (<i>however</i>)
RGR	comparative degree adverb (<i>more, less</i>)
RGT	superlative degree adverb (<i>most, least</i>)

(Continued)

Table 7.3 The CLAWS 7 (C7) tagset (*Continued*)

<i>Tag</i>	<i>Description</i>
RL	locative adverb (e.g. <i>alongside, forward</i>)
RP	prep. adverb, particle (e.g. <i>about, in</i>)
RPK	prep. adv., catenative (<i>about</i> in <i>be about to</i>)
RR	general adverb
RRQ	wh-general adverb (<i>where, when, why, how</i>)
RRQV	wh-ever general adverb (<i>wherever, whenever</i>)
RRR	comparative general adverb (e.g. <i>better, longer</i>)
RRT	superlative general adverb (e.g. <i>best, longest</i>)
RT	quasi-nominal adverb of time (e.g. <i>now, tomorrow</i>)
TO	infinitive marker (<i>to</i>)
UH	interjection (e.g. <i>oh, yes, um</i>)
VB0	<i>be</i> base form (finite i.e. imperative, subjunctive)
VBDR	<i>were</i>
VBDZ	<i>was</i>
VBG	<i>being</i>
VBI	<i>be</i> infinitive (<i>To be or not... It will be...</i>)
VBM	<i>am</i>
VBN	<i>been</i>
VBR	<i>are</i>
VBZ	<i>is</i>
VD0	<i>do</i> base form (finite)
VDD	<i>did</i>
VDG	<i>doing</i>
VDI	<i>do</i> infinitive (<i>I may do... To do...</i>)
VDN	<i>done</i>
VDZ	<i>does</i>
VH0	<i>have</i> base form (finite)
VHD	<i>had</i> (past tense)
VHG	<i>having</i>
VHI	<i>have</i> infinitive
VHN	<i>had</i> (past participle)
VHZ	<i>has</i>
VM	modal auxiliary (<i>can, will, would, etc.</i>)
VMK	modal catenative (<i>ought, used</i>)
VV0	base form of lexical verb (e.g. <i>give, work</i>)
VVD	past tense of lexical verb (e.g. <i>gave, worked</i>)
VVG	-ing participle of lexical verb (e.g. <i>giving, working</i>)
VVGK	-ing participle catenative (<i>going</i> in <i>be going to</i>)
VVI	infinitive (e.g. <i>to give... It will work...</i>)
VVN	past participle of lexical verb (e.g. <i>given, worked</i>)
VVNK	past participle catenative (e.g. <i>bound</i> in <i>be bound to</i>)
VVZ	-s form of lexical verb (e.g. <i>gives, works</i>)
XX	<i>not, n't</i>
ZZ1	singular letter of the alphabet (e.g. <i>A, b</i>)
ZZ2	plural letter of the alphabet (e.g. <i>A's, b's</i>)

As you should have been able to observe easily, the Penn tagset is much more compact, whereas the C7 tagset makes a lot of rather fine-grained distinctions. This makes it easier to search for highly specific grammatical phenomena, but also forces the user to have a high degree of awareness of which categories exist and what their exact meaning is. As we don't have enough space here to discuss all the differences in detail, I'll just point out a few exemplary ones, and briefly discuss their potential significance here.

Where the Penn tagset occasionally conflates categories – sometimes for reasons that aren't immediately obvious, such as, for example, to use the same tag for 'subordinating conjunctions' and prepositions – C7 goes exactly the opposite way, making much finer sub-distinctions. Even just for the conjunctions, it has 5 sub-categories (CS, CSA, CSN, CST, & CSW) that are applied in cases where the word forms themselves are grammatically polysemous, for instance using CST to mark *that* in its function as a conjunction (which, incidentally, subsumes relative pronouns), as opposed to that of a demonstrative determiner (DD1). Regarding prepositions, it distinguishes between general ones (II), and again specific forms that may be grammatically polysemous, namely *for* (IF, as opposed to CS), of (IO, as opposed to as adjective, in e.g. *matter of fact*, *out of date*, or adverb, in e.g. *of course*, *sort of*, in *multi-word units*), or without (IW, as opposed to as adverb, e.g. in *without giving away*).

Whichever tagset is more useful, though, really depends on the kind of analysis that is being conducted. The Penn tagset, for example, was specifically 'optimised' to be used as a basis for further syntactic analysis (see Santorini 1990), where many categories (as e.g. for nouns) can easily be 'lumped together' because they behave in a very similar manner from a syntactic point of view. If we're more interested in issues of 'semantic compatibility', i.e. collocability, however, then a tagset like C7 may provide us with more insights.

7.2 Tagging Your Own Data

Now, let's start taking a look at how you can make use of tagging in your own data, beginning by exploring how you can actually get your texts tagged in the first place. Unfortunately, most of the best-known taggers that have been developed over the years, such as, for example, the CLAWS system, for which a number of tagsets, such as the C7 discussed above, have been developed, are generally not freely available to the public or may require the purchase of a commercial licence for tagging suitable quantities of text. Furthermore, even though some tagger implementations are available as freeware/open source programs, these frequently come in the shape of programming modules or command-line based programs that many linguists are not accustomed to using, may necessitate additional software to be installed on your computer, require special configuration, or produce an output that's not easily adaptable to one's needs without much effort. In other

words, free taggers often lack simple, easy-to-use interfaces that make them accessible to the average researcher.

As a further exercise, let's download a freely available and easy-to-use PoS tagger, and see how we can tag our own data and post-edit it.

Exercise 35

Go to the Project Gutenberg website and download a copy of *The Adventures of Sherlock Holmes*.

As tagging longer texts may take a fairly long time, let's first prepare a relatively short sample. Make a copy of the file you just downloaded, calling it 'adv_of_sherlock_holmes_beginning.txt'. Tip: Whenever you make copies of files, always use the appropriate functionality inside your file manager.

Open the copy in your editor and place the cursor at the end of line 66, which should give us enough text to evaluate the tagging. Tip: In many editors, pressing 'Ctrl + g' will allow you to 'go to' a particular line.

Delete the rest of the text, using the time-saving methods we learnt earlier, and save the text.

Download my 'Simple PoS Tagger' from <http://martinweisser.org/tools/Tagger/Tagger.zip> and extract it to your computer/memory stick. As before, if you're using Mac OS X or Linux, you'll currently need to run it through Wine.

Run the program by (double-)clicking Tagger.exe.

Choose 'File→Load input file' from the menu or press 'Ctrl + f' on the keyboard, then find the file you just prepared. Select, and open it, so that it gets displayed in the window on the left-hand side.

Click on **Tag** or press 'Ctrl + t'. This will automatically tag your file and display the result in the right-hand window, as shown in Figure 7.1:

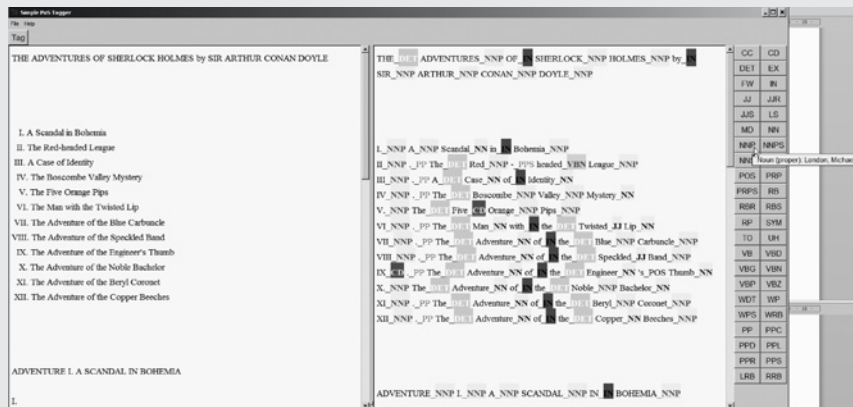


Figure 7.1 Sample output of the Simple PoS Tagger

Look through the text and see whether you can spot any problems with the tagging that may need correcting. As the tagset is similar to the Penn tagset we just looked at, you should already have a rough idea about what the tags mean, but if you're unsure, just find the relevant tag among the buttons on the right-hand side and hover your cursor over it in order to get a tooltip displayed.

Whenever you find a tagging error, correct it by selecting the erroneous tag and clicking the relevant button in order to insert the right tag. Also think about whether you can imagine why the tagger has made such an error.

Once you've corrected a number of errors, you can update the colour-coding display by pressing 'F5' to refresh the view.

Once you've finished, save the tagged file, either by choosing 'File→Tagged file→Save tagged file' or pressing 'Ctrl + s'. You can also save the file if you're unable to finish the corrections directly, and open it again in the interface later by choosing 'File→Tagged file→Open for editing' or pressing 'Ctrl + e'.

Another way to get a sizable amount of text tagged is to use the CLAWS trial service. This allows you to get a maximum of 100,000 words annotated in either the shorter C5 or the full-length C7 tagset. While this may seem a lot, just remember that a 150-page novel may already contain about 60,000 words, so this will definitely not be enough for larger files or projects...

Exercise 36

Open the CLAWS trial page at <http://ucrel.lancs.ac.uk/claws/trial.html>.

Open our Sherlock Holmes sample and copy and paste the contents into the text box on the page. Make sure the output style is set to 'Horizontal', and select the C7 tagset.

Run the tagging operation, and copy and paste the results into another text file in your editor.

Save the file and compare the results to the output produced by the Simple PoS Tagger, focussing on mainly higher-level category tag elements to ensure comparability.

Having now compared the results produced by two different taggers, you'll hopefully already have developed some basic sense of how reliable such programs in fact are – or rather, how unreliable they may be under certain circumstances. As this book is only of an introductory nature, though, I cannot really describe

all the reasons for this in proper detail, but, in the rest of the chapter, will try to present some of the causes for this in a highly, but hopefully not over-simplified, manner.

In the solutions to the exercises, I've already hinted at the fact that, sometimes, the taggers may not have had 'knowledge' of an appropriate rule or 'understood' the complexity of particular linguistic structures. Just as in the preceding sentence, I often used words that refer to meaning or understanding of linguistic rules in scare quotes, to indicate that, frequently, the notion of linguistic rules and knowledge in tagging may not really represent a proper kind of understanding that *is/can* be applied to morpho-syntactic annotation. What you may, perhaps, naïvely have assumed, without any prior knowledge of how modern taggers work, is that most of them use a lexicon to look up words and then apply some linguistic – in technical terms referred to as *symbolic* – morphological or syntactic rules in order to assign a PoS tag. This, however, is not true for many modern PoS taggers because they often primarily rely on lexica in combination with statistical rules of co-occurrence for fixed, and highly limited, sequences of words extracted from existing pre-annotated data. And, although the CLAWS tagger is exceptional here in the sense that it uses a hybrid approach combining probabilities and a substantial number of symbolic rules – which already allows it to perform somewhat better than the Simple PoS Tagger –, it's still limited by the fact that the probabilities do not reflect exceptions or longer sequences of words, such as complex NPs, well. Hence, although the results of modern taggers are fairly remarkable, with claimed accuracy rates generally between 95–98%, as we've seen from our short exercises, it's always advisable to check their output as thoroughly as possible before using it for any kind of analysis, especially when this is carried out on a more quantitative than qualitative basis. Furthermore, we also need to be aware of the fact that taggers do not always perform equally well on all types of texts or genres. Therefore, because most tagsets and taggers are generally designed for written language analysis, their adequacy tends to drop strongly when used with spoken data, partly because some of the PoS categories may not always be appropriate. We'll learn a little more about that issue when we practice annotating some spoken data in Chapter 11. However, even for written language, the accuracy of a tagger across different text types/genres may vary strongly. According to Paroubek (2007: 6, based on Illouz 2000), the performance of the TreeTagger (see below for more details) "varies from 85% to 98% with an average value of 94.6%" for the different categories of the Brown Corpus, which means that, in the worst case, 15 out of 100 tags may be wrongly assigned!

Now that you should have a basic understanding of morpho-syntactic tagging, we can soon learn how to exploit this feature in concordancing in order to improve our search results. Before we do so, though, we should probably still talk briefly about the availability of taggers for languages other than English. Whereas most of the early taggers developed for English were predominantly rule based, these days, most taggers contain fewer and fewer linguistic rules, but more and more probabilistic components instead. In other words, they generally use large lexica

to look up word classes and are trained on pre-annotated data in order to ‘learn about’ sequences of (likely) co-occurrences of PoS tags that may help to disambiguate grammatical polysemy. This makes it easier to design new taggers, and is at least part of the reason why many modern taggers can often handle multiple languages. A good example of a tagger that can do so, simply by adding so-called ‘parameter files’ for a language, is the *TreeTagger* (Schmid 1994), which is available for Bulgarian, Chinese, Dutch, English, Estonian, Finnish, French, Galician, German, Italian, Portuguese, Latin, Mongolian, Polish, Russian, Slovak, Spanish, and Swahili. Although claims of up to around 98% accuracy for such taggers are made, the main problems with them are that they are generally trained on relatively limited data, such as (for English) the *Penn Treebank* or freely available data from the *Wall Street Journal*, which are often not representative of the whole language, and that such probabilistic approaches do have strong limitations, as we’ve already seen in the solution to Exercise 35 and in our previous analyses of the tagged texts. Thus, their output always needs to be treated with caution, at least if a high degree of linguistic accuracy is required for one’s analyses. A further drawback is that the tagsets they tend to use are usually rather small, often derivatives of the Penn Treebank tagset, and may therefore not allow fine-grained grammatical/semantic distinctions, so that frequently they tend to be more suitable for language engineering tasks where a certain error rate is deemed acceptable in order to achieve a task.

Some languages, such as Chinese, are also more difficult to tag because recognising word boundaries is an issue, due to their not using spaces between words. We’ll discuss issues like this further in Chapter 9, but, for the moment, suffice it to say that the PoS tagging of such languages normally requires programs to artificially introduce spaces during the so-called *tokenisation* process. As a basic rule, morphologically (slightly) richer languages, such as German, tend to be somewhat easier to tag, because they often exhibit less grammatical polysemy due to *case-marking* or other features, such as the capitalisation of nouns in German, although of course such morphological features may also be ambiguous at times. Morphologically highly complex languages like Korean, for example, again present more issues because they tend to fuse a number of items related to grammar or honorifics into *agglutinated* word forms, a process that may also involve a certain amount of morphological adjustments.

Solutions to/Comments on the Exercises

Exercise 34

Most of the tags listed in the tagset should probably be relatively familiar to you, apart from maybe *UH*, interjection, which basically represents something like exclamations or expressions of surprise, such as maybe *wow*, *blimey*, *crikey*, *shoot*, etc., or *RP*, particle, which represents words you may be more familiar with as prepositions, but are generally used with so-called ‘phrasal or prepositional verbs’,

rather than NPs, as in, for example, *run off*, *give up*, etc. Another, possibly unfamiliar, category is *PDT*, predeterminer, which most grammars would probably categorise as *quantifier*.

What may be a little more surprising to you is to find separate categories for *EX* (existential *there*), *TO* (*to*), *POS* (possessive ending). This is basically because they may – or do – fulfil different functions from their adverb and preposition, or contracted BE-form ‘look-alikes’. In other words, *there* in an existential construction no longer functions as a locative adverb, *to* as an infinitive marker is nearly twice as frequent in the BNC than in its function as a preposition (16,470 vs. 9343; see Leech et al. 2001: 111), although, actually, the tagset treats them as equal here, and the ‘s-genitive is different from the contracted form of *is*. In contrast, you may have been surprised to find that the category *IN* subsumes both prepositions and ‘subordinating’ conjunctions (albeit the latter has always been a bit of a misnomer).

Some other categories that probably look relative unfamiliar to you, but are easy to understand when we think about the diversity that exists in general (written) corpora, are *SYM* (symbol), *FW* (foreign word), and *LS* (list item marker). The first refers to mathematical and other symbols that may (predominantly) occur in scientific or academic texts, the second can easily occur in humanities writing, newspaper reportage, or data that includes code-switching (such as the ICE corpora), while the final one can also occur in a large number of different types of written data, including online materials, where numbered or bulleted lists are frequently used to ‘summarise’ facts.

In terms of the punctuation marks, which are obviously not words but still need to be included, it’s interesting to note that all final punctuation is here subsumed under the full stop, and both colon and semi-colon are represented by a colon. While some punctuation labels are thus ‘conflated’, some others appear to be more fine-grained than necessary, for example, through making a distinction between straight double quotes, and opening and closing typographical ones, something that seems to be motivated by the nature of the data used when establishing the tagset. The same goes for the use of a hash mark (labelled ‘Pound sign’, as it’s an American tagset), which could normally have been subsumed under *SYM*, but must have occurred sufficiently frequently in the original data and represented an important enough feature to make the distinction necessary.

Another thing that’s interesting here is that, on the surface, no distinction is made between the right closing single quote and an apostrophe marking contractions. However, from some original data I obtained that was annotated with the tagset, it appears as if the apostrophe in such constructions (like *it’s*) is actually counted as part of the paradigm for *BE* and therefore rendered as *’s/VBZ*.

Exercise 35

The first thing you’ll probably notice in terms of tagging errors is that most of the Roman numerals, apart from, strangely, “IX”, have been tagged as proper nouns

(NNP), apparently because the tagger has mistaken them for initials. All of these should be changed to CD for ‘cardinal number’, and if they’re followed by PP (‘Punctuation, sentence ender’), the full stop and the tag deleted.

In the first title, “A” has been marked as a proper noun (NNP), while it should be a determiner (DET). In the second one, “Red_NNP -_PPS headed_VBN League_NNP” has apparently been interpreted as compound proper noun (NNP), which would – theoretically – be acceptable, but what is problematic here is that the hyphen in *red-headed* has been treated like a dash, consecutively marked as punctuation (PPS), rather than a hyphen, and hence the whole word has erroneously been split, with *headed* marked as a past participle (VBN), rather than as being part of a compound pre-modifying adjective. In general, probably either the whole expression should be tagged as a proper noun, or “Read-headed” as adjective (JJ) and “League” as a common noun (NN). Similar issues affect the rest of the titles, presumably due to the capitalisation of content words that ‘confused’ the tagger into believing that they were parts of names. As we’ve just seen, dealing with proper names may involve difficult decisions, even for human annotators, but the problem here still remains that some parts of what are potentially compound proper names are inconsistently marked as NNP, while others are marked as general nouns, etc. We’ll soon discuss why such issues may arise in tagging.

Moving slightly further down the text, we can see that, again, the determiner “A” in the first section/chapter title has, again, erroneously been tagged as NNP, but what’s more surprising is that the word “Scandal”, which was previously tagged as NN, is now tagged as NNP, too. Next, the cardinal Roman number “I.”, which was previously tagged as NNP, is now confusingly marked as PRP, that is, a pronoun, apparently because the tagger confused it with the first person singular personal pronoun. This should, again, be changed to CD. Overall, we’ve already seen that the tagger seems to have difficulties in making the right decisions when it encounters a number of words with initial capitals or that are completely capitalised, as may frequently occur in headings.

In the main text, things seem to be going better. The first real errors we encounter are that both *eclipses* and *predominates* are tagged as plural nouns (NNS), while they should be 3rd person singular present tense verb forms (VBZ), which indicates that the tagger seems to ‘know’ no rule that states that subject pronouns are generally not followed by nouns, but by verbs. Next, we find the combination “that_IN one_CD”, where *that* is tagged as a conjunction, when in fact it’s a demonstrative determiner. Here, the tagger has apparently ‘overlooked’ the fact that *that* is grammatically polysemous. Now, we may assume that *one* has been tagged correctly as a number, or numerical quantifier, to be more precise, if we simply go by form, rather than considering the function of word forms. However, as it’s not followed by a noun here, it actually constitutes the head of the noun phrase, and it would hence probably be better to tag it as a noun or pronoun. The next error we encounter is *abhorrent* being marked as a general noun (NN), when it’s really an adjective. The final issue we’ll discuss here is that *most*

in “the most perfect” has been identified as a superlative adjective, when in fact it modifies *perfect* and is therefore a superlative adverb and should have been tagged RBS. I’ll leave the identification and possible explanation of the rest of the errors to you here, but include a (hopefully) fully corrected version below:

THE_DET ADVENTURES_NNP OF_IN SHERLOCK_NNP HOLMES_NNP
by_IN SIR_NNP ARTHUR_NNP CONAN_NNP DOYLE_NNP

I_CD A_DET Scandal_NN in_IN Bohemia_NNP

II_CD The_DET Red-headed_JJ League_NNP

III_CD A_DET Case_NN of_IN Identity_NN

IV_CD The_DET Boscombe_NNP Valley_NNP Mystery_NN

V_CD The_DET Five_CD Orange_NNP Pips_NNP

VI_CD The_DET Man_NN with_IN the_DET Twisted_JJ Lip_NN

VII_CD The_DET Adventure_NN of_IN the_DET Blue_NNP Carbuncle_NNP

VIII_CD The_DET Adventure_NN of_IN the_DET Speckled_JJ Band_NNP

IX_CD The_DET Adventure_NN of_IN the_DET Engineer_NN ’s_POS
Thumb_NN

X_CD The_DET Adventure_NN of_IN the_DET Noble_NN Bachelor_NN

XI_CD The_DET Adventure_NN of_IN the_DET Beryl_NNP Coronet_NNP

XII_CD The_DET Adventure_NN of_IN the_DET Copper_NN Beeches_NNP

ADVENTURE_NN I_CD A_DET SCANDAL_NN IN_IN BOHEMIA_NNP

I_CD

To_TO Sherlock_NNP Holmes_NNP she_PRP is_VBZ always_RB THE_DET
woman_NN .PP

I_PRP have_VBP seldom_RB heard_VBN him_PRP mention_VBP her_PRP
under_IN any_DET

other_JJ name_NN .PP In_IN his_PRPS eyes_NNS she_PRP eclipses_VBZ
and_CC

predominates_VBZ the_DET whole_JJ of_IN her_PRPS sex_NN .PP It_PRP
was_VBD not_RB

that_IN he_PRP felt_VBD any_DET emotion_NN akin_JJ to_TO love_VB for_IN
Irene_NNP

Adler_NNP .PP All_DET emotions_NNS ,_PPC and_CC that_DET one_NN
particularly_RB ,_PPC were_VBD abhorrent_JJ to_TO his_PRPS cold_JJ ,_PPC
precise_JJ

but_CC admirably_RB balanced_JJ mind_NN .PP He_PRP was_VBD ,_PPC
I_PRP take_VBP

it_PRP ,_PPC the_DET most_RBS perfect_JJ reasoning_NN and_CC observ-
ing_VBG

machine_NN that_IN the_DET world_NN has_VBZ seen_VBN ,_PPC but_CC
as_IN a_DET

lover_NN he_PRP would_MD have_VB placed_VBN himself_PRP in_IN a_DET
false_JJ

position_NN .PP He_PRP never_RB spoke_VBD of_IN the_DET softer_JJR

passions_NNS ,_PPC save_IN with_IN a_DET gibe_NN and_CC a_DET
sneer_NN .PP

They_PRP were_VBD admirable_JJ things_NNS for_IN the_DET observer_NN
 -_PPS
 excellent_JJ for_IN drawing_VBG the_DET veil_NN from_IN men_NNS 's_POS
 motives_NNS
 and_CC actions_NNS ._PP But_CC for_IN the_DET trained_JJ reasoner_NN
 to_TO admit_VB
 such_DET intrusions_NNS into_IN his_PRPS own_JJ delicate_JJ and_CC
 finely_RB adjusted_JJ
 temperament_NN was_VBD to_TO introduce_VB a_DET distracting_JJ
 factor_NN which_WDT
 might_MD throw_VB a_DET doubt_NN upon_IN all_DET his_PRPS mental_JJ
 results_NNS ._PP Grit_NN in_IN a_DET sensitive_JJ instrument_NN ,_PPC
 or_CC a_DET
 crack_NN in_IN one_CD of_IN his_PRPS own_JJ high-power_JJ lenses_NNS
 ,_PPC would_MD
 not_RB be_VB more_RBR disturbing_JJ than_IN a_DET strong_JJ emotion_NN
 in_IN a_DET
 nature_NN such_IN as_IN his_PRPS ._PP And_CC yet_RB there_RB was_VBD
 but_CC one_CD
 woman_NN to_TO him_PRP ,_PPC and_CC that_DET woman_NN was_VBD
 the_DET late_JJ
 Irene_NNP Adler_NNP ,_PPC of_IN dubious_JJ and_CC questionable_JJ
 memory_NN ._PP

 I_PRP had_VBD seen_VBN little_JJ of_IN Holmes_NNP lately_RB ._PP My_PRPS
 marriage_NN had_VBD drifted_VBD us_PRP away_RB from_IN each_DET
 other_NN ._PP
 My_PRPS own_JJ complete_JJ happiness_NN ,_PPC and_CC the_DET home-
 centred_JJ
 interests_NNS which_WDT rise_VBP up_RB around_IN the_DET man_NN
 who_WP first_RB
 finds_VBZ himself_PRP master_NN of_IN his_PRPS own_JJ establishment_NN
 ,_PPC
 were_VBD sufficient_JJ to_TO absorb_VB all_DET my_PRPS attention_NN ,_PPC
 while_IN
 Holmes_NNP ,_PPC who_WP loathed_VBD every_DET form_NN of_IN
 society_NN with_IN
 his_PRPS whole_JJ Bohemian_JJ soul_NN ,_PPC remained_VBD in_IN our_PRPS
 lodgings_NNS in_IN Baker_NNP Street_NNP ,_PPC buried_VBN among_IN
 his_PRPS old_JJ
 books_NNS ,_PPC and_CC alternating_VBG from_IN week_NN to_TO week_NN
 between_IN
 cocaine_NN and_CC ambition_NN ,_PPC the_DET drowsiness_NN of_IN
 the_DET
 drug_NN ,_PPC and_CC the_DET fierce_JJ energy_NN of_IN his_PRPS own_JJ
 keen_JJ
 nature_NN ._PP He_PRP was_VBD still_RB ,_PPC as_RB ever_RB ,_PPC
 deeply_RB

attracted_VBN by_IN the_DET study_NN of_IN crime_NN ,_PPC and_CC
 occupied_VBD
 his_PRPS immense_JJ faculties_NNS and_CC extraordinary_JJ powers_NNS of_IN
 observation_NN in_IN following_VBG out_IN those_DET clues_NNS ,_PPC
 and_CC
 clearing_VBG up_IN those_DET mysteries_NNS which_WDT had_VBD
 been_VBN
 abandoned_VBN as_IN hopeless_JJ by_IN the_DET official_JJ police_NN ._PP
 From_IN
 time_NN to_TO time_NN I_PRP heard_VBD some_DET vague_JJ account_NN
 of_IN his_PRPS
 doings_NNS :_PPS of_IN his_PRPS summons_NN to_TO Odessa_NNP in_IN
 the_DET
 case_NN of_IN the_DET Trepoff_NNP murder_NN ,_PPC of_IN his_PRPS
 clearing_NN up_IN
 of_IN the_DET singular_JJ tragedy_NN of_IN the_DET Atkinson_NNP
 brothers_NNS at_IN
 Trincomalee_NNP ,_PPC and_CC finally_RB of_IN the_DET mission_NN
 which_WDT he_PRP
 had_VBD accomplished_VBN so_IN delicately_RB and_CC successfully_RB for_IN
 the_DET
 reigning_JJ family_NN of_IN Holland_NNP ._PP
 Beyond_IN these_DET signs_NNS of_IN his_PRPS activity_NN ,_PPC
 however_RB ,_PPC
 which_WDT I_PRP merely_RB shared_VBN with_IN all_DET the_DET
 readers_NNS of_IN
 the_DET daily_JJ press_NN ,_PPC I_PRP knew_VBD little_JJ of_IN my_PRPS
 former_JJ
 friend_NN and_CC companion_NN ._PP

Exercise 36

To do the comparison between the two results, it would be best if you either printed out both versions and then marked up whatever errors may exist or what either tagger may have done better than the other in colour, for example, red or orange for errors, and green for better performance. Alternatively, you could also use Notepad++ or another editor that allows you to display two files side by side and then examine the files in parallel. In Notepad++ you can achieve a split screen by opening both files and then clicking on one of the tabs and selecting ‘Move to other view’.

The CLAWS output is a little more difficult to read through because the (trial) tagger unfortunately didn’t preserve the original line breaks, apparently in an attempt to either end the line at a sentence end or to truncate it below 80 characters to ensure convenient on-screen display/formatting.

Concerning the actual tagging performance, the first thing you'll probably notice here is that CLAWS already performs somewhat better on the Roman numerals, although its input format guidelines (available on the web page) indicate that these should have been entered without a following dot to be recognised properly. Only 6 out of the 14 Roman numerals indicating section titles have been mis-tagged, mainly as singular proper nouns (NPI), but some also as singular general nouns (NN1), and "V." strangely as a general preposition (II), perhaps because the tagger 'assumed' that it may be an even further abbreviated form of *vs.*, hence 'contrasting' the two titles. The dots following the numerals are sometimes integrated with the numerals, but sometimes also tagged separately as sentence-final punctuation.

In the first title, both in the list of sections and in the heading for the first section, the initial determiner has again been mis-tagged, only this time not as a general noun, but a special form thereof, the name of a letter of the alphabet, that is, potentially part of a spelling sequence. Thus, both taggers had difficulties with this, but 'chose' to somehow identify the word as a noun form. In the second title/heading, CLAWS fares better at marking "Red-headed" as a compound adjective because it doesn't split at the hyphen.

CLAWS also correctly identifies the two word forms *eclipses* and *predominates* further down as verb forms, but makes the same mistake as the other tagger in mis-identifying *that* in "that one" as a conjunction (CST), rather than a determiner, despite that fact that it's identified "one" here as a pronoun (PNI). In the same sentence, we find the grammatically polysemous form "cold" having been incorrectly tagged as its noun representation (NN1) because it immediately follows the possessive pronoun "his" (APPGE). The reason for this error seems to be that CLAWS was unable to identify the remainder of the sentence as being the rest of a complex NP, presumably because it doesn't 'understand' comma-separated lists that well, and thus 'mis-took' the comma as a phrase boundary, in which case the annotation would have made perfect sense.

Unlike the other tagger, CLAWS correctly classifies *most* in "the most perfect" as a grading adverb (RGT), but is inconsistent in marking *reasoning* and *observing*, which form part of the complex compound noun *reasoning and observing machine*, because it marks the first one as a noun, while the second one is tagged as a verbal -ing form. As before, this seems to be due to its inability to recognise complex NPs. A little further down *save* in "save with a gibe" is marked as the base form of the verb (VV0), while in fact it's used as a conjunction meaning 'apart from', the conjunction "But" in "But for the trained observer" incorrectly identified as part of a multi-word preposition (II21 + II22), and "Grit" at the beginning of the next sentence again as a verbal base form – which, theoretically, ought to then cause the sentence to be an imperative, whereas it's in fact a noun. A little further on in the same sentence, we find "than" marked as a conjunction (CSN), when in fact it's part of a comparison and should thus be counted as a preposition. A similar thing happens with "but", marked CCB, in the next sentence, which, in

this case actually has the – perhaps slightly old-fashioned – meaning of ‘only’, and is therefore an adverb.

As before, for reasons of space, I won’t discuss the rest of the text here, but leave it up to you to identify any further potential issues...

Sources and Further Reading

- Cloren, Jan. (1999). Tagsets. In van Halteren, H. (Ed.). (1999). *Syntactic Wordclass Tagging*. Dordrecht: Kluwer Academic Publishers.
- DeRose, Stephen. (1988). Grammatical Category Disambiguation by Statistical Optimization. *Computational Linguistics* 14(1), pp. 31–39.
- Garside, Roger, Leech, Geoffrey, & McEnery, Anthony. (Eds.) (1997). *Corpus Annotation: Linguistic Information from Computer Text Corpora*. London: Longman.
- Garside, Roger & Smith, Nicholas. (1997). A Hybrid Grammatical Tagger: CLAWS4. In Garside, R., Leech, G., & McEnery, A. (Eds.) (1997). *Corpus Annotation: Linguistic Information from Computer Text Corpora*. London: Longman.
- Santorini, Beatrice. (1995). Part of Speech Tagging Guidelines for the Penn Treebank Project (3rd Revision, 2nd Printing).
- van Halteren, Hans. (Ed.). (1999). *Syntactic Wordclass Tagging*. Dordrecht: Kluwer Academic Publishers.

8

Using Online Interfaces to Query Mega Corpora

While smaller corpora may often not allow us to make as many generalisations about language as we might want to, modern mega corpora, such as the BNC, ANC, or COCA, offer remarkable possibilities for investigating general aspects of language and drawing more valid conclusions from them. As they're very costly and time-consuming to produce, though, one of their potential drawbacks is that their full versions may not always be available for free, and obtaining them may be prohibitively expensive for the individual (non-funded) researcher. Furthermore, even if they are obtainable, there may be a number of issues that make it difficult to handle them for the average corpus user. First of all, working with them, due to their sheer size, may require a relatively powerful computer with adequate memory and processing power. With increasing sophistication of PCs, this is turning into less and less of an issue. What may be more problematic, though, is that such mega corpora are often annotated in ways that make them difficult to use without dedicated software that has been written specifically for analysing them or displaying their contents in a way that's still easy enough to digest. Such software may, in the worst case, only run on very specific computer systems, or require complex installation procedures or changes to the system configuration that most linguists will be unable to carry out.

Luckily, these are problems that have already been overcome to some extent by the advent of web-based interfaces to these mega corpora, which, even if they may not allow us to do everything we might want to do with such a corpus, already provide many facilities for investigating the data in relatively complex ways that

will probably satisfy the needs of most researchers. In the following sections, we'll explore two of these web interfaces, one that allows us to work with the BNC for general British English, and the other with the COCA, for investigating features of American English. In doing so, we'll also make use of the knowledge you gained in the previous chapters for specifying linguistic patterns and working with PoS tags, in order to fine-tune our searches.

8.1 Searching the BNC with BNCweb

8.1.1 What is BNCweb?

BNCweb is a web interface to the BNC which is fairly straightforward to use. Essentially, being a concordance facility, too, some of its basic features are rather similar to the ones we've already discussed for AntConc. In order to be able to use BNCweb for free, you need to register at <http://bncweb.lancs.ac.uk/bncwebSignup/user/register.php>, providing a valid email address and other details. Once you've registered successfully, you can always access the site directly at <http://bncweb.lancs.ac.uk/>.

The following illustration shows the startup screen you'll see once you've logged in successfully. I'd suggest that you take a few seconds to try and familiarise yourself with this, and to see whether you can already understand what kind of functionality you may be able to expect from the interface.

The screenshot shows the BNCweb (CQP-Edition) interface. On the left is a 'Main menu' with various navigation options. The main content area is titled 'Standard Query' and contains a search form. The form includes a 'Query mode' dropdown set to 'Simple query (ignore case)', a 'Number of hits per page' dropdown set to '50', and a 'Restriction' dropdown set to 'None (search whole corpus)'. Below the form are 'Start Query' and 'Reset Query' buttons. A 'News' section at the bottom contains text about audio data access and a bug report link.

Figure 8.1 The BNCweb startup screen

As BNCweb in fact allows you to access the BNC not in its original form, but through a database interface, in technical terms, you don't run a *search* on the BNC, but formulate a *query*, as searches are known in database terminology. This is why you can see the word *query* appearing a number of times in Figure 8.1.

In the latest edition of BNCweb, there’s only one top-level option available for doing queries, the ‘Standard Query’. As a sub-type of this query option, there are also two highly useful menu options for querying in only spoken or written texts. In addition, these also provide the user with a convenient interface for selecting a number of different textual properties, as we’ll see later.

8.1.2 Basic standard queries

The standard query in BNCweb is similar to a combination of the basic and regex searches in AntConc and, in order to use it efficiently, you need to be aware of a few conventions. Before we start exploring these, though, let’s begin with a very basic search, just so you can get a feel for the new interface.

Exercise 37

Type the word *assume* into the query box and run the query by clicking on [Start Query](#).

As before, pay particular attention to issues of polysemy when you look at the results. We’ll later explore ways of disambiguating these, at least with respect to grammatical polysemy (though not for this example).

Your query "[word="assume"%c]" returned 4052 hits in 1491 different texts (98,313,429 words [4,048 texts]; frequency: 41.22 instances per million words)

No	Filename	Hits 1 to 50	Page 1 / 82
1	A01.407	Many people wrongly assume that all they have automatically goes to their loved ones.	
2	A04.170	The description is rather slender, but Pater was able to assume some existing knowledge on the part of his reader: 'We all know the face and hands of the figure, set in its marble chair, in that circle of fantastic rocks, as in some faint light under sea.'	
3	A05.1282	It was lively enough to marry Bellow to a 'stylish Radcliffe graduate' of whom Roth had been 'enamoured' — if we are to assume that The Facts has not imagined the connection.	
4	A06.1358	Here is an example of an impro exercise for two actors: 'An actor is asked to assume the character of a close family friend who arrives at the house with the news of the death of the wife's husband in an accident.	
5	A06.1413	Read newspapers, and don't assume that the whole world is as interested in acting as you are.	
6	A06.1547	Your first Equity card will be a provisional one; to become a full member you need to have worked for at least 30 weeks, not necessarily consecutively, Nevertheless, it's only realistic to assume that you may face longish periods of unemployment and to be prepared to deal with this constructively.	
7	A07.306	The meaning of democracy shifts even further once it is interpreted within the terms of the Calvinist principle of the Godly society, where it is the lot of the just to assume power and to guide the citizens in the paths of righteousness.	
8	A07.815	One explicit guiding principle of papal teaching which came to influence the Irish constitution and successive debates culminating in the church — state imbroglio of 1951, dealt with below, was that of 'subsidiarity': that no 'secondary' institution such as the state should take on duties which 'primary' and 'intermediate' ones, such as the family, could assume .	
9	A07.1278	From 1899, the intention of the clergy to assume and maintain absolute control of their schools led a number of them to refuse to appoint members of the Irish National Teachers' Organization so as to avoid trade union interference.	
10	A08.43	It is only during the time of waiting, wrote Harsnet, during the time without hope, that these things assume significance, only during the time without hope that one is conscious of them, that one remembers them with despair, that one anticipates them with dread.	

Figure 8.2 Results for simple search for *assume*

The default view we get here is not the familiar KWIC view, but something called ‘Sentence View’. This, however, can easily be ‘fixed’ by changing the default through adjusting the ‘User settings’ from the main page, or, temporarily, by clicking on [KWIC View](#).

Exercise 38

Experiment with the two different views by switching back and forth between [Sentence View](#) and [KWIC View](#).

Also try hovering with your mouse over the individual hits and the file-names on the left-hand side to see what happens.

8.1.3 Navigating through and exploring search results

At the very top, you can see a display showing your query string, the number of matches/hits, how many texts these occurred in, and some basic statistics. The ‘bar’ below it contains the navigation links that allow you to navigate through all the results pages, where |< means ‘go to the very first page’, << ‘one page back’, >> ‘one page forward’ and >| ‘go to last page’. As you can see, you can also ‘jump’ straight to a particular page if you remember where you’ve found specific results.

Exercise 39

Practise navigating through the pages of results, jumping to the next or previous page, or the very beginning or end of the set of results.

As you’ll have observed, the concordance output itself consists of the number of the hit, the name of the file it was found in – as a hyperlink – and the concordance result. As the order of the hits that are returned is exactly the same as the ordering in the BNC itself, simply looking at the hits that are returned may sometimes give a misleading picture, though, because the initial ones will all be from the general domain of informative written texts. In order to try and remedy this potential bias, you can click on the [Show in random order](#) button, which will ‘jumble up’ the results for you.

Exercise 40

Try this and see whether it changes your impression of the earlier results.

As the results are now in random order, if we do want to know which particular category of the corpus (or ‘genre’) the individual result was found in, we need to check the category details. This can be done by following the hyperlink behind the file name on the left, which will switch to a full, static, display of all the file details, rather than just a tooltip when you hover over the link. Tip: If you want

to explore the bibliographical details in more depth, I'd suggest you use a right mouse click to open the information in a new tab. This way, you just need to close the tab in order to return to the concordance results.

Exercise 41

Click on one or more of the links, and explore the exact nature of the information you can get on the file the search term was found in.

The dropdown list to the left of the **Go!** button allows you to access the set of options shown in Figure 8.3:

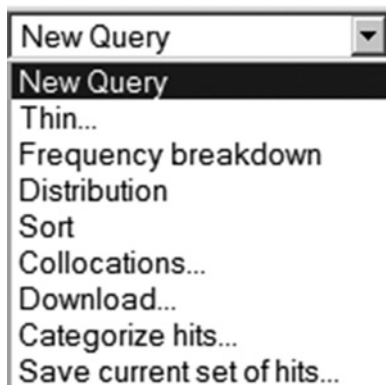


Figure 8.3 BNCweb query follow-on options

We'll explore some of these options in more detail in other sections, but, for the moment, we only want to take a look at the two basic ways of storing our results somewhere. These are hidden behind the two options 'Download...' and 'Save current set of hits...', where the latter may be somewhat misleading as the results won't actually be saved physically somewhere on your own computer/memory stick, but stored under your BNCweb user account where you can always retrieve them again. Thus, in most cases, if you want to save the results physically, perhaps in order to analyse them further using different programs, you'll probably prefer the 'Download...' option. Let's try both of these methods, so you can preserve the results of your search.

Exercise 42

Select 'Download...' from the dropdown list and click on **Go!**.

Take a look at the 'Output format options', try to understand them, and choose the ones you think are most appropriate for you.

For the moment, ignore all the category meta-information options listed in the section in the bottom half of the page.

Change the name of the output file to something sensible and click [Download!](#).

Once the file has been saved, open it in your editor to see what the output looks like.

Next, let's practise the other option for 'saving' our results.

Exercise 43

Return to the query by clicking on the back arrow in your browser (or pressing 'Alt + ←' on the keyboard in Windows/Linux, '⌘ + ←' on the Mac).

Choose the 'Save current set of hits...' option and click on [Go!](#).

Enter a suitable name for the query. If you want to choose a multi-word name, use underscores (_) instead of spaces, and click on [Go!](#) again.

Go back to the main BNCweb page. The easiest way is to keep 'New query' selected in the dropdown box and click on [Go!](#).

Click the 'Saved queries' link below 'User-specific functions' on the top left-hand side to verify that the query result has been saved correctly, then click on the link with the query name to see the original results re-displayed.

8.1.4 More advanced standard query options

The next thing we might want to be able to investigate in BNCweb is how to evaluate different options or alternative constructions. The mechanisms for this in BNCweb are sometimes misleadingly similar to the use of regular expressions we've learnt in Chapter 6, but the most basic forms employ a different system referred to as *wildcards*, whereas genuine regular expressions are in fact a feature of the CQP (Corpus Query Processor) syntax that BNCweb uses internally for its queries (without you necessarily noticing it), or that you can use to write more advanced and complex queries yourself.

8.1.5 Wildcards

Wildcards look like the basic quantifiers we're familiar with from regular expressions – i.e. ?, *, and + –, but they don't actually quantify the characters or character classes that precede them, and instead are comparable to the regex combination of a dot (any character) + quantifier, though not always the same. To make this

easier to understand, let's take a look at the different options, including some (non-exhaustive) examples of possible results:

- `?`: represents any single character, but *not* an optional one! Therefore, `yell??` finds `yellow`, `yelled`, `yeller`, etc., but not `yell` or `yells`.
- `*`: 0- (theoretically) 'unlimited', but only within single words! Hence, `call*` finds `call`, `calls`, `called`, `caller`, `calling`, `callous`, `calligraphy`, etc.
- `+`: 1- 'unlimited', again, only within single words! So, `call+` finds `calls`, `called`, `caller`, `calling`, `callous`, `calligraphy`, etc., but not `call`.

Now, let's put theory into practice and try this out on an example that illustrates spelling differences between British and American English.

Exercise 44

Type `colo*r` into the query box, think about which result(s) you would expect, and then run the query.

Rather than having to scroll through the list to find potential different variants, you can use a handy feature called 'Frequency breakdown', triggered from the dropdown menu on the top right-hand side, to explore this.

Select this and click on `Go!`.

The results probably aren't quite what you expected...

Exercise 45

Next, try replacing the `*` by a `?` and see what the result is, again using the 'Frequency breakdown' feature to see how this changes the results.

Unfortunately, the results may still not conform to our expectations, but we'll soon learn how to fix that ☺

The wildcard syntax in BNCweb also has a number of shorthand expressions that, again, sometimes look misleadingly similar to regex shorthands in that they have lower- and uppercase variants. However, while, as we've seen, in regexes lowercase refers to a 'regular' character class and using uppercase indicates the 'negation' of this character class, in BNCweb lowercase indicates the occurrence of single characters and uppercase of multiple (potentially unlimited) ones. For instance, `\w` finds a single word character, and `\W` multiple word characters, so that our

‘colour’ examples from above could also have been written `colo\wr` instead of `colo*r`, or rather `colo+r`, to be more precise, and `colo\wr` instead of `colo?r`. For further options, see Hoffmann et al. (2008: 105).

One other important thing about wildcards in BNCweb is that they can also represent word *slots*, if separated from other words in a phrase query. Thus, a `?` can stand for a single-character word token, punctuation or apostrophe, while `*` can stand for a word that may or may not occur, and `+` essentially means that a word has to occur in a slot.

8.1.6 Word and phrase alternation

BNCweb allows us to express alternation in a number of different ways. This can sometimes be bewildering to the beginner, so we’ll explore the options step-by-step. In the *Simple Query Syntax*, which is the one we want to use at this level, we have two different options that allow us to use wildcards in a more complex way, one form that is normally used to express alternatives within a word, and one that’s used for specifying phrases. The first one essentially looks very similar to the way in which we defined character classes in regular expressions. It uses the same form of square brackets to express alternative variants, only that, this time, they’re not restricted to single character only, and all alternatives in this form of notation have to be separated from each other by a comma. Therefore, to find the antonyms *thoughtful* and *thoughtless* at the same time, we could write `thought [ful, less]`. In addition, it’s also possible to specify nothing as an element within an alternation by writing a comma followed by nothing inside the brackets.

Exercise 46

Try expressing the different spelling variants now by using simple alternation.

Next, try to see whether you can use alternation to express all three different spelling variants of *ice cream*: *ice cream*, *ice-cream*, and *icecream*.

To be able to complete the above task, we need to employ *phrase alternation*, which is meant to allow us to specify searches for a number of words at a time, and looks rather like the kind of alternation we know from regular expressions. As this option is specifically designed to look for phrases, that is, combinations of words, it also allows us to specify spaces. As with regexes, we need to use round brackets and pipe symbols to delimit the alternatives, only that here, we cannot use the round brackets to group only part of the search term. Instead, each alternative has to be spelt out fully inside the brackets, as the alternative representation of our earlier example of antonyms, `(thoughtful | thoughtless)`.

Exercise 47

With this knowledge, try to now write a query that will find all the three variants, including the one with a space.

Once you've succeeded, take a closer look at the frequency breakdown.

The results should present a relatively clear indication as to which form of the compound is the most commonly used, and therefore probably also the one that should be taught to learners. Frequency breakdowns of this kind can often help us in making decisions as to which forms can be seen as more appropriate, and maybe in which context, too. They therefore represent a highly useful tool, both for pedagogical purposes as well as helping us personally to identify the right grammatical or stylistic choices, something that can equally well be used by advanced learners of a language to enhance their awareness of such choices, and native speakers who want to get around the limitations of a standard word-based thesaurus.

The following represents a brief summary of some of the linguistic points of interest the different wildcards and alternation features in BNCweb may be used to search for/investigate:

Table 8.1 Wildcards and their uses for investigating linguistic features

<i>wildcard/option</i>	<i>linguistic feature (within word)</i>	<i>linguistic feature (within slot)</i>
*	optional pre- and suffixes/ inflectional morphemes	word (optional)
+	(non-optional) pre- or suffixes/ inflectional morphemes	word (non-optional)
?	vowel alternation in strong verb paradigms	single-character word token, punctuation or apostrophe
[]	simple spelling variants	n/a
()	spelling variation in compounds & phrases	

One further feature you should be aware of involving the bracket notation is that, just like in proper regular expressions, the round brackets can be quantified to allow for optional or multiple element slots.

8.1.7 Restricting searches through PoS tags

As we've seen a few times previously when concordancing in AntConc, grammatical polysemy may 'interfere with' our search for a particular word form that can represent different parts-of-speech. As the BNC data available through BNCweb has been PoS-tagged using the C5 tagset, a tagset that is more complex than the Penn one, though not as much as its 'big brother' C7, this becomes much less of an issue. Now, we can simply restrict our searches to forms that belong to a

particular PoS category only, which then only leaves us with cases where there's basic semantic polysemy to contend with, which may or may not be a problem, depending on the particular goal we're pursuing.

However, a word of warning is in order before we start using this feature. As constructing the BNC was a major exercise involving the digitisation of very large amounts of text, sorting out meta-information as much as possible, and PoS tagging and annotating the data in a number of ways, the care taken in checking and correcting the final result of the tagging has, at least to some extent, been sub-optimal. Thus, we'll frequently find word forms that have either been tagged with two possible options for tags, even when, for the human annotator, only one would have been correct and possibly even easily identifiable, or many tags that are simply incorrect. This 'feature' does affect the reliability of the results that you can get out of BNCweb and other interfaces to some extent, the more so if you rely only on quantitative analyses of the results extracted. Therefore, when using such data, especially in the form of frequency lists, you should always be a little bit suspicious and try, as much as possible, to double-check the results to see if they may be affected in any way by this issue. Nevertheless, this now shouldn't discourage you from using the BNC, as it still represents a highly useful and amazing resource for investigating relatively recent British English in a form that is highly representative and will probably remain unmatched for a very long time.

After this brief excursion, let's return to investigating how we can make use of PoS tagging in BNCweb. The general notation that allows us to look for a combination of word form and tag here involves specifying a word form, followed by an underscore (`_`), followed by a PoS tag. As it turns out, the wildcard features we've just learnt about can also help us to a great extent in simplifying the specification of PoS patterns, so that we don't need to remember or specify all the different possible variant tags for a particular PoS category if, for instance, we want to look for all verb, noun, or adjective forms. Let's see how this will allow us to restrict our searches for a few grammatically polysemous word forms we've already encountered.

Exercise 48

To ensure that you don't get biased results, based on the order of the texts in the corpus, first go to the 'User settings' from the main query page and change the 'Default display order of concordances' to 'random order' and click on [Update Settings](#).

Run a search on the word form *mind*, initially without specifying a PoS tag.

Inspect the results visually by reading through them and trying to identify roughly what kind of a distribution in terms of nouns and verbs you may have in your random sample.

Next, to confirm your intuitions – and to verify the tagging –, hover over the hits to see which tag CLAWS assigned to them, and whether this is always unambiguous.

Select the ‘Frequency breakdown’ option from the options menu in the top right-hand corner and click **Go!**.

Next, select the option ‘Frequency breakdown of word and tag combinations’ and click **Go!** again. This will show you a breakdown of how often the word form occurs with a particular PoS tag, at the same time allowing you to see which tags it may occur with in the first place.

Investigate the different tag options first, then start a new query where you use a combination of word form, underscore, and a suitable wildcard for extracting all verb forms at the same time.

Do the same thing for *round*, first exploring the different word-class options through the breakdown, and then extracting only those that are adjectives.

To get a better sense of the query options and the C5 tagset, you can also take a look at – and possibly download – the ‘Simple Query Syntax help’, a PDF hyperlinked to the main query page. If you choose not to download it, I’d suggest that you open the link in a new tab and keep it open whenever you run anything but the most basic queries.

8.1.8 Headword and lemma queries

So far, we’ve learnt that we can specify different word forms/phrases via wildcards and/or tag restrictions. However, since it’s somewhat of a nuisance having to do this all the time, apart from being error-prone, it’s quite useful to be able to specify a search for all different forms of a *headword* or *lemma*. Here, the distinction between the two is essentially that the headword encompasses all the occurrences of a base form, regardless of PoS, while the lemma always represents a combination of base form + PoS tag (forms). Let’s explore the two different options, beginning with searches for the headwords *run* and *take*. To be able to do this, we simply need to enclose the particular base form in a set of paired curly brackets ({...}).

Exercise 49

Start by searching for the headword *run*.

Explore the results through a frequency breakdown (with tags), as we did in the previous exercise.

Repeat the same for the headword *take*.

To limit our searches to lemmas by specifying a PoS, BNCweb provides us with two separate options, both based on the following simplified (representation of a) tagset:

Table 8.2 Simplified tags in BNCweb

<i>Simplified Tag(s)</i>	<i>Coverage</i>
A or ADJ	adjectives
ADV	adverbs
ART	articles
CONJ	conjunctions
INT or INTERJ	interjections
N or SUBST	nouns
PREP	prepositions
PRON	pronouns
V or VERB	verbs
\$ or STOP	punctuation
UNC	unclear

To specify the lemma form, BNCweb allows two different variants. The first one requires both base form and simplified tag to be included inside the same set of curly brackets, but separated by a forward slash, e.g. {make/N} to find all occurrences of the noun *make*. The second one allows us to specify the headword and the simplified tag separately, both enclosed in curly brackets, but this time separated by an underscore, e.g. {make}_{N}. Essentially, the two options produce the same results, but I wanted to introduce the second option to you here because enclosing the simplified tag in curly brackets in this way also allows us to use it when we're not looking for lemmas, but for sequences of words where we may only want to specify the word class, rather than a word form + tag, and use a wildcard to find any word that occurs with this particular word class.

8.2 Exploring COCA through the BYU Web-Interface

Unlike the BNC, which is now freely available in its offline version, too, the COCA – at least in its free online version – can only be accessed through its interface at Brigham Young University. As there's also a database behind the interface, many of the operations that you can carry out in COCA resemble those in BNCweb. Unfortunately, however, the syntax for anything but the most basic word searches is fairly different, so that we need to learn new ways of doing old things, at least to some extent. Furthermore, the logic behind the interface, depicted in Figure 8.4, is also rather different in that it offers a wider initial choice of display options, groupings for results, or even comparison to other corpora available on the website, including a different form of access to the BNC.

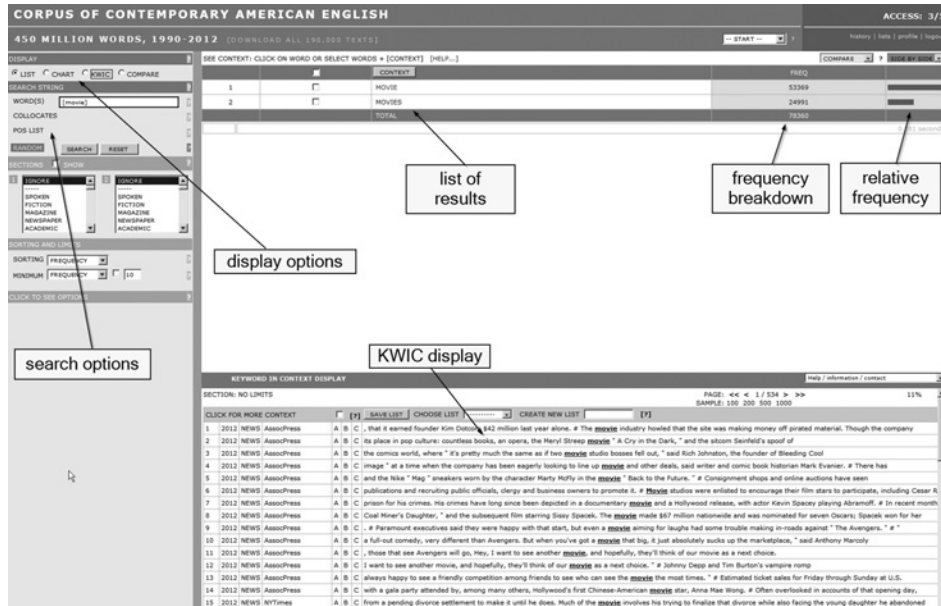


Figure 8.4 The basic COCA interface

In our discussions, apart from providing a general overview of the interface, we'll mainly try to focus on the most important differences that not only allow us to investigate American English here, but also conduct analyses of corpus data that weren't possible in BNCweb. Here, one of the biggest strengths of the COCA interface is probably that it was also designed for comparisons across different parts of the corpus, as well as to other corpora, from within the same interface.

As the COCA is a monitor corpus, you also need to bear in mind that the overall results you might get will probably change over time. Thus, for instance the number of hits I'll be reporting below, as well as their relative frequencies, may well change over time, due to changes in the language itself, and should only be considered accurate at the time of writing this book.

8.2.1 The basic syntax

As hinted at above, the syntax for basic word and phrase queries is somewhat similar to BNCweb in that typing in a word or a number of words separated by spaces and clicking on **SEARCH** will find instances of this word or phrase. If the display option is set to the default, LIST, something that looks quite similar to the frequency breakdown in BNCweb will appear in the top window on the right-hand side, which can be seen in Figure 8.4 for the lemma of the word *movie*. To the right of the listing for the word(s), the frequency of occurrence for each sub-item is shown, followed by a bar that indicates the percentage to which the hit

contributes to the overall search results. Unfortunately, though, there's no way to get the frequency breakdown for a word form in terms of different PoS categories.

Luckily for us, though, wildcards work in the same way as in BNCweb, so there's really nothing new to learn there, and we can now focus directly on identifying the options and differences to BNCweb in the query syntax. For instance, alternatives can be specified as list separated by forward slash (or pipe), but this time without any brackets around them. For example, *thoughtful/thoughtless* finds both antonyms at the same time, as well as providing a convenient frequency breakdown (see Figure 8.5).

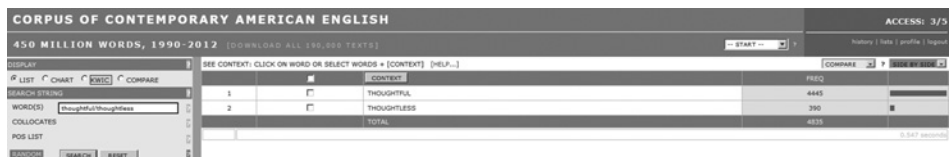


Figure 8.5 Display of antonyms *thoughtful* and *thoughtless* as alternatives

Here, it becomes fairly obvious simply by looking at the relative frequency bars that the positive member of the antonym pair is far more frequently used than the negative one.

As a general rule, any searches that go beyond simple words and phrases involve the use of (possibly multiple) square brackets (*[...]*) and allow the user to find PoS-tagged words, words followed by a particular PoS tag, or lemma forms (see Figure 8.4), where COCA here makes no terminological distinction between headword and lemma as in BNCweb. The consistent use of square brackets probably makes it easier to learn the syntax, whereas we've already seen that BNCweb forces us to remember different types of syntax for different purposes. The latter is especially true for the advanced query options we haven't been able to discuss earlier.

The following list represents all the basic word-level options, which can be 'picked and mixed' for phrase queries again. Please note that I sometimes change the slot for illustrating a given feature to either occur on the left or right if multiple slots are involved in a query. Plus symbols used below do not appear in queries, but simply indicate combinatorial options in a more abstract form:

- word form(s): finds exact words or phrases only
- word form(s) including wild cards: finds variable words or phrases
- *[base form]* finds lemma: e.g. *[mind]* finds *mind*, *minds*, *minded*, *mindings*, both as nouns and verbs (equivalent to headword search in BNCweb)
- word form + . + *[tag]*: e.g. *mind. [n*]* finds single word form *mind* only as a noun
- word form + space + *[tag]*: e.g. *go [r*]* finds single word form *go* followed by any adverb

- `-[tag]` + space + word form: e.g. `-[n*] water?` finds word form *water* or *waters* (including *watery*), preceded by anything that's not a noun, e.g. for excluding noun-noun compounds, such as *mineral water*, from searches for different types of *water* in count and non-count form
- `[base form]` + `.` + `[tag]` finds lemma of specific word class: e.g. `[mind] . [v*]` finds all word forms *mind*, *minds*, *mindings*, but only as verbs (equivalent to lemma search in BNCweb)
- `[=base form]` finds 'synonyms', e.g. `[=strong]` finds a number of interesting suggestions, although not all of these might correspond to our expectations for synonyms ☺

All tags (including in the BNC interface) are C7 tags and need to be written in lowercase, unlike BNCweb, where they're C5 tags, written in uppercase. This is basically everything you need to know about standard word and phrase queries in COCA, so let's see how you can apply this, first to a basic lemma query, and then to a comparison of American and British English.

8.2.2 Comparing corpora in the BYU interface

Exercise 50

Start by running a lemma query on the word *movie*, as depicted in Figure 8.4.

Investigate the results by first clicking on the hyperlink for the singular form, then doing the same for the plural, and last, but not least, by ticking the two boxes next to the separate forms and clicking on **CONTEXT** to get all results displayed at once.

The obvious advantages the COCA interface has over the BNCweb search results here are that a) we get a frequency breakdown of all forms immediately, rather than having to switch to a separate view first, and b) the results can be displayed in the bottom frame for individual sub-results immediately without having to open them in a new window or needing to navigate back a page each time we want to look at another sub-result. This is especially useful if we want to investigate lemmas with many different potential forms, as, for example, for verbs. The main disadvantage of the COCA interface, on the other hand, lies in the fact that it's not possible to get the frequency breakdown according to tags, which makes it more difficult to discern between polysemous forms.

In the next step, we want to see how we can compare this result to occurrences in British English, where the original word for what's called a *movie* in American English is generally assumed to be *film*, although some older people may also refer to films shown in the cinema as *pictures*. The latter basically comes from the same

expression as its American counterpart because the word *movie* is derived from the expression *moving picture(s)*, where the first component has simply been clipped in American English, while in British English, the second part of the compound noun was originally retained.

Exercise 51

In order to compare the results to the equivalent data from the BNC, click to open the dropdown list that initially reads SIDE BY SIDE. Here, it may be a bit misleading at first that you don't use COMPARE, but this would simply switch us to the other corpus.

Select BNC and wait for a few seconds for the result to appear.

The initial result of the side-by-side-comparison should look like Figure 8.6.

SEE CONTEXT: CLICK ON WORD (ALL SECTIONS) OR NUMBER (SPECIFIED SECTION) [HELP...]

COCA: 450,000,000 WORDS [COMPARE 2]						BNC: 100,000,000 WORDS [COMPARE 2] REMOVE COCA					
WORD/PHRASE	1: COCA	2: BNC	PM 1	PM 2	RATIO	WORD/PHRASE	1: BNC	1: COCA	PM 2	PM 1	RATIO
1 MOVIE	53369	1716	118.60	17.16	6.91	1 MOVIES	1013	24991	10.13	55.54	0.18
2 MOVIES	24991	1013	55.54	10.13	5.48	2 MOVIE	1716	53369	17.16	118.60	0.14

Figure 8.6 Side-by-side comparison for the lemma of *movie* in the COCA and BNC

These results show us a number of interesting things. First of all, in the online interface, what appears in light grey on the left-hand side in Figure 8.6 would be in green, which indicates that both sub-forms of the lemma are, somewhat unsurprisingly, far more frequent in COCA than in the BNC, while red or pink (here appearing in dark grey) indicates considerably lower frequency. Another thing we can see here is that the plural form is ranked more highly in the BNC half on the right-hand side, as apparently the ranking is based on the ratio of hits per million words. We'll talk more about frequency norming issues and what they might signify in a later part of the book. For the moment, all we need to know is that frequency norming simplifies the comparison between data sets of unequal size. For our analysis at the moment, the ordering on the right-hand side is actually irrelevant, anyway, and we can essentially get the information we wanted from the left-hand side, where the ratio tells us that the singular form is nearly 7 times, and the plural from 5½ times, as frequent in the COCA as in the BNC. This large difference already seems to indicate that the word *movie*, although it's certainly used in British English today, still doesn't seem to have replaced the word *film* yet. However, in order to verify whether this is actually true, and of course also whether *movie* is indeed the preferred form in American English, we really ought test and see how frequently the noun *film* is used in both varieties, so let's do this as our next exercise.

Exercise 52

Click on the dropdown list that initially reads ‘-- START --’ and select ‘COCA’. This will return you to the basic single-corpus COCA interface.

Run a search for the lemma of *film*, this time constraining the results to nouns, as, of course, *film* can also be a verb and we’re not interested in this.

Next, switch to the side-by-side comparison again and investigate the results.

As you’ve hopefully seen, this kind of comparison between two different varieties is quite easy to carry out in the BYU web interface, and you can also use your knowledge of the query syntax to investigate further varieties through the Corpus of Global Web-Based English (GloWbE), which is accessible through the same interface.

We’ll explore further analysis options in COCA in some of the following sections, but before we do so, I still want to mention a further display option COCA offers, the KWIC display, which can again be selected directly from within the display options in the left-hand search frame. Choosing this option will initially provide you with a selection of 100 random samples (adjustable to a maximum of 1,000) that are distinctively colour-coded for the main content word PoS categories, but unfortunately not for function word categories, which are marked with a single colour. The output can also be sorted using variable sorting options for a visual indication of the PoS of a search term(s), plus surrounding tags. This type of display may already be highly useful for identifying most of the potential for grammatical and semantic polysemy, but the maximum number may not allow you to extract enough samples in cases where a search term has a high degree of polysemy on both levels.

Solutions to/Comments on the Exercises

Exercise 37

As you’ll hopefully see from the examples on the first page, the verb *assume* here has two distinct meanings, one that represents a verb of cognition (similar to *think*), and the other a behavioural verb that may mean to ‘adopt a position or role’, either literally or metaphorically. The text displayed here just looks like any plain text from a book, article, etc., with only the hit highlighted and formatted as a hyperlink.

Exercise 38

Toggling the view to KWIC will provide you with a display that’s similar to that in AntConc, with the keyword centred in the window. When you hover over the

hyperlink for the hit itself, you should be able to get a tooltip that shows the hit and displayed context as morpho-syntactically annotated data where the hit is highlighted in red. Hovering over the file name on the left displays a different tooltip, this time providing fairly detailed information about the text the hit was found in.

Exercise 39

This exercise should present no difficulty, so there's nothing to discuss here ☺

Exercise 40

Because you'll now get random results, obviously there's no guarantee that you'll always be able to observe any difference. However, what may well happen is that some of the results illustrate the different meaning potentials more clearly, so, for instance if you were trying to find suitable examples for teaching or creating a textbook, repeatedly running the same query may be a useful option.

Exercise 41

As you'll have seen, the file information for every file in the BNC contains various types of information extracted from its header, partly referring to textual statistics, such as the length in words, and approximate sentence length, the full file name, as well as certain types of bibliographical information. However, what you may also – sadly – notice fairly frequently is that, although there's the basic provision for a possible piece of information in the left-hand column of the table displayed, often the information itself is actually marked as 'unknown' in the right-hand column.

Exercise 42

In choosing your options, you first need to make a decision as to whether you want any meta-information included, so you either need to have some of the options selected in the bottom half of the page or simply leave all un-ticked. I'd suggest that you initially remove all absolutely unnecessary information, such as meta-data or information required for re-importing into BNCweb, etc., so that you end up with a relatively simple list that primarily contains the hits with the query results marked, similar to the output we got from AntConc.

In order to get the most plain – but still useful – form of output, make sure that none of the boxes in the second half of the page are checked and also choose 'no' for 'Download both tagged and untagged version of your results'. In order to make it easier to see the hit, keep the 'Mark query result...' option set to 'yes', though. Under the options for choosing the operating system, make sure that you select something appropriate, as this'll affect the line endings in the output. If you're only viewing the result in a good text editor, this won't really make any difference, but if you may be planning to use other programs to further process your

results automatically, it may make a difference if these programs expect operating-system specific line endings, and thus potentially lead to errors. Keep the option ‘Write information about order...’ set to ‘no’ as well because it only describes the order of the basic output fields, that is, number of hit, file name, unit number where the hit was found, possibly speaker ID (for spoken language), left context, hit, and right context. The output should then look like the short example excerpt provided below:

- 1 A01 407 Many people wrongly <<< assume >>> that all they have automatically goes to their loved ones .
- 2 A04 170 The description is rather slender , but Pater was able to <<< assume >>> some existing knowledge on the part of his reader : [We all know the face and hands of the figure , set in its marble chair , in that circle of fantastic rocks , as in some faint light under sea .]
- 3 A05 1282 It was lively enough to marry Bellow to a [stylish Radcliffe graduate] of whom Roth had been [enamoured] -- if we are to <<< assume >>> that The Facts has not imagined the connection .
- 4 A06 1358 Here is an example of an impro exercise for two actors : [An actor is asked to <<< assume >>> the character of a close family friend who arrives at the house with the news of the death of the wife ’s husband in an accident .
- 5 A06 1413 Read newspapers , and do n’t <<< assume >>> that the whole world is as interested in acting as you are .

If you choose the option for downloading the tagged version, too, you get something that looks like this:

- 1 A01 407 Many people wrongly <<< assume >>> that all they have automatically goes to their loved ones . Many_DT0 people_NN0 wrongly_AV0 <<< assume_VVB >>> that_CJT all_DT0 they_PNP have_VHB automatically_AV0 goes_VVZ to_PRP their_DPS loved_AJ0 ones_NN2 ._PUN
- 2 A04 170 The description is rather slender , but Pater was able to <<< assume >>> some existing knowledge on the part of his reader : [We all know the face and hands of the figure , set in its marble chair , in that circle of fantastic rocks , as in some faint light under sea .] The_AT0 description_NN1 is_VBZ rather_AV0 slender_AJ0 ._PUN but_CJC Pater_NP0-NN1 was_VBD able_AJ0 to_TO0 <<< assume_VVI >>> some_DT0 existing_AJ0 knowledge_NN1 on_PRP the_AT0 part_NN1 of_PRF his_DPS reader_NN1 :_PUN [_PUQ We_PNP all_DT0 know_VVB the_AT0 face_NN1 and_CJC hands_NN2 of_PRF the_AT0 figure_NN1 ,_PUN set_VVN-VVD in_PRP-AVP its_DPS marble_NN1 chair_NN1 ,_PUN in_PRP that_DT0 circle_NN1 of_PRF fantastic_AJ0 rocks_NN2 ,_PUN as_CJS in_PRP some_DT0 faint_AJ0 light_NN1 under_PRP sea_NN1 ._PUN]_PUQ

Here, the output’s basically the same as before, only that the PoS tagged version follows immediately after the non-tagged output. This, however, creates a bit of redundancy, and it would be nice to be able to save the tagged version only, but

unfortunately, there's no provision to do so. Therefore, if you only want to retain the tagged part, you need to delete the non-tagged one manually.

Later on, depending on how much information you might need about a text, you can also output as much meta-information as required (or is actually available), in case you need to determine and/or distinguish which particular type of texts the results come from. When outputting meta-information, to keep it easily readable, I'd suggest you always go for the 'full values' option because otherwise you'll simply have to look up what each number means. This really only makes sense if you're planning to put the result into a relational database for complex analysis and annotation, and where you'll automatically be able to look up what the numbers mean from a lookup table.

To illustrate the difference, here are two short samples that only contain the first hit and all associated meta-information:

1	A01	407	Many people wrongly <<< assume >>> that all they have automatically goes to their loved ones .							
			1	5	64	3	2	5		
3	4	1	0	0	0	1	3	3	1	3
4	7	3	5	5	7	3	5	6	6	30

This first sample contains only the reference numbers, while the next one contains the full textual descriptions, where 'n/a' (not applicable) essentially indicates that those fields don't apply to written, but only spoken, texts.

1	A01	407	Many people wrongly <<< assume >>> that all they have automatically goes to their loved ones .								Written	Written	miscellaneous
			W:non_ac:medicine	1985-1993	Non-academic	prose and biography	Composite						
			Miscellaneous	published	Informative:	Social science	Low	unknown					
			unknown	unknown	Corporate	Adult	Mixed	Low	n/a	n/a			
			n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a			

Exercise 43

Essentially, doing this exercise should be straightforward. The only thing that may conceivably go wrong here is if you try to save the query with a name that contains spaces, in which case you'll get an error message saying: "Names for saved queries can only contain letters, numbers and the underscore character ("_")! Please use the "back"-button of your browser and change your input accordingly."

Exercise 44

The results of this exercise may initially be somewhat surprising to you because you may have expected to simply get the British and American spelling variants of this word. Instead, the query will have returned any word that starts with the grapheme combination/character sequence <colo>, followed by any number of characters, and ends in an <r>, because the wildcard asterisk (*) means 'zero or more characters'.

Exercise 45

Based on your knowledge of quantifiers, you might have thought that you could find the two spelling variants by replacing the `*` by a `?`, but unfortunately this still isn't how wildcards work. You may again be surprised by the result because you probably would have expected this to work, when in fact this query only returned examples of the British spelling. The reason for this is that the wildcard `?` always stands for any character and is thus the equivalent to the dot in regular expressions. Given our current means, it may therefore seem as if we can't really express what we want using wildcards because the `+` always represents at least one character, but potentially an unlimited number, too, so we clearly cannot use it in this case, either. However, luckily for us, there is a way to solve this problem, which is through expressing simple alternation.

Exercise 46

Using simple alternation in the query string `colo[u,]r`, you can now get the two spelling variants (only) displayed. Apart from the fact that you can even find 115 examples (1% of the total forms) of the American spelling in the British corpus data at all, the results should not be surprising now. You may now be tempted to investigate further why you can find these American spelling variants in the corpus, and of course the easiest option for this is to click on the link for 'colour' in the frequency breakdown to get to the concordance for the results, and then checking the meta-information for each result. Unfortunately, however, doing so remains relatively un-insightful, as in most cases you'll only find information about the author's domicile being marked as 'unknown', or, surprisingly, in some cases as 'UK and Ireland'.

Of course, one thing we haven't done yet here is to ensure that we really retrieve all forms of the noun paradigm by including the plurals, which we can do using the same mechanism and writing `colo[u,]r[s,]` instead. This, however, doesn't change the distribution, so the results, apart from now being complete, aren't really any more interesting.

Solving the second part of the exercise may again pose more difficulties than expected. When you try to achieve this task, you'll probably at best manage to create a query that will find two of the above forms, and instead get an error message saying "bracketing is not balanced". This is because the space is not part of the wildcard/simple alternation syntax, so we need to look for alternative ways of solving the problem.

Exercise 47

With a little bit of experimenting, you'll hopefully have come up with the solution (`icecream|ice-cream|ice cream`). This does indeed allow us to find all three alternatives at once, but we can still make it more compact by writing (`ice[-,]cream|ice cream`) instead, where the first part allows us to express

the hyphenated compound and the one completely written together as one word at the same time.

Exercise 48

As your results are randomised now, I can only discuss the solution in a more abstract way here, but hopefully, this'll reflect your results well.

First of all, when you do the visual inspection, you may notice that the random sampling could give you a little bit of a skewed impression because it might just so happen that BNCweb has only returned either a majority of noun or verb hits. However, even if this may be good enough for finding suitable examples to illustrate the usage of the word forms, we don't really get any idea as to whether it's used more frequently as one or the other, and it certainly doesn't allow us to easily extract only noun or verb usages. You may also notice that, occasionally, as I've pointed out before, CLAWS has assigned two tags to a word form (e.g. VVB-NN1), even if you might think that definitely just one of the two can be correct.

When you display the 'Frequency breakdown', the first result will not be very exciting because, as we've specified only one word form, it only displays that, together with the 'No. of occurrences', and the 'Percent' value, which will be 100%. What may be a little puzzling, though, is that the word form displayed is actually the one with an initial capital, which is perhaps not the most representative way of displaying it because this exact form is bound to be much rarer than the non-capitalised one.

Once you've specified the option for viewing according to 'Frequency breakdown of word and tag combinations', your display will show you the exact word forms in combinations with all the tags – or 'ambiguous' tag combinations – according to their frequency of occurrence. The results, which you can also download by clicking 'Download whole table', should look similar to Table 8.3:

Table 8.3 Word + PoS tags breakdown for *mind*

No.	Word and PoS-tags	No. of occurrences	Percent
1	mind_NN1	18270	67.5%
2	mind_VVI	3496	12.92%
3	mind_VVB	2551	9.42%
4	mind_NN1-VVB	2025	7.48%
5	Mind_VVB-NN1	724	2.67%
6	MIND_NP0	1	0%

As you can see, around 10% of all forms of *mind* have been tagged as ambiguous, something that wasn't as apparent when looking only at a small number of hits, and hopefully serves to reinforce my point about having to be careful when relying on the tagging in the BNC. Focussing on the ambiguous tags more closely, we

can also notice that sometimes the noun tag precedes the hyphen and sometimes the verb tag. This ordering indicates that CLAWS ‘assumed’ that the first one is the more likely of the two. What the breakdown also clearly shows us is that the noun usage of *mind* is predominant.

However, investigating frequency distributions wasn’t our main objective here, so let’s return to the main task at hand, exploring how we can actually make use of the tagging in our searches directly. First of all, when you take a close look at the listing of words + tags in the table, you’ll notice that they’re in fact hyperlinks that, once clicked, provide you with a concordance of exactly the combination specified, so that you can already narrow down your search in this way. This is still limited, though, because the different tags for verbs and nouns are further sub-divided in the output, so, in order to be able to find only verb forms, you need to use a wildcard query like `mind_v*`. This will then retrieve all hits where the tag starts with a verb marker as the first element, including those that have an ambiguous tag where the verb tag is listed first, but excluding the ones where the noun tag comes first. In order to also capture the latter, we need to make use of ‘word-level’ alternation in a slightly more complex form, specifying `mind_[V,N??-]*`.

Exercise 49

As you’ll be able to see when you run the search, you’ll immediately get a number of different forms of the verb and noun paradigms for the basic concordance. When you then switch to the frequency breakdown, you may be surprised a little because you not only get the regular forms of the two word categories displayed, but also two slightly more unusual forms, the archaic form *runmeth*, and *runnin’*, which represents the pronunciation variant with so-called ‘g-dropping’, where the final consonant is not realised as a velar, but an alveolar nasal. Now, you might expect that all these forms actually come from the spoken part of the BNC, but closer inspection of the meta-information reveals that they’re in fact from written materials, where authors have simply tried to represent a stigmatised pronunciation form. The last interesting ‘form’, or rather feature, we can observe here is that there are 5 instances of *run* with the UNC (unclear) tag. Closer examination by following the hyperlink reveals that these are in fact by no means ‘unclear’ examples, but in fact represent interrupted words that occur in the spoken part of the BNC. However, despite the fact that interrupted words are very common in spoken language, even that of highly fluent speakers, the CLAWS tagset provides no tag for this, something that is probably due to the CLAWS tagsets originally having been created for the morpho-syntactic annotation of written language, and later adjusted for spoken language to some extent.

The headword search for *take* doesn’t really have any surprises to offer. The distribution indicates very clearly that the noun form is comparatively rare, and that there’s a clear ranking in terms of frequency for the verb form, with the infinitive being the most frequent, followed by the simple past, the ED-form, and the ing-form. We again find one unclear form, but this time it doesn’t mark an incomplete

word, but instead a form that should probably be marked as an infinitive, despite the fact that it's missing the infinitive marker.

As before, for both headword searches, we again get a considerable number of ambiguous tags ☹

Exercise 50

Provided that you've used the correct syntax to search for the lemma, [movie], COCA should have found 53,369 instances of the singular form, and 24,991 of the plural. This exercise, on its own, doesn't really reveal anything remarkable. All we can really observe here is that, out of the overall 78,360 hits, the singular form accounts for slightly more than 2/3 of all results, while the plural is far less frequent. Rather than seeing the relative frequency in terms of percentages as in BNCweb, though, the bars on the right-hand side provide us with a visual indication of the extent to which each sub-form of the lemma contributes to the total.

Exercise 51

Other than what was already discussed in the chapter itself, you should have been able to observe that, in the BNC, the singular form is also more common than the plural, with 1,716 hits, and the plural only amounting to 1,013, so that the total is 2,729. In other words, the distribution of singular to plural forms is somewhat similar to that in the COCA, although the singular forms still make up less than two thirds.

Exercise 52

The correct version of the lemma query, which excludes anything but (common) noun forms, should be [film] . [nn*] . This should first yield 57,917 hits for the singular, and 17,421 hits for the plural in COCA (total 75,338). Once you open the comparison page, you should get 9,879 hits for the singular, and 3,189 for the plural in the BNC (total 13,068). Actually, the number of singulars should be adjusted for the BNC, as there's also one instance where *film* followed by a full stop has erroneously been marked separately because the punctuation mark was apparently not split off correctly during the tagging process, probably due to the fact that it was – equally erroneously – followed by a comma. This can easily be verified here by clicking on the raw frequency indication of the hit on the right-hand side, which is hyperlinked and will therefore display the hit in the bottom frame, as do all the entries for raw frequencies in the comparison table. Investigating the ratios of 1.30 and 1.21 essentially shows us that, unlike we assumed before, *film* isn't really only the more common British variant, but actually appears to be an alternative in American English that is almost equally common overall, but even more common in its singular form, which is a surprising result that clearly seems to contradict the stereotypical assumptions that Americans always 'talk about' *movies*,

while Brits always say *films*. The reason why I hedged the claim above is that, without more in-depth investigation of a large number of samples of the lemma, we shouldn't be making such a strong claim, as of course the word *film* doesn't only refer to 'moving pictures' but may also be used to refer to other concepts, such as the pre-digital medium for capturing/storing photos, or 'thin layers/coat of a substance' (e.g. *a film of ice*), etc.

Sources and Further Reading

- Anderson, Wendy & Corbett, John. (2009). *Exploring English with Online Corpora: An Introduction*. Basingstoke: Palgrave Macmillan.
- Davies, Mark. (2009). The 385+ Million Word *Corpus of Contemporary American English* (1990–2008+): Design Architecture, and Linguistic Insights. *International Journal of Corpus Linguistics*, 14(2).
- DeRose, Stephen. (1988). Grammatical Category Disambiguation by Statistical Optimization. *Computational Linguistics*, 14(1), pp. 31–39.
- Hoffmann, Sebastian, Evert, Stefan, Smith, Nicholas, Lee, David, & Berglund Prytz, Ylva. (2008). *Corpus Linguistics with BNCweb – A Practical Guide*. Frankfurt: Peter Lang.

9

Basic Frequency Analysis – or What Can (Single) Words Tell Us About Texts?

The methods described in this chapter can be considered a starting point for providing us with some quick hints as to which particular type of language, or perhaps even genre, we're dealing with in our corpus analysis by investigating how frequently certain words occur in a corpus. In other words, what we want to do here is to try and develop an understanding of *how much*, but perhaps also to some extent *how little*, lists of single words can tell us about the texts they occur in. This type of analysis will then be continued in the next chapter, where we'll discuss fixed or variable combinations of words that may be equally, or sometimes even more so, relevant to a particular type of text or genre.

In order to develop this understanding thoroughly, as so often in corpus linguistics we need to look at this task from at least two different angles, a theoretical and a practical one. We'll start with some theoretical considerations first, and then see whether or how this may affect the way we carry out frequency analyses, or need to interpret them.

9.1 Understanding Basic Units in Texts

Every text has multiple levels of meaning, and these levels tend to be – at least to some extent – linked to the 'physical', structural, units we may encounter, ranging from single 'words' via phrases, clauses, sentences, etc., to whole texts. There's frequently no direct one-to-one mapping, though, which means that we need to make certain decisions as to which sizes of chunks of texts may be relevant

to various types of analysis. In order to develop some more concrete notions of what the different potential units of meaning may be, and how they relate to the structural units we can investigate, we'll start with a bottom-up description of potential units, working our way up from the level of the 'word'. In later sections, we'll then move on to discussing longer sequences of units, both fixed, as in, for example, idioms or proverbs, and flexible, as in different types of more or less formulaic phrases.

9.1.1 What's a word?

One of the fundamental issues that we encounter when processing texts is the question of what exactly we should treat as a word. Initially, we may naïvely start out with the idea that words are entities in texts that are separated by white-space or punctuation. Unfortunately, this is even the case for the PoS tagging of the BNC, where the guidelines state that "our tagging practice in general follows the default assumption that an orthographic word (separated by spaces, with or without punctuation, from adjacent words) is the appropriate unit for word-class tagging" (Leech & Smith 2000). These guidelines, however, also recognise important exceptions, some of which will partly be discussed below, too, so they're not quite as narrow as the above quote may initially lead us to believe. Any attempt to primarily define words in this way, though, largely ignores the fact that what in practice functions as a single word need not only consist of a single entity delimited in this way.

In English, for example, *compounds* may be represented by a combination of 'words' that can be

- 1 *written together*, in which case our naïve definition would generally work, although it would technically exclude text-/paragraph-initial or -final words,
- 2 *hyphenated*, in which case our definition is likely to fail if we interpret the hyphen as a type of punctuation mark, or
- 3 represented as two textual units *with spaces between them*, where our naïve definition would most certainly fail.

We thus have three different (primary) means of creating a compound. As we've already seen when we examined phrase-level alternation in BNCweb, for the word *ice cream* we can find all three of these different variants within the BNC:

- 1 *icecream*: 28 matches in (17 texts),
- 2 *ice-cream*: 368 matches (174 texts),
- 3 *ice cream*: 471 matches (203 texts).

Based on this data, it's certainly the last variant that's the most frequent one, but also the one our naïve detection algorithm would fail most miserably on. We can find further similar problematic examples for other types of composite words

as well, such as for the word *however*, with *how-ever* occurring only once, *how ever* used as a true adversative conjunction occurring at least 6 times, and *however* – subsuming the true adversative and its ‘comparative’ use – 59,730 times. And even for the negated version of the word *smoker*, which may at first glance seem to be quite uncontroversial, we find the three variants *non smoker* (3x in 1 text), un-hyphenated *nonsmoker* (7x in 4 texts) and – most frequently – *non-smoker* (55 in 36 texts).

Therefore, the choice of how to represent composite words clearly isn’t fixed by a rule, but essentially seems to be a case of whatever form becomes conventionally more or less accepted by a majority of people in/for its written form. This is a particular problem we have in dealing with written language or, more generally, language that’s represented in some kind of orthographic form. In spoken language, in contrast, this issue doesn’t normally arise because a) words there aren’t generally separated by pauses – which the space is to some extent the equivalent of in English and other Western languages – and b) the prosody/stress patterns usually help us to recognise which units belong together or not, at least for relatively proficient speakers.

In other languages, such as Chinese, defining a word by the spaces surrounding it makes even less – or almost no – sense at all, as most sentences there do not even contain any spaces between characters. As a matter of fact, in the majority of cases, a Chinese word is made up of two characters, with the ‘exact’ ratio of characters to words apparently being around 1.7. In French, to cite an example from a language that’s closer in nature to English, noun compounds such as, for example, *machine à laver* (‘washing machine’) are generally formed from a noun + PP involving the prepositions *à* or *de*, while in German, extremely long compounds without any spaces can be created, although this is often exaggerated jokingly, as in the example of *Donaudampfschiffahrtskapitänswitwe* (‘widow of the captain of a steam boat that runs on the river Danube’).

Returning to English, further, but similar, problems are caused by other multi-word units (often abbreviated *MWUs*), such as phrasal (prepositional) verbs, for example, *give up/in* or *get along with*. Only that, in this case, we don’t need to deal with three different potential forms for one ‘word’ but a sequence of up to three orthographic units belonging together that act as one. The same goes for multi-word conjunctions, such as *as far as*, *as if*, *provided that*, etc. (c.f. Biber et al. 1999: 85–86), which are at least occasionally treated as one unit in so-called *ditto tags* (Cloren 1999: 45), for example, *As_CS31 far_CS32 as_CS33* in C7. Here, the whole MWU is tagged as a subordinating conjunction, with the first number indicating the total number of elements and the second the position within the sequence.

With contractions – such as *can’t*, *won’t*, *she’s*, *he’d*, etc. –, on the other hand, we really have the opposite problem, that is, we only have one single word form, but might in fact want to interpret this as two different words. In this case, if we purely look at individual (untagged) words and don’t actually analyse the

context, we won't necessarily be able to group the second part (i.e. the clitic) with its appropriate full counterpart in a frequency list (something we'll discuss in detail in Section 9.2). For instance, we wouldn't know whether a *d* following an apostrophe actually represents a form of *had* or *would* or an *s* stands for a third person singular form of *be* or a possessive marker, etc. If frequency lists are ordered alphabetically, the same issue doesn't necessarily arise because then the un-contracted and contracted 'first parts' will appear close together, but if they're sorted in terms of their frequencies, we may overlook that one form or the other could occur more frequently, as we'll soon test practically in AntConc.

In this context, problems with the semantic content of frequency lists also already become apparent to some extent, due to the polysemy of the little function word clitics indicated above. Of course, cases of polysemy aren't restricted to function words, something we've seen from the very beginning when we started exploring concordances, but may also occur with content words. Here, for instance, we have the well-known example of *bank* as a noun, either denoting a 'financial institution' or the 'sloping land/grounds alongside a river'. It may get even worse if we further add the potential for grammatical polysemy, such as in our example word *bank*, which may not only be a noun, but also a verb, where again it can have two or more different senses, i.e. 'turning steeply' for planes, 'having a bank account *with*' or 'counting/relying *on*', each time in conjunction with the highlighted preposition. And, of course, a simple listing of the single word form without context in a frequency list would (normally) not allow us to disambiguate our examples, as would for example be possible through sorting a concordance of the word by its left or right context.

All of the above are issues that are often largely neglected in the analysis of corpus data, especially in more automated and quantitatively oriented types of corpus analysis, where the norm still seems to be to assume that the traditional 'definition' of a word is relatively unproblematic, and that synonymous expressions generally consist of single words only. Now, while of course it's generally not possible for us to directly change the design of any corpus tools we may be using to allow us to deal with this issue, we at least ought to bear this 'handicap' in mind in many of our analyses, and see whether at least some of the tools allow us to avoid any of these problems, or whether we may be able to find a way to work around certain issues by manipulating our data ourselves in simple ways.

9.1.2 Types and tokens

The problems we discussed earlier in deciding what exactly constitutes a word all relate to the issue of how to assign a frequency count to a suitable representation form of a word, something slightly similar to creating the entry for a headword in a dictionary. Such a representative instance/word form in a frequency list is referred to as a *type*, and each individual occurrence of a particular type as a *token*, which is why splitting a text into individual word-level units is also referred to as

tokenisation. As we've already seen above, there are quite a few cases where we may have difficulty in assigning multiple words to one single type, but it isn't only for multi-word units that we encounter such problems. Even for single units that are clearly delimited by spaces or punctuation, we may end up having problems assigning them to one and the same type because of such issues as polysemy discussed above, but also alternative spellings (*colour* vs. *color*) due to dialectal or historical variants, typos (*teh* instead of *the*), or capitalisation/non-capitalisation.

For the latter, we can even distinguish two different types: capitals indicating 'sentence' beginnings or proper names vs. words that are completely in uppercase, such as in emphatic headings or attempts at representing increased loudness in spoken materials. Here, we may, for example, encounter the rendering of more loudly spoken/shouted passages in fiction, or even in spoken and orthographically transcribed corpora, such as the Hong Kong Spoken English Corpus (HK SCE; Cheng et al. 2008), where uppercase characters are used to indicate stressed syllables. One of the nastiest kinds of emphasis I've ever encountered is that some web page authors use uppercase letters that are each separated by a single space in order to highlight different sections of pages, for example, N E W S, where a normal tokenisation routine that works on whitespace would count each individual letter as a token of the letter type, rather than finding an occurrence of the whole word. Under normal circumstances, if we didn't realise that this feature/format were used in order to emphasise/highlight something, the impression this would probably create is that of someone spelling out the word *news* – where we'd actually want to count the spelt letters as tokens –, but which was clearly not the intention on these web pages.

In some cases, though, we may even deliberately want to force the grouping together of disparate forms. This may for example be the case if we want to group all forms of a certain paradigm together, such as all the forms of the suppletive verb *be* or all different forms (infinitive, third person singular present, present/past participle) of other verbs. This is referred to as *lemmatisation* (c.f. also Section 8.1.8, where we looked at lemma queries in BNCweb), and many programs that produce frequency lists offer this kind of facility. A similar thing can be done by expanding abbreviations to their full form, but with both lemmatisation and expansion, one always has to bear in mind that the individual forms may potentially also cause a change in meaning of the whole textual unit in which they occur, or may have been used deliberately as a (stylistic) feature of a particular type of genre. For instance, some publishers force their book or journal authors to use word sequences like *that is* instead of the more academic abbreviation *i.e.*, which has direct implications for the writing style and – to some extent – even the length of the article/book.

Now that you know about some of the issues that may affect our counting of words in a text/corpus, we can start looking at frequency lists properly. Basically, these lists can be created in two different ways by a program (or a person), either by producing a token list first, then sorting it and counting how many times a given word form occurs in a row, or, more efficiently, by keeping track of each

word form encountered and increasing its count as it recurs in the text. Both ways of course require the text to be suitably cleaned, normalised, and tokenised into words in the first place, which is at least part of the reason why I placed such a lot of emphasis on cleaning up our data in earlier sections of this book.

And of course, because frequency lists consist of a combination of word types and their associated frequencies, we also have multiple ways of sorting and displaying our results. The most useful output format in many areas of corpus linguistics for such a list is generally to have a list that is first sorted according to the frequency in descending order, that is, with the most common words occurring first, and then ‘alphabetically’ if the same number of tokens occurs for more than one type. However, we may also sometimes want to look at the least frequent words first, assuming that they can possibly tell us something very specific about the terminology used, for instance in a highly technical text that contains a number of specialised words. If we’re working in the area of lexicography, though, and are trying to create a comprehensive dictionary of a particular type of (sub-)language, we may well want to start with an alphabetically sorted list first, and then investigate the individual types according to their specific features that would allow us to classify and describe them optimally. To investigate this further, we’ll now look at what exactly frequency lists may look like, and what they could be useful/used for, from a more applied angle.

9.2 Word (Frequency) Lists in AntConc

We’ll begin our exploration into word (frequency) lists by creating a basic list of a small corpus in AntConc to see what such a list may look like, and also whether we can directly observe some of the problematic issues described above in our data.

Creating a word frequency list in AntConc is a very simple task. All you need to do to create a basic single-word list is load a corpus, select the ‘Word List’ tab, and click Start . The output of this tool consists of either three or four separate sub-windows, depending on which options you’ve chosen for it under the program’s ‘Tool Preferences’, and can be seen in Figure 9.1.

The first window from the left lists the rank of the word inside the frequency list, the second the frequency itself, the third the word form, and the fourth, if present, the lemma associated with the word form. The latter, however, is only shown if the option for this is activated in the ‘Tool Preferences’ for word lists, and a suitable lemma list loaded. Additional information indicated in Figure 9.1 is provided regarding how many different types overall have been found and how many tokens in total. The output can also be sorted in different ways, which we’ll try out later on in this section. For the moment, let’s just practise creating this basic list on a small corpus that consists of a selection of files I’ve compiled for you from the Trains (93) corpus. For more information on this particular corpus, see <http://www.cs.rochester.edu/research/cisd/resources/trains.html>.

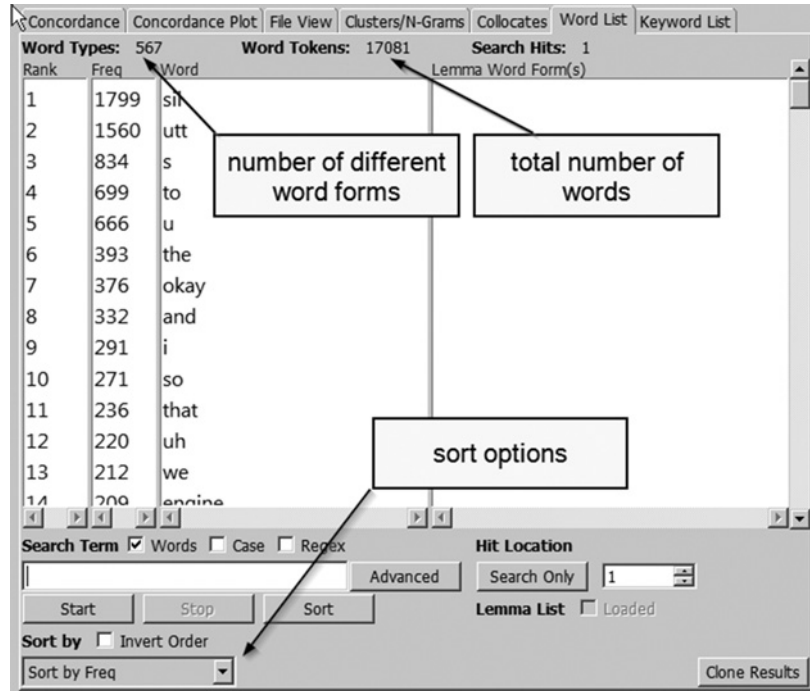


Figure 9.1 Output of a basic frequency list in AntConc

Exercise 53

Download the selection from http://martinweisser.org/pract_cl/data/trains93_selection.zip.

Extract it to the folder where you're keeping your downloaded data, retaining the original folder name, so that you can easily load the whole corpus.

Open AntConc and load the corpus.

As we're not interested in lemmas at the moment, turn off the 'Lemma Word Forms(s)' option under 'Tool Preferences→Word List'. Also turn off the option 'Treat all data as lowercase' in the same dialogue.

Switch to the 'Word List' tab or press **F6** and create the list, keeping the default set to 'Sort by Freq'.

Scroll through the list and try to understand what type of texts we may be dealing with here, both in terms of text type/category and domain/topic. Which particular words can help you to identify these and why?

If you're unsure what a particular 'word' entry in the list means, click on it and this will take you to a concordance of that entry. To get back to the original list, simply go back to the 'Word List' tab.

Keep the list open, as we want to explore further options using the same data later.

Exercise 53 was designed quite deliberately using this type of data in order to demonstrate the potential usefulness of word frequency lists for identifying distinctions between spoken and written language, as well as finding indicators for possible domains. However, be warned that if you create frequency lists of more general written data, it will probably be much more difficult to discern which words may be particularly indicative of certain genres or domains.

The high incidence of function words is of course something you'll be able to observe in both spoken and written texts. Perhaps the only types of texts/textual units where this is not so are the *telegram*, the *heading*, and the *bullet point*, where function words are deliberately left out in order to save space or to create a more 'immediate', summarising, effect by only including content words.

As pointed out in the general discussion on frequency lists earlier in this section, it's fairly difficult to define what exactly a word is, and we may at least partly have to accept simplified definitions or ones that are different from what we might find acceptable. This is especially true when working with software written by someone else, and possibly also for a particular purpose we may not even be aware of. Thus it's generally advisable to first check any output of a frequency list produced by some program to see whether it may exhibit any unusual features that could influence the analysis negatively. You should probably minimally verify how the program deals with contractions or hyphenations to be aware of what exactly the author's definition of a word is, as well as whether that particular definition fits your own purpose.

As will hopefully have become clear from the discussion of Exercise 53, the default frequency list in AntConc treats clitics, such as 's (but without the apostrophe) as separate words, which is often what we want because they're in fact abbreviated words that have been fused with a preceding word. This, however, makes contractions less easy to spot, let alone count, for the untrained observer, when they're still a very useful indication of spoken language, until such time as more people begin to realise that expanding and not using them in written language is actually counter-productive because it not only disrupts the reading flow to some extent but also creates the impression that at least some of the function words that would normally be de-emphasised as clitics may in fact be stressed, which is what using their full form indicates phonetically. Yet, in order to ascertain the 'spokenness' of a text/corpus until such time, we may often want to handle contractions as single units.

One of the great advantages of AntConc here is that it in fact allows you to (re-)define what exactly constitutes a word token for your own purposes by editing the definition of characters that are allowed to occur inside a word. Figure 9.2

shows the token re-definition options for including the apostrophe and the hyphen to allow contractions and hyphenated compounds to be counted as single words.

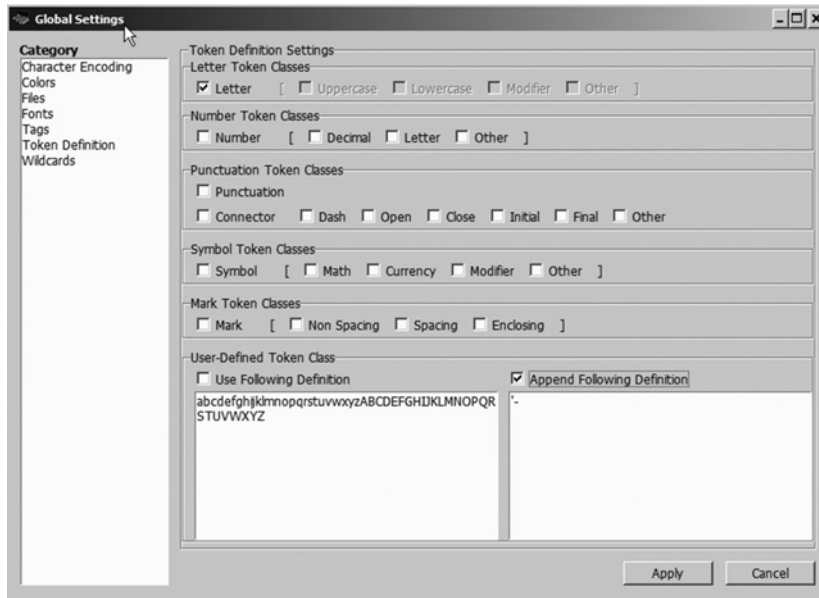


Figure 9.2 Token (word) (re-)definition in AntConc

Exercise 54

Select ‘Global Settings→Token Definition’.

Add the apostrophe and hyphen in the box below ‘Append Following Definition’ under ‘User-Defined Token Classes’, and tick the check box.

You may be tempted to simply tick the check box next to ‘Use Following Definition’, as the definition below this at first glance appears to include all the characters used in the English alphabet. This, however, may in fact exclude some of the rare accented words borrowed from other languages – although this is not applicable to our current data – and is therefore best avoided.

Don’t forget to apply the setting, too.

Now, simply click **[Start]** again to re-create the frequency list including the two extra characters, and observe the changes in the list by scrolling through it.

As before, keep the list open, so we can re-use the data.

An important corollary of being able to control the parameters for creating frequency lists is that, whenever you’re reporting any results of frequency analyses

in your research, you should always indicate what you're treating as a word and how you've controlled for this.

The above exercise should have demonstrated quite clearly what kinds of differences to our analyses changes in token definitions might make, but still cannot show us a full picture of all the advantages provided by creating customised frequency lists on the computer. In order to understand this better, we also need to take a look at the sorting options provided by AntConc. Unlike the sorting options we had for concordance lines, where we were able to sort according to *n* number of words to the left or the right quite freely, in this case, we have a more limited set of options, based on the options for combinations of output for types and frequencies, as already mentioned above.

Out of these options, we've already used one, sorting by frequency, when we created our first basic frequency lists. However, what you may not have noticed when looking through the results is that there's an implicit secondary sort order, which becomes relevant when we have the same number of types for different word forms. You can see this, for instance, when you take a look at the hits ranked numbers 28–30 in the 'Rank' window. All three of the words there – *at*, *boxcars*, and *oranges* – actually occur with the same frequency of 123 tokens, yet *at* 'ranks' at the top, while the other two are 'ranked' lower, despite having the same frequency. The reason for this is that, in order to be able to distinguish between the types, the secondary sort order also sorts them alphabetically. Now, as we learnt in Section 3.3.1, the computer normally distinguishes between upper- and lower-case characters by using different (ranges of) numbers to refer to them, where the uppercase characters normally always get listed first, so we should expect to find any instances where types have the same number of tokens, but differ in case, to exhibit this behaviour. However, if we scroll further down the list until we find ranks 112–116, which all have the same token frequency of 20, we find that *guess* is in fact listed above *OJ* because it starts with letter *g*. This is because AntConc already automatically corrects the computer's 'natural' sort order in order to allow us to see types that occur with the same letter together, something that's more natural for human 'consumers' of such frequency lists. If you do want to insist on seeing things the 'computer way', AntConc also allows you to do so by checking the option 'Treat case in sort' in the settings for the 'Word List' tool. Doing this will then sort the list 'asciibetically', that is, place *OJ* before *guess*. Let's explore the remaining options through another exercise.

Exercise 55

Return to the original frequency list sorted by frequency.

Under 'Sort by', switch the option from the dropdown list to 'Sort by Word' and click on .

Scroll through the resulting list and observe the effects to see what you can learn.

The remaining option under ‘Sort by’, ‘Sort by Word End’, is useful if you’re carrying out morphological analyses on corpus data, as it groups together words with the same endings, so that for instance plural forms of nouns or third person singular and other forms of verbs will end up being grouped together. You can test this, if you want, by applying this sort option, then typing *ing* into the search box, and clicking on **Search Only**. This’ll search through the word list for the first occurrence of the string *ing*, which, in most cases, will be followed by instances of what’s commonly referred to as a present participle, although a better, more neutral term for it would be *ing-form*. In some cases, though, the words below this will also be instances of deverbal *ing*-adjectives or simply nouns or other word forms ending in the grapheme sequence <ing>. As English these days exhibits relatively little inflection, using this feature may not appear very useful, but for other, more morphology-rich, languages this represents a highly useful way of investigating morphological regularities, as well as productivity.

Despite the fact that we’ve now explored all the options from the dropdown list, there’s still another possibility we’ve so far left unexplored. This is the ‘Invert Order’ option that appears in the form of a check box above the dropdown list. This allows you to perform reverse sorting for all the options in the dropdown list, that is, from z-a for alphabetical sorts, from 1 to the most frequent (n) types in frequency sorts, and z-a for endings, too, where the latter for instance sorts all negated and contracted forms together quite nicely.

Once you’re happy with the results of your frequency list, no matter which output format you’ve chosen, you can save the list to a text file again, just like you were able to do with the concordance output. This way, you can not only keep a record of it, but also analyse it further in a spreadsheet application, such as Microsoft Excel or OpenOffice Calc, or compare it to lists from other corpora, something we’ll also explore later on.

9.2.1 Stop words – good or bad?

Almost all texts, apart from maybe certain text types including telegrams and recipes, tend to have a rather high occurrence of high frequency function words, something we’ve just seen during our first explorations of frequency lists. Since these words don’t actually tell us much about the lexical richness or the content of a text/corpus, anyway, they’re often regarded as redundant and thus lists that exclude them, at least in theory, ought to help us develop better insights into the nature of any text or corpus under investigation. Words that contribute little to the semantics of a text are also referred to as *stop words*, and are often compiled into *stop word lists* that are excluded from frequency counts.

Table 9.1 shows the 15 most frequent word types occurring in section A (Press: Reportage) of the LOB corpus, together with their absolute and relative frequencies, rounded to two decimals. For convenience, I’ve also totalled up the frequencies.

Table 9.1 Top 15 most frequent word types in section A of the LOB Corpus

<i>Rank</i>	<i>Abs. Freq.</i>	<i>Rel. Freq.</i>	<i>Type</i>
1	5,863	6.44%	the
2	2,743	3.01%	of
3	2,212	2.43%	to
4	2,181	2.40%	and
5	2,149	2.40%	a
6	1,875	2.10%	in
7	925	1.02%	for
8	913	1.00%	is
9	888	0.98%	he
10	859	0.94%	was
11	843	0.93%	that
12	756	0.83%	on
13	688	0.76%	at
14	660	0.72%	's
15	643	0.71%	it
Total	24,198	26.67%	

In Table 9.1 you can easily see that all of the types listed belong to the category of function words, comprising determiners, prepositions, conjunctions, auxiliaries, and pronouns. The frequency list in general looks relatively similar to those AntConc produces for us, but unlike the ones there, this one has an extra column, labelled ‘Rel.(ative) Freq.(uency)’. What this column displays is the percentage to which any of the particular types contribute to the overall number of tokens in the corpus section. This information is more explicit and useful than the pure rank, even if the latter is accompanied by the raw frequency, as the raw frequency only tells us whether the contribution is larger than that of other types, but not to which extent this is actually significant regarding the data.

The top 15 types above already contribute to more than a quarter of all word types in section A. This is in line with Zipf’s (1949) observation that a small number of words generally account for the majority of tokens in language. And, as in fact none of the items in the list is a content word, the semantic information provided in Table 9.1 is essentially next to none. In other words, there’s absolutely no indication as to what the content of the section may be. To be able to simplify the exploration of semantically relevant vocabulary in the data, it thus appears justified to remove such function words from our frequency analyses.

However, there are certain problems in simply excluding specific types of function words from frequency counts. Whereas it’s relatively safe to exclude articles/determiners or pronouns from frequency lists, we already need to be somewhat more careful with the auxiliaries (*be/have*; modals), since some forms may actually represent full verbs, such as *have*, or even nouns, for example, *being*,

and, for example, question particles like *which* may well be relative pronouns that are important parts of the content. Abney (1996: 6) lists an example of an extremely atypical case of the noun phrase “The a are of I”, which at first glance seems to consist entirely of function words until one realises that *are* in this case actually refers to a measure of size (i.e. 100 m²), *I* to a quadrant on a map and *a* modifies *are* by designating a particular *are* that has been assigned the letter *a*.

Just as with auxiliaries and pronouns, we also ought to be very careful when eliminating prepositions and conjunctions from our frequency lists because they may equally tell us something about the domain or genre of a particular text. Imagine, for example, a text from the domain of finance about developments on the stock market, where certain values *rise above/to* or *fall below/down to* certain thresholds, etc., where the verbs on their own may not give us enough grounds to distinguish the type of domain, just like the verbs that form part of phrasal/prepositional verb combinations are often semantically relatively empty.

We can avoid at least some of these problems in using stop word lists by tagging our data grammatically before excluding any stop words, but there may not be such a simple solution in terms of deciding which of the semantically ambiguous types of potential stop words ought to be in- or excluded from our lists.

9.2.2 Defining and using stop words in AntConc

As we’ve learnt in our discussion of stop words above, the large number of function words that’s so typical of most spoken and written texts to some extent ‘obscures’ the content words that are deemed most relevant for the recognition of genres/domains. This is why most search engines on the Internet, for example, tend to have a list of these stop words that they exclude from their searches and possibly also the production of indexes that these searches are based on. We can do a similar thing in AntConc if we change the ‘Word List Preferences’ under ‘Tool Preferences’ to include a stop word list, as shown in Figure 9.3.

To use a stop word list, first tick ‘Use a stoplist below’. As you can see, AntConc then offers us two different ways of including stop words, one by specifying an existing file that includes the list, and the other by adding words to the list on an ad hoc basis by typing them in the box next to the label ‘Add Word’ and adding them to the list using the **Add** button. Unfortunately, when using the latter option, you have to type and add each word individually, so it’s often best to prepare a list beforehand and then add to this manually if you find that it doesn’t filter the list enough yet.

In my brief list shown in Figure 9.3, you can at least partly see that I’ve excluded the *utt* marker signalling each individual *utterance* in the Trains dialogues, *s* and *u*, the speaker identification codes, and a number of basic function words, mainly *determiners* and different forms of the auxiliaries *be* and *have*.

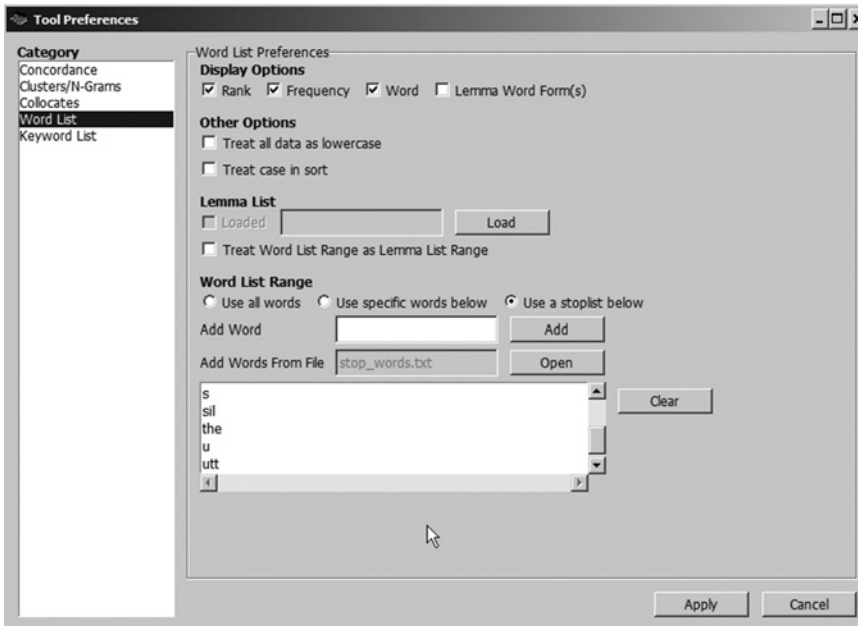


Figure 9.3 AntConc Word List preferences

Exercise 56

Experiment with this yourself by first specifying different function words you want to exclude from the word list manually.

Gradually widen the list of exclusions to encompass other words that aren't normally classified as function words. Each time you've added a few words, check to see whether your list has become 'more explicit'.

Once you're happy with the results, copy your list into a new text file and save it as `stop_words_trains.txt`.

Be warned, though, that sometimes excluding too many function words, such as possibly *prepositions* or *conjunctions*, may also skew your results because the very frequency of particular word classes may be highly indicative of particular genres/domains, as discussed previously, and illustrated by the prepositions *to* and *from* playing a rather important role in our dialogues. In cases where you think that the exclusion of specific (function) words may have such an effect, you can of course always create and load an alternative stop word list. As a matter of fact, you might want to create separate lists for many different types of data you work with.

From all the problems we've seen above, it may seem as if single word frequency lists are actually best avoided, but nevertheless, they may provide us with at least

some superficial information as to *lexical density* or makeup/type of a text/corpus. In *information retrieval*, a frequency list, if properly constructed and filtered, may also provide the basis for accessing indexes of search engines by ranking pages according to the frequencies of occurrence of individual or combined search terms. In *stylistics*, a word list created for the corpus of all writings of a particular author may also tell us something about their preferences for expressions, so that this may help us in deciding whether we should attribute a particular piece of writing whose origin is deemed debatable to this particular author or not, as is, for instance, done in the area of *stylometry*. And, of course, word frequency lists also provide the basis for many probabilistic approaches to language processing, such as establishing collocation measures or conducting probabilistic PoS tagging, some of which we've already discussed before, and others we'll turn to soon.

Of course, suitably constructed word lists aren't only useful for analysing individual texts or corpora from a purely linguistic point of view, but can also have other practical uses. In language teaching and learning, they can for instance be used by teachers to analyse materials and select appropriate or required vocabulary items, or by students to identify vocabulary they may need in order to cope with specific types of language materials, for instance in English for Academic or Specific Purposes (EAP/ESP). Attempts at creating such pre-fabricated word lists for EAP from corpus materials have already been made in the past. One particular example here would be the Academic Word List (AWL; Coxhead 2002), which appears to have gained widespread recognition as a resource for academic vocabulary training, despite the fact that it's heavily skewed towards the domains of law and economics, due to rather imbalanced sampling that over-emphasises these domains in comparison to the other domains of arts and science that are also included in the corpus used for its construction.

9.3 Word Lists in BNCweb

Unfortunately, the COCA interface doesn't allow us to create any frequency lists, so, out of the two online interfaces we've explored so far, we can only investigate how to do this with BNCweb.

9.3.1 Standard options

BNCweb allows you to create different types of frequency lists, either from the whole of the BNC or individual parts of it, by following the link to 'Frequency lists' in the 'Main menu' on the left-hand side of the browser window. Due to the fact that 90% of the BNC contains written materials and only 10% spoken ones, creating frequency lists of all words in the BNC, or even of individual words or word classes, rarely makes sense because this would simply give us a wrong impression of how frequent particular items may be in a heavily skewed selection

that isn't really representative of the whole language. This is why, in order to be able to make sensible use of frequency lists in BNCweb, we first need to learn how to restrict our selection(s).

To be able to limit frequency lists to either the spoken or written part of the BNC already makes far more sense because it allows us to understand these different categories of language better, and also compare frequencies across them. Creating such a categorised list is relatively straightforward, as can be seen in Figure 9.4, which depicts the general choices for frequency lists, where I've chosen to produce a list for 'Spoken Texts only' on the left-hand side.

Word frequencies	Headword or lemma frequencies
Choose one or several POS-tags: no restrictions ▲ any verb any noun any adjective any adverb any article any preposition ▼	Choose one or more lemma classes: no restrictions ▲ ADJ ADV ART CONJ INTERJ PREP ▼
Word pattern: starting with []	Word pattern: starting with []
Range of texts: Spoken Texts only ▼	Range of texts: Whole BNC ▼
Range of frequency (optional): from [] to []	Range of frequency (optional): from [] to []
Show individual tag frequencies: no ▼	Show individual tag frequencies: no ▼
Type of ordering: descending ▼	Type of ordering: descending ▼
Number of items shown per page: 50 ▼	Number of items shown per page: 50 ▼
Show list Reset	Show list Reset

Figure 9.4 BNCweb frequency list selection options

In the same way, you can just as easily select 'Written Texts only', but there are also many other options that allow us to restrict our selections. These include creating lists from *user-defined subcorpora*, something we'll explore soon. While the left-hand side contains options for basic (word) frequency lists, the right-hand one does the same for headword/lemma lists.

Most of the options here should be more or less self-explanatory, predominantly related to restricting or sorting the output in ways already discussed. One of the most useful features in this setup, though, is probably that you're able to restrict the choices in different ways that can also be combined with one another. This will, for instance, allow you to investigate the frequencies of individual word classes, possibly in combination with different patterns (specified under 'Word pattern'), such as pre- or suffixes, etc., or even to state how many times such a word must occur minimally. Such features may for instance be useful in cases where you're investigating issues of *productivity* in *word-formation* through *affixation*, or if you want to create word lists from particular frequency bands for graded vocabulary

acquisition. Let's experiment with this part of the interface a little, so you can get a better feel for what it may help you to achieve.

Exercise 57

Try out the different individual options first to get a feel for the results.

Each time, step through at least a few of the results pages by clicking on >>, and see whether you can make any interesting observations.

Next, try to combine some of the options, and see how this changes the results.

As the options for headwords/lemmas are essentially the same, they should not require any further discussion here, and I'll leave it up to you to explore them. And, before we move on to the next section, it's perhaps also worth mentioning here that different types of frequency lists can also be downloaded directly from <http://ucrel.lancs.ac.uk/bncfreq/>, the website that accompanies Leech, Rayson & Wilson (2001).

9.3.2 Investigating subcorpora

One special feature of BNCweb that potentially allows you to compare your own corpus data with that of the BNC is the ability to create subcorpora, based on different selection criteria. Figure 9.5 shows the different top-level options available for this:



Figure 9.5 Options for defining subcorpora in BNCweb

This dropdown list can be accessed via the 'Make/edit subcorpora' option from the main page. Out of these options, we'll only explore the first four because the remaining ones are either too special for our current purposes or may not yield appropriate results quickly enough. The most extensive set of choices is offered by the two options for using meta-textual categories, either for written or spoken texts. These are based on different types of information stored inside the headers for the individual files for the BNC, and are too numerous to depict in an illustration, so we'll explore them again as an exercise.

Exercise 58

Select ‘Spoken meta-textual categories’ from the dropdown list and click on **Go!**.

First explore all the options, and try to develop an understanding of what they might mean.

Once you’ve finished exploring, select ‘Dialogue’ from ‘Interaction Type’, and ‘Leisure’ from the ‘Domain’ options, respectively.

Click on **Get text IDs/Speaker IDs**.

Explore the list of texts and then tick the option for ‘include all’ in the top right-hand corner.

Make sure the option for ‘New subcorpus’ is still selected in the dropdown list and click on **Add**.

On the following page, name the subcorpus ‘dialogues_leisure’ and click on ‘Submit name’. You should then get a page confirming that the new subcorpus has been created, including information on the number of texts and words it contains.

Return to the main page by clicking on **Go!**, then click the link to the ‘Frequency lists’ page.

From the ‘Range of texts’ dropdown list, select the subcorpus we just added and create a frequency list.

Select ‘Download whole FrequencyList’ from the options, click on **Go!** and save the list as `bnc_dialogues_leisure_frequency_list.txt` to your results folder.

When I first started writing this book, downloaded lists based on user-defined subcorpora in BNCweb, unlike those based on pre-defined sub-parts of the BNC, were not created with the sorting option we’d like to have by default, which is according to frequency in descending numerical order. This initially made such lists rather less than useful for our purposes, so in order to ‘fix’ this problem, I created the next two exercises. And even though the issue has now been sorted out in the interface itself, I decided to keep them, as they allow you to learn how you can import lists into a spreadsheet application, such as MS Excel or OpenOffice Calc, and then re-sort the data in order to achieve a similar flexibility to that we have in AntConc.

The procedures described here, along with the screenshots, are based on Excel 2010, and different versions of Excel, or different spreadsheet applications like Calc, may provide other options which unfortunately cannot all be covered here. However, the basic logic behind importing the list into a spreadsheet application and sorting it will always be the same, and should also prove useful for other analysis purposes, such as, for example, comparing frequency lists from different corpora, as we will see in Section 9.5.

Exercise 59

Start Excel (or Calc) and click on ‘File→Open’ (or press ‘Ctrl + o’).

Select the appropriate file type that allows you to import text, generally *.txt and/or *.csv. The latter extension stands for ‘comma-separated values’, but usually also covers tab-delimited data. In Excel, the option should read ‘Text Files (*.prn;*.txt;*.csv)’ and in Calc ‘Text CSV (*.csv;*.txt)’.

Select the frequency list you just saved and click **Open**.

Excel’s (or Calc’s) Text Import Wizard will start and display the dialogue shown in Figure 9.6:

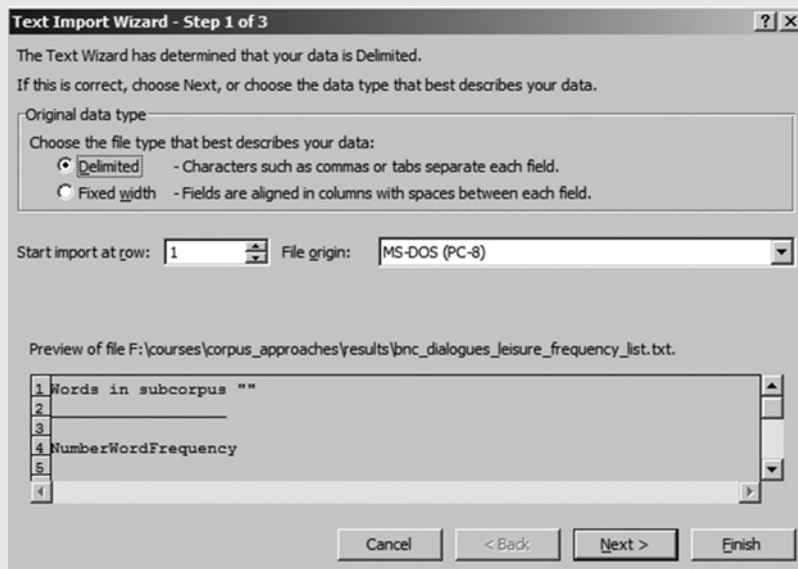


Figure 9.6 Excel text import wizard (stage 1)

In Excel, next change the option for ‘Start import at row’ to 4 in order to skip the first three lines, and click **Next >** twice, simply accepting the options for stage 2. In Calc, the dialogue looks very similar here, but has fewer steps, and you should select ‘Tab’, ‘Space’ and ‘Merge delimiters’ under ‘Separated by’. You also need to select the second column (the one that shows ‘Word’ in the fourth row) and change the ‘Column type’ to ‘Text’. Clicking **OK** will then complete the import.

In the next Excel dialogue, first click on the ‘Word’ field in the ‘Data preview’ box (see Figure 9.7), then select ‘Text’ from ‘Column data format’. This step is necessary because there may be some special characters in the data that could otherwise be re-interpreted as mathematical operators by the spreadsheet application.

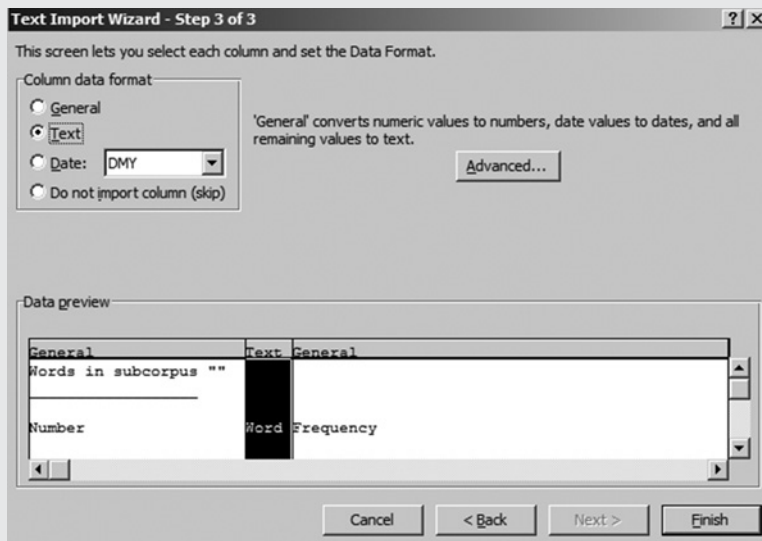


Figure 9.7 Excel text import wizard (stage 3)

Click **Finish** and **OK** to accept the suggested output location.

We now have an empty row between our header row and the data, so delete this by first clicking on the '2' indicating the second row, then using a right mouse click and choosing 'Delete'.

Next, check to see whether all fields have been split correctly by scrolling through the whole list and fixing any potential errors.

Once you're happy with the results, save the file, ideally using the same file name you used for the text file, apart from the extension.

We now have our frequency list stored in a very convenient format, as spreadsheets not only allow us to re-sort our data easily (and repeatedly, if necessary), but also because this makes it possible to investigate and enrich the data in various ways. Lists in spreadsheet format can, for instance, be filtered quite easily, cells colour-coded to indicate interesting features, or even comments or new category labels added to classify different types of words. If you want to, you can also cut less useful entries, such as maybe those pertaining to numbers or stopwords, from the list and paste them to another if you're not sure whether you might need them again later. Unfortunately, we don't have space to discuss all of the above options here, so I suggest you find suitable (online) resources about working with spreadsheets and/or experiment with such options yourself. For now, we'll focus on how to sort the list in order to bring it into a more suitable format for our purposes.

Exercise 60

Select all data on the spreadsheet. The easiest way to do this is to click in cell A1 (the one that reads ‘Number’), then press ‘Ctrl + Shift + →’, followed by ‘Ctrl + Shift + ↓’, which should highlight all consecutive cells that contain frequency data, but of course you can also use the mouse if you want to.

Activate the ‘Data’ tab in Excel, then click on ‘Sort’, and set the options as depicted in Figure 9.8. To achieve the secondary sort, you need to add a level. In Calc, you need to click on the ‘Data’ menu and select ‘Sort...’ from there to get custom options and select similar options to the ones displayed for Excel below.

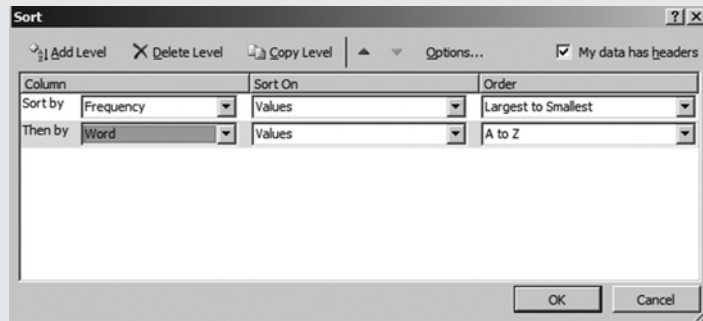


Figure 9.8 Sort options in Excel

Click OK.

Save the list again, and look through it to see whether you can identify some interesting points.

As you’ll hopefully have seen from the previous two exercises, there are some very distinct similarities to the earlier frequency list of spoken language data we created from the Trains corpus selection. Furthermore, this will now also have given you an opportunity to, at least to some extent, understand in which way(s) spreadsheet applications allow you to work with frequency data. In addition to being able to import tab- or comma-delimited data, spreadsheet applications usually also allow you to export your results to such plain-text formats. This will obviously incur a loss of any special formatting, such as colour-coding, that you may have applied to your data, but, on the other hand, makes it easier to exchange the data you’ve already processed with other people who may not have any spreadsheet applications installed on their systems, but who will still be able to view the data in a basic editor. For even more complex analysis and annotation options,

such data can also, in much the same way as we just discussed, be imported into databases. However, how to go on from there is a much more advanced topic we cannot discuss here. We'll explore some further options related to using spreadsheets in later sections, but for now let's move on to investigating other ways of creating subcorpora and associated frequency lists in BNCweb.

Going through all the meta-textual categories on the restrictions pages for spoken and written language may sometimes involve making a lot of decisions and also doesn't allow you to select mixed data from both media. This is why, for simple domain-dependent tasks, it may occasionally be easier to use either pre-defined categories or make a fine-tuned selection from the 'Genre labels' page accessed through the 'Make/edit subcorpora' options list.

If you look at Figure 9.9, you'll see that there are two options for making selections, a single dropdown list with pre-defined choices in the top half, and three selection boxes in the bottom half. The dropdown list consists of hierarchically structured entries that have either two or three parts. Rather than going through a whole set of options, you can simply pick from a few pre-defined categories/domains here, although, of course, the options for choosing and combining different meta-textual criteria in the bottom half give you far greater control over the selection process, as the explanation below the three boxes indicates.

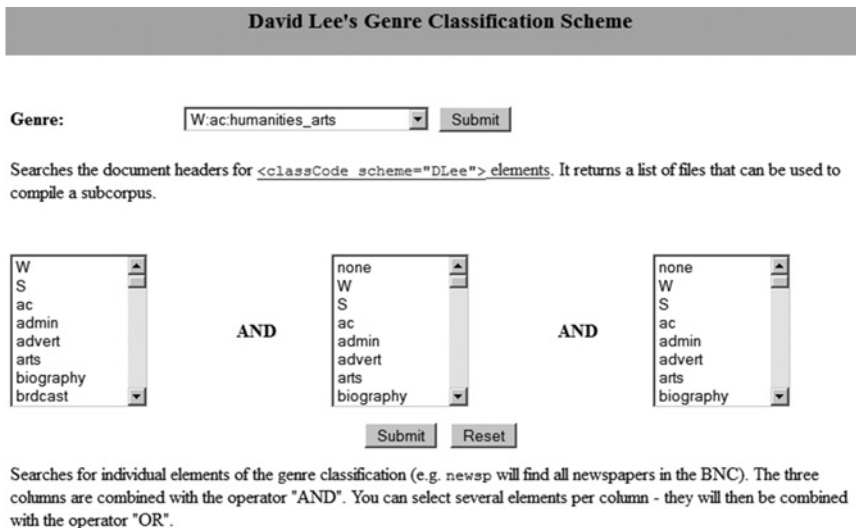


Figure 9.9 Options for defining subcorpora according to genre

In cases where you think you might not need a high degree of control over the individual meta-textual factors, you might well want to investigate first whether going through the options listed at the top is likely to give you the relevant results. Let's practise this in trying to find some suitable data we could compare to the essays from the *Uppsala Student English corpus* that we downloaded from the Oxford Text Archive in Exercise 9.

Exercise 61

From the ‘Make/edit subcorpora’ page, choose ‘Genre labels’ from the dropdown list, and click on **Go!**.

Open the ‘Genre’ dropdown list and see whether there may be a suitable pre-defined category. Strong hint: Remember, we want essays written by university students.

Select the appropriate category and then click on **Submit**.

Add all the files that were found to a new subcorpus named `university_essays`.

Create a frequency list based on the new subcorpus, import it into a spreadsheet, and sort it as we just did in the previous exercise.

The last form of creating a subcorpus we’ll discuss here is to use a ‘Keyword/title scan’. ‘The Keyword(s)’ search option allows you to select files based on library keywords or BNC-specific ones. We’ll ignore this option here, though, and will instead use the ‘Title word(s)’ scan with the option ‘any word’ to search for tutorials and lessons as depicted in Figure 9.10.

Scan BNC Keywords and Titles

Match: all words any word

Keyword(s): **Type:**

Searches the document headers for <keyword> elements (see the *User Reference Guide*, section 5.3.5: <txtClass>). Returns a list of files with matching keywords.

Match: all words any word phrase

Title word(s):

Searches the document headers for <title> elements (see the *User Reference Guide*, section 5.1.1: <titleStmt>). Returns a list of files with matching title words.

Figure 9.10 BNCweb keyword and title scan

Exercise 62

Open the ‘Keyword/title scan’ page and fill in the relevant information, then click **Submit**.

Include all files in a new subcorpus called `tutorials_and_lessons`.

Go to the ‘Make/edit subcorpora’ page and ‘Compile’ the frequency list. Save the list as a text file named `bnc_tutorials_and_lessons.txt`.

Based on the above exercises, it may now appear as if subcorpora created in this way are only useful for comparing data from the BNC with your own, but of course they can also be created in order to carry out analyses or comparisons in particular domains within the BNC itself, both in terms of frequencies and other options, such as ‘simple’ concordancing. To do the latter on a subcorpus you’ve created, you can simply select your corpus from the BNCweb start page from the dropdown list next to where it reads ‘Restrictions’.

9.3.3 Keyword lists

Another way to identify genre/domain relevant vocabulary is to generate *keyword lists* (see Scott 1997). This approach is essentially based on comparing two word lists, one from the corpus under investigation – referred to as the *source* corpus – and the other from a (general) reference corpus – the *target* corpus. The output then shows the *significant differences* as a filtered word frequency list of the corpus that’s being analysed. These differences between the lists are usually computed based on the average values of types occurring in both corpora, potentially using different statistical analysis algorithms, such as the *log-likelihood* ratio or a chi-square(d) (χ^2) test.

A further distinction can be drawn between *positive* and *negative* keywords, where the former represent types with an unusually high frequency in the source corpus, while the latter are types with an unusually low frequency in comparison to the target corpus.

9.4 Keyword Lists in AntConc and BNCweb

9.4.1 Keyword lists in AntConc

As of version 3.2.3, AntConc makes it possible to create keyword lists in two different ways, either by specifying a list of files that ‘act’ as a reference corpus, or selecting a frequency list created from such a set of files previously. The latter has definite advantages in that you don’t need to select and load a number of different files each time, but only a single one, which is also much easier to exchange with colleagues who may not have access to the original data you used, or, in our case, to use a frequency list based on part of the BNC. Figure 9.11 shows the ‘Keyword List Preferences’ settings we’ll later use to investigate the differences between the trains data and the spoken part of the BNC:

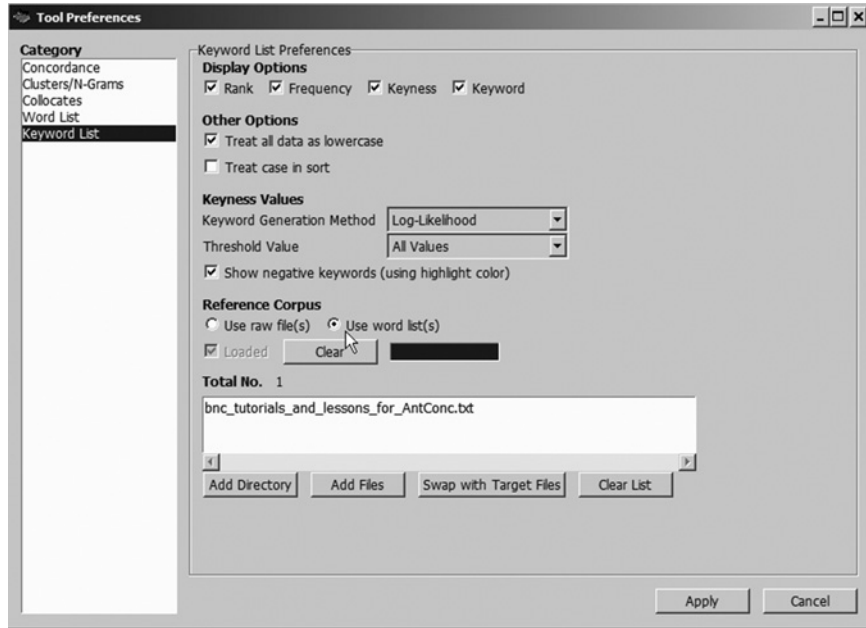


Figure 9.11 AntConc Keyword preferences

The format expected for the frequency list file by AntConc is RANK FREQUENCY WORD (separated by whitespace or tabs). This unfortunately isn't the same as in the frequency list we can extract through BNCweb, which has the order RANK WORD FREQUENCY, so we first need to change the order of the columns before we can use the BNC list in AntConc. We'll do this as part of the exercise for creating the keyword list below.

Exercise 63

Import the frequency list you named 'bnc_tutorials_and_lessons.txt' into the spreadsheet application as we've done before.

AntConc uses a different concept of 'word' from BNCweb, based on the token definitions you've chosen, and may thus throw an error when it encounters something that doesn't fit this definition, aborting the keyword list comparison. Because of this, it's best to prune the list as described in the solution to Exercise 60. If you still want to retain the original list without pruning, though, you can use a little trick and simply add a # symbol in front of the number indicating the rank, and when you later save the list as text, all lines marked thus will be excluded from the analysis when you use it in AntConc.

Once you've cleaned up the list, sort it as before, and then change the ordering of the second and third data columns, which require swapping. In order to do so, click on the column header above the word 'Frequency' (which should normally be 'C'), thereby selecting the whole column. Next, use a right mouse click inside the selection and choose 'Cut'.

Select the 'Word' column ('B'), right-click inside the selection, and choose 'Insert Cut Cells'. This should effectively have swapped the columns for you.

Delete the header row, as AntConc only requires the data rows.

Save the file as a text file, choosing the option 'Text (Tab delimited) (*.txt)', and naming it 'bnc_tutorials_and_lessons_for_AntConc.txt'.

Open AntConc and load the Trains data via the 'File' menu.

Next, change the 'Tool Preferences' for 'Keyword List' as shown in Figure 9.11. To do this, select 'Use word list(s)', click on **Add Files**, select the frequency list file you just created, click on **Load**, and finally on **Apply**. If you don't get an error message regarding an incorrect token definition, skip the next step.

Any errors reported by AntConc (as of version 3.4) will indicate both the line number of the error in your list file as well as the token entry that caused the error. The easiest way to fix the error is to open the file in your editor, locate the line, and prepend a # to it. Hint: In Notepad++ and many other editors you can jump straight to the line by pressing 'Ctrl + g', and typing in the line number. Save the file, and click on **Load** again, repeating this process until the file loads successfully, then click **Apply**.

Go to the Keyword List tab and click **Start**. Unless you've previously created a frequency list of the Trains data, AntConc will tell you that it needs to create the list, which you can simply accept.

To compare the two lists, you can either use the **Clone Results** button on the 'Word List' or the 'Keyword List' tab, and then switch to the other tab, ideally positioning the two windows side-by-side.

As Exercise 63 will have shown you, the keyword list, at least in our case and for positive keywords, may not necessarily provide you with more information than a frequency list that has been filtered well through the use of stopwords. However, unless you need to prune the reference list extensively, it can certainly allow you to identify some key terms much more quickly, and may therefore be seen as an alternative way of looking at single-word lists for identifying genre-dependent or semantic features of a corpus. In addition, the ability to highlight negative keywords in AntConc may also allow you to investigate under-use of specific vocabulary relatively easily, for instance when comparing learner data with that produced by native speakers, etc., an option that, obviously, a pure frequency list of only the source corpus is unable to provide.

9.4.2 Keyword lists in BNCweb

BNCweb provides two different ‘views’ of what may be ‘key’ in different subcorpora/domains, one where the differences between the occurrences of words are listed side-by-side, with the ‘dominant’ side highlighted, and the other where it’s possible to display only lists of words that occur in one subcorpus/domain or the other. The options for this are depicted in the top and bottom halves, respectively, of Figure 9.12, and we’ll explore each of them in turn.

The screenshot shows the BNCweb (CQP-Edition) interface for keyword analysis. The main title is "Keywords". Below it, there are two dropdown menus for "Select frequency list 1:" (set to "Spoken component") and "Select frequency list 2:" (set to "Written component"). A "Compare:" dropdown is set to "Words". Under "Options for keyword analysis:", there are four fields: "Min. freq(list 1):" (5), "Min. freq(list 2):" (5), "Method:" (Log-likelihood), and "Significance threshold:" (0.01%). A "Calculate keywords!" button is located below these options. Under "Options for comparing frequency lists:", there is a "Display words that only occur in:" dropdown set to "Frequency list 1" and a "Compare lists!" button.

Figure 9.12 Keyword options in BNCweb

In order to be able to see how the output of the two options differs, it’s best to do some exercises where we get to evaluate them, starting with the side-by-side comparison of keywords.

Exercise 64

Go to the ‘Keywords’ page and select the subcorpus ‘university_essays’ we created in Exercise 61 for the first frequency list, and ‘Written component’ for the second.

Leave the ‘Compare’ option set to ‘Words’, and all the analysis options set to their defaults. In our exercises, we’ll actually trust them to be correct and will never make any changes to them.

Click on **Calculate keywords!** and try to understand the results.

Let’s discuss the output briefly to clarify some of the points you will hopefully already have noticed. In order to do so, we’ll use a screenshot (Figure 9.13) showing the top 31 results of the log-likelihood (LL) analysis, a statistic that compares the frequency of the items in both subcorpora on a basis that is independent of the sample size.

The screenshot shows a window titled 'Keywords for 'university_essays' and 'Written component''. It contains a table with 31 rows of keyword data. The columns are: No., Word, Frequency in 'university_essays', Frequency in 'Written component', +/-, and LL-value. The table is sorted by LL-value in descending order. The 'Frequency in 'Written component'' column is hyperlinked. The '+'/- column indicates the direction of keyness: '+' for higher frequency in the first subcorpus and '-' for higher frequency in the second subcorpus. Darker grey shading is used for the 'Frequency in 'Written component'' column when the frequency is significantly higher than in the first subcorpus.

No.	Word	Frequency in 'university_essays'	Frequency in 'Written component'	+/-	LL-value
1	communication	94	5,845	+	428.15
2	i	56	559,077	-	382.72
3	subjects	91	7,444	+	367.16
4	citizenship	51	878	+	358.32
5	hemisphere	49	894	+	338.8
6	women	141	35,792	+	281.08
7	she	16	308,706	-	274.77
8	verbal	46	1,456	+	269.18
9	stimulus	45	1,411	+	264.15
10	he	102	564,053	-	248.89
11	recall	51	2,794	+	244.63
12	adaptation	36	757	+	239.07
13	stimuli	37	883	+	236.68
14	experiment	50	2,998	+	231.21
15	prisms	21	46	+	226.55
16	power	114	30,217	+	219.02
17	political	112	29,369	+	217.18
18	tachistoscope	17	19	+	201.03
19	active	59	6,954	+	197.83
20	paces	17	22	+	197.41
21	visuo-spatial	17	23	+	196.28
22	her	30	287,695	-	192.66
23	neurotransmitter	20	89	+	191.27
24	participatory	21	130	+	188.19
25	rfa	16	19	+	187.81
26	democracy	47	3,930	+	187.69
27	risc	30	815	+	184.4
28	memory	55	7,008	+	176.54
29	reaction	49	5,154	+	174.63
30	adjectives	25	481	+	170.3
31	you	75	398,900	-	169.01

Figure 9.13 Keyword comparison of university essays and written component of the BNC (top 31 entries)

Essentially, there are two separate indicators of ‘relative’ skewness of the frequency occurrences of the individual types. The first one is that whenever the type occurs with a significantly higher ‘relative’ frequency in one of the subcorpora, the (hyperlinked) frequency output field inside the table for this type will appear in a darker shade of grey. The second one can be found in the ‘+/-’ column, where a + indicates a positive keyness in the first subcorpus, and a - indicates a negative keyness. In other words, whenever there’s a +, the relevant type occurs with a statistically significant higher frequency in subcorpus 1 (written), but if there’s a -, it’s much rarer in subcorpus 1 than in subcorpus 2.

To see how the list comparison feature works, let’s use a slightly different approach.

Exercise 65

Start a ‘New keyword calculation’.

Change the frequency list options to ‘Written component’ and ‘Spoken component’, respectively.

Set the ‘Compare’ option to ‘Word + POS-tag combinations’ and leave the ‘Display words that only occur in’ option set to the default, ‘Frequency list 1’.

Run the comparison by clicking on and evaluate the results. Do you notice anything unusual? To check on ‘strange items’ in the list, you can use a right mouse click on the frequency to display a concordance of the item in a new tab. Please note, though, that for some special characters this link may not work, in which case you may need to change the URL part behind where it reads ‘theData=’, either simply putting a backslash after the = sign, or sometimes, if the special character has been encoded, replacing everything between ‘theData=’ and ‘&chunk’ by a backslash followed by the character in question.

As the previous exercises have hopefully shown, keyword analysis does have at least some potential in identifying domain-specific content, although it doesn’t necessarily always perform better than a well-executed single-word analysis. The latter will generally take a little longer to set up, but may in fact also force the researcher to pay closer attention to the data. One further caveat in the automatic generation of keywords based on statistical measures has been very clearly summarised in Scott (2010: 50–51), who states that

The order of KWs is not intrinsically trustworthy, because it depends not only on the frequency in the text we are studying [...] but also on their frequencies in the reference corpus.

and

[...] that there is no statistical defence of the whole set of KWs, but only of each one, though the more there are the higher the chance that some of the comparisons came up by chance, and that it is not certain that the order of the items in the set itself reflects their importance. The implication of those conclusions is that KWs are pointers which suggest to the prospector areas which are worth mining but they are not themselves nuggets of gold.

It’s thus well worth bearing the above-mentioned factors in mind when conducting any kind of statistics-based keyword analysis, and especially when reporting on the presumed importance of particular keywords for a given text/corpus.

9.5 Comparing and Reporting Frequency Counts

While the keyword analysis procedures in AntConc, BNCweb, and other concordancers/interfaces perform comparisons of frequency lists with the express purpose of identifying keywords or unique word types, sometimes we simply want to compare the distributions of types in two different sets of data. In order to do so, we may need to *norm* our frequency counts to make them comparable because otherwise the size of the corpora may provide us with misleading information.

In order to be able to norm the data, we essentially need to establish the relative frequency of the tokens and multiply these by a sensible common factor/denominator. I deliberately said “sensible” here because the literature frequently only refers to fixed factors, such as per thousand/ten thousand/one million words, disregarding that, for corpora whose overall size is less than these factors, this would be mathematically inaccurate and exaggerate the relative size by interpolating values that do not actually exist (cf. Lindquist 2009: 42). For instance, despite the fact that Biber, Conrad & Reppen (1998: 264) caution against using too high a basis themselves, their made-up norming example for the occurrence of modals in two texts, where one text is only assumed to contain 750 tokens and the other 1,000, still reports results per 1,000 words. To see the effect of this, let’s take a look at Table 9.2.

Table 9.2 Recalculated norming sample from Biber, Conrad & Reppen (1998: 263)

<i>text</i>	<i>modals</i>	<i>size</i>	<i>rel. freq.</i>	<i>x500</i>	<i>x750</i>	<i>x1,000</i>
1	20	750	0.027	13	20	27
2	20	1,200	0.017	8	13	17

Table 9.2 illustrates the normed frequency counts for three factors, two of which, 500 and 750, are common denominators of both texts, while the last one, 1,000, isn’t. As we can easily see from this example, norming by 1,000 here exaggerates the overall number of occurrences, pretending that there are in fact 7 more instances, thereby also increasing the perceived difference between the two texts. In the same way, but without postulating non-existing values, norming by 500 plays the difference down a little, though not quite as much as the interpolated counts do. This clearly demonstrates that it’s generally best to pick a number closest to the lowest common denominator for comparison, which may even be the exact size of the smallest corpus, although then the numbers may end up not looking very round.

Of course, when comparing individual items in two texts/corpora directly, one doesn’t even need to normalise in this way, but can easily get an indication of the differences by looking at the ratio of the relative frequencies, something that, unfortunately, none of the textbooks I consulted suggests, although, as we saw in

Section 8.2.2, the BYU interface uses this in the comparison across corpora. For instance, the ratio for the above example, which we obtain by dividing the relative frequency of text 1 by that of text 2, would be 1.6, which clearly indicates that modals are more than 1½ times as frequent in text 1. Conversely, if the ratio had been below 1, we could easily have observed that they were more frequent in text 2. In order to test these two different options and to see how we can use them, let's do another exercise, based on more data from BNCweb, where we investigate potential differences in the use of positions in economics texts.

Exercise 66

Open BNCweb and create two more subcorpora. Name these `commerce_general` and `commerce_in_newspapers`, based on the genre labels 'W:commerce' and 'W:newsp:other:commerce', respectively.

Open your spreadsheet program and create a header row that looks like this:

rank_g	type	freq_g	freq_n	rel_g	rel_n	ratio	rank_n	type_n	freq_n
--------	------	--------	--------	-------	-------	-------	--------	--------	--------

Notice that `freq_n` appears twice. This is deliberate, as we'll later transfer frequencies from one column to another.

Save the spreadsheet and call it 'norming', plus suitable extension, depending on which program you're using.

For both subcorpora, extract and put the top 50 items into the spreadsheet via copy-and-paste. When you paste the data, make sure you use the 'Paste Special...' option and select 'Unicode Text' (or 'Text') because otherwise the numbers in the frequency column may not be interpreted as numbers by the spreadsheet, but still retain some HTML coding. Paste the data from the general subcorpus into the cells below `rank_g`, `type`, and `freq_g`, and the other data into the corresponding rows `rank_n`, `type_n`, and `freq_n`, for now leaving the cells in between the sets empty.

Sort both lists alphabetically, but independently of one another. For the general set, you should sort according to `type`, and for the newspaper data, according to `type_n`.

Next, get the token count from the 'Make/edit subcorpora' page and paste it into the spreadsheet, ideally at the top and to the right of the second frequency list, as we may need to shift some items in the list down later to align the data. Once you've pasted a total, click in the box immediately above cell A1 of the spreadsheet and type `n_general` and `n_newspapers`, respectively, followed by pressing the Enter key. This will name the cells for you and later make it easier to calculate the relative frequencies.

To align the data, check to see which types in the columns `type` and `type_n` are identical, and transfer the corresponding values from the right-most `freq_n` column to the one on the left.

When you find any differences, either in the general or newspaper list, insert new rows for the types in the spreadsheet by clicking on the row number where the difference occurs, clicking the right mouse button, and choosing ‘Insert’ (Excel) or ‘Insert Rows’ (Calc), which will create a new blank row above the selected one.

Transfer the type that only exists in one list into that row, making sure that the rank is also moved to the appropriate place for the list it occurs in, and setting the rank and frequency in the list where the type’s missing to 0.

Continue this operation until all the cells below the headings `type_n` and `freq_n` are empty, then delete these columns. If you think you’ve made a mistake, don’t panic, but simply use ‘Ctrl + z’ to undo the last few operations.

Save the file.

Next, place the cursor in the first empty cell in the `rel_g` column, 3 across from the word *about*, type `=`, then click in the cell below `freq_g` that contains the frequency for this word, type `/`, and click on the number that contains the total for the tokens in the general corpus (Excel), or type `n_general` (Calc), then press the enter key. The formula in the formula bar when you click in the same cell again should then read `=C2/n_general`. You’ll probably need to adjust the decimals display for the cell until you see something meaningful (i.e. other than just 0) because the resulting number will be quite low.

Repeat this step for the frequency in the newspaper corpus, ensuring that the formula bar reads `=D2/n_newspapers`.

Click in the relative frequency cell for *about* in the general corpus. You should see a black frame with a small filled square also black now. When you hover the cursor over that square, it should turn to a black cross. Once you see this cross, click and hold the mouse button down and then drag all the way down the same column to the cell across from the word *worth*. When you release the mouse button, the spreadsheet application will automatically have calculated and filled in all the relative frequencies for the general corpus.

Repeat the same process for the newspaper corpus.

Save the file again.

In the final step, we’ll calculate the ratio. This can easily be done by clicking in the cell below the heading `ratio`, again typing `=`, clicking the cell containing the general relative frequency to select it, typing `/` again, and

then selecting the cell containing the relative frequency for the newspapers, and pressing enter.

Now you can simply select that cell again and drag downwards to get the spreadsheet program to automatically calculate all the remaining ratios for you. Don't worry about any errors that read #DIV/0, as these are only due to instances where the newspaper subcorpus didn't have any tokens at all, for which there's obviously no ratio 😊

Once you've finished, don't forget to save the file again, and then see whether you get any interesting results.

One thing we also need to bear in mind when discussing or reporting on frequency comparisons of these two types, either presenting normed frequency counts or ratios, is that, while they're fairly uncontroversial for comparisons of single texts, for comparing corpora, both forms assume a certain level of homogeneity of the data and do not take any variability or *dispersion*, that is, the distributions across different files, within the individual corpora into account.

As the above discussion should have indicated to you, it's very important to have all the relevant information, not only about the normed number of tokens, but also about overall size (and composition) of all corpora involved in the comparison in order to be able to judge frequency comparisons properly. This is why we should usually ideally also report the raw frequency and the corpus sizes along with any normed counts, which will then enable fellow researchers to judge our results fully.

9.6 Investigating Genre-Specific Distributions in COCA

The BYU interface makes it fairly easy to pick two sub-genres and compare the frequencies of specific words or word classes, although it unfortunately doesn't allow you to create, save, or download frequency lists as BNCweb does. On top of that, being a monitor corpus, it also makes it possible to track potential changes across different periods in the same way. To see how the former works, let's do a short exercise exploring the distribution of modal verbs in two different types of academic writing.

Exercise 67

Open the COCA interface.

In the display frame on the left, keep 'LIST' selected, then click in the 'WORD(S)' box to select it.

Next, click on ‘POS LIST’ and choose ‘verb.MODAL’. This should put [vm*] into the ‘WORD(S)’ box above.

In the two list boxes below, scroll down until you find the sub-section for academic texts, which start with ‘ACAD’, followed by a colon.

In the box on the left-hand side select ‘ACAD:Humanities’, and in the one on the right-hand side ‘ACAD:Sci/Tech’.

Change the ‘SORT BY’ options to ‘FREQUENCY’.

Click and observe the results. To verify the results for any particular type in either subcorpus, you can use the hyperlinks in the columns labelled ‘TOKENS 1’ and ‘TOKENS 2’, respectively to have KWIC concordances displayed in the frame in the bottom half.

Unfortunately, the BYU interface doesn’t allow you to save these results to disk, due to copyright restrictions, so, to store the results on your own computer, you’d need to use the copy-and-paste options we explored earlier for extracting parts of frequency lists from BNCweb, and then re-arrange the results a little in your spreadsheet program, as the headings will unfortunately be mis-aligned, due to the web layout.

Solutions to/Comments on the Exercises

Exercise 53

The first thing you should perhaps have observed before even looking at the list itself is that the number of word types is 567, and that of word tokens 17,081. We’ll learn more about what this may mean in Section 9.1.2.

If you’ve studied the list fairly closely, you should have recognised a number of things: In terms of text type/category, the fairly high number of tokens for words, such as *okay*, *so*, *uh*, *um*, etc., clearly indicates that we’re here dealing with a corpus of spoken texts.

The initially perhaps odd-seeming type *sil* is further proof of this, as it represents an abbreviation for ‘silence’, that is, a pause of undefined length, while *utt* represents markup for an utterance. You can verify this for *sil* by clicking on it in the ‘Word’ window, which will take you to a concordance view of the item, where you can also see that this annotation normally appears in angle brackets in the source texts. If you do the same thing for *utt*, you should be able to notice that each occurrence is followed by the number of the utterance in the dialogue, a number of spaces, then a colon, sometimes followed by space + *s* or *u* + colon, then again a number of spaces, and finally the utterance itself. Here *s* and *u* represent codes for the speakers, and they only occur when the *turn* changes from one speaker to the other.

However, *s* also appears more frequently (834 times) in our list than *u* (only 666 times), which would be strange in a dialogue because we'd expect both speakers to contribute more or less equally to the discourse, so we need to look for a different explanation here. Again, clicking on the item and investigating the concordance lines will soon tell us that *s* isn't only used to mark a particular speaker, but of course also represents the clitic (contraction) forms of *is* (as in *that's*) and *us* (as in *let's*), although there are no possessive markers in the corpus. It also occasionally represents the first letter of interrupted words, where it's followed by a hyphen to indicate the ellipsis.

Apart from the high incidence of *discourse markers* and *fillers* referred to above, as well as certain contracted forms like *gonna* or *wanna* (cf. Section 4.3), that are such typical indicators of spoken interaction, the number of function word tokens is also fairly high.

There are also a number of indicators that, taken together, may help us in identifying the topic/domain of the dialogues: frequent reference is, for example, made to place names (*Corning, Elmira, Bath, Dansville, Avon*); the preposition *from* occurs 136 times, while *to* occurs even more frequently (699 times; of course, probably at least partly due to its function as infinitive marker); the verbs *take* and *go* occur 162 and 118 times, respectively. There are also a number of general nouns that might help us to identify the domain more precisely, but this is something we'll hopefully be able to make clearer soon, once we filter the list successively. For the moment, we can at least say that the dialogues involve some form of movement between different locations, and that certain things, such as *oranges* or *bananas*, are being transported.

Exercise 54

The first thing you can observe here is that the number of word types has increased to 644 (from previously 567), but the number of tokens in fact decreased to 16,582 (from previously 17,081). If this hasn't happened for you, you've probably not applied the settings properly before re-creating the list ☺ So, how do we explain this when it would appear that, by allowing words to be joined together, we should in fact end up with both fewer types and fewer tokens? The explanation lies in the fact that, by allowing word forms to be joined, we make it possible for word forms that previously may have only existed independently, such as, for example, *it*, *that* and *ll* in *it'll* or *that'll*, or *he* and *s* in *he's* to now occur as parts of combinations that may exist as independent types, whereas previously they would have been subsumed as tokens under the individual types of their parts.

Another reason for why the number of tokens has risen here is that the hyphen in fact appears to be ill-defined; as in most instances in this particular type of data, it isn't actually a true hyphen as it would appear in hyphenated words or at the end of a line that has been hyphenated, which would not occur in this type of data, anyway. In terms of regexes, a true hyphen would be defined as `\w- (\w|§)`,

that is, a ‘dash’ that has to be preceded by a word character and either followed by another word character or the end of a line (and one on the next as well), while our definition here essentially only covers the former and a ‘dash’ that appears to occur independently, which is how it appears in the list. However, upon closer inspection of the data, it turns out that the hyphens that are supposed to signal incomplete words in the Trains data may also be preceded by what the transcriber assumed to be the completion of the incomplete word, but surrounded by round brackets, which do not form part of our definition of what a word may contain.

Furthermore, if you scroll down the list until you reach rank position 78 (*can*) and 79 (*can't*), you'll now be able to observe what I referred to earlier, i.e. that, generally, contractions in alphabetically sorted lists tend to be sorted together with their un-contracted, or positive, counterparts. A perhaps even more interesting example of this can be found at positions 116–118, where, apart from the negated contraction form of *could*, *couldn't*, we also find the unusual form *could've*, which serves as a perfect example of a transfer from phonological to orthographic form gone wrong, as it's actually incorrect in missing an indication of the vowel (shwa), since phonetically/phonologically, this would be [kudəv] and not [kudv], as the (conventionalised) spelling suggests.

Through being able to observe the above, it should now definitely have become even easier to recognise the spoken nature of the dialogues, both in the high number of contractions and false starts, where the latter signal the kinds of disfluencies that are an integral part of normal, unplanned language, as speakers have to sometimes modify their utterances when they realise that they may have said something wrong, or maybe need to formulate what they want to say in a different way to make it more explicit.

Exercise 55

The first thing you should note when looking through the alphabetically sorted list is that now the hyphen (due to our current word definition) appears at the top of the list, followed by the lowercase letter *a* and its uppercase equivalent *A* and the lowercase *a* plus hyphen (*a-*). Now, whereas before the ‘Rank’ field correlated with higher frequencies, here this is no longer the case. Interestingly enough, though, AntConc does appear to have a secondary sort order based on the frequency because otherwise uppercase *A* would have to appear before lowercase *a*, and the latter is only ranked higher because it has a frequency of 161 as opposed to a single token of the former. And, just in case you're curious to find a single letter *A* in the data (as you should be), you can investigate this through the concordance by clicking on it. If you do this without first checking the option for ‘Case’ next to the ‘Search Term’ box, you may be in for a surprise because you'll suddenly end up finding all 162 tokens for both upper- and lowercase forms, which is of course not very useful. This is because the default option for the concordance module is to ignore case.

Scrolling further through the list, you'll probably notice many incomplete words, all indicated via a hyphen at the end, as well as a few combinations of 'stranded' characters that, upon closer inspection, will again turn out to be instances of words where the transcribers have tried to complete originally incomplete words to the best of their understanding. This goes to show that, by observing items in a frequency list, we may often be able to see things we might have overlooked or ignored while concordancing, simply because the results would have been easier to understand. Something similar goes for observing instances of information that describe non-verbal, vocal events in the dialogue, such as *clear-throat* or *laughter*, which appear out of place in comparison with the rest of the vocabulary.

Exercise 56

Of course, ultimately, the list that you'll have constructed will reflect your own choices. However, as a general rule, some of the words that you might have chosen to exclude apart from the ones I mentioned earlier will probably have been discourse markers, such as *so* or *right*, fillers, such as *uh* and *um*, response markers like *yes* or *mm-hm*, conjunctions like *and*, indications of non-verbal behaviour, such as, for example, *brth*, that signals audible breathing, references to meta-information regarding the dialogues themselves (e.g. *Dialogue*, *Estimated*, *files*, *Length*, *Number*), etc. Many of these may not be considered stop words in a general sense, and would therefore not be applicable to other types of files/domain, but are highly particular to this specific type of dialogue.

Exercise 57

I've deliberately left the choices for this exercise open to encourage you to explore the different options, but just in case you run out of ideas, here are a few options/combinations you could investigate. For instance, it might be interesting to investigate nouns first. As the default option for the sorting is 'descending', this will automatically show you the most frequent ones first. If you use the same option I'd set, that is, for only spoken parts of the BNC, you may be in for a bit of a surprise, as you may not just be getting an indication of the most popular 'topics' (*people*, references to times, *money*, *thing(s)*, etc.) through this, but also find a number of unusual little words there, such as *way*, *bit*, *sort*, and *lot*. Under normal circumstances, we'd probably not expect to find these nouns to be among the most frequent ones talked about in conversations. Upon closer inspection, though, it turns out that they simply form parts of longer multi-word expressions that tend to characterise spoken language, such as *in a way*, *this way*, *no way*, for *way*, *bit* mainly occurring in *a bit*, *sort* as part of the hedging device *sort of*, and *lot* in *a lot*, all of which don't signal any specific topic information. Switching to written language, we do get some of the items, namely *people*, *way*, as well as

references to times/dates, recurring, but most other nouns are already far more indicative of content, referring to such topics as family life, politics, locations like London, business, etc. I'll leave it up to you to investigate this in more detail. If you do so, though, and want to draw conclusions about written language only from this, please also bear in mind that the written part of the BNC also includes fiction, where at least some parts 'model' spoken language, as in the direct speech of characters in novels or plays, etc., so that we should perhaps not see everything here as exclusively written.

Of course, you could also attempt to investigate the nouns in the BNC from a different angle, which would be to look at the rarest ones first. In order to do so, we could simply switch the 'Type of ordering' option to 'ascending' and then see which rare, or perhaps exotic, nouns we may find. Unfortunately, though, this won't have the desired effect, as the first 3,900 word forms will represent numbers (with or without a dollar prefix), foreign words (apparently mainly Czech), etc., which may in many instances – at least as far as the numbers are concerned – well act as NPs, but not actually constitute nouns themselves. Later, predominantly mis-tagged forms contain leading quotation marks or hyphens, or further numerical/measurement units that are at best pre-modifiers, rather than nouns, where in many cases a number has not been split off from its following unit of measurement. Other similar errors continue up until we reach approximately 12,000 word forms or so, as, up to this point and a little further on, most of the types constitute *hapax legomena*, that is, word unique forms. From roughly this point onwards, at least some of the following hapax legomena appear to be proper names, so this is perhaps where the tagging errors gradually begin to peter out. And even if this number represents only 0.01% of all the words in the written parts of the BNC, the number of potential errors, which appear mainly due to tokenisation errors, is staggering, particularly when considering that this affects only one of the parts of speech represented in the corpus. You may then think that perhaps if hapax legomena are such a problem, you could start investigating instances of types that occur at least twice. However, sadly, these are beset by similar problems, again mostly related to mis-categorised numbers, such as recurring years or even apparently numbers indicating section hierarchies in documents (such as 4.2.3, etc.), where approximately the first 7,700 types are concerned before we end up with more likely noun-candidates. Increasing the minimum frequency to 3 improves the situation somewhat, as the cut-off point for what may be genuine nouns is now 'already' at around 3,180, but what this clearly shows is that, due to too many tokenisation or tagging errors, perhaps the BNC is unfortunately not the best resource for investigating rare vocabulary! One thing we can already conclude from this, though, is that CLAWS seems to have some clear issues when it comes to tagging nouns, especially where numbers are concerned, but also in cases of pre-modification, where frequently pre-modifying adjectives are 'mistaken' for nouns, possibly because the probabilistic component of the tagger may over-rate the potential of noun-noun combinations in compounds or a similar, non-probabilistic, rule may be causing the issue.

For verbs, perhaps the most obvious thing to investigate would again be the most or least frequent types, but, as these still exhibit far more inflectional options than other parts of speech, it may also be interesting to investigate inflectional suffix patterns by selecting the ‘ending in’ option for ‘Word pattern’, or possibly verbs that have potential negative prefixes by choosing the ‘starting with’ option. The final option of this set, ‘containing’, is probably not exclusively relevant for verbs, but could be used to investigate derived forms of nouns, verbs, adjectives, or adverbs based on restrictions for particular stems.

Most other PoS categories can be investigated in similar ways, either looking for pre- or suffixes, most/least frequently occurring types, etc.

Exercise 58

You should already be somewhat familiar with the composition and features of the BNC from Exercise 4 in Chapter 2, as well as some other discussions in Chapter 3. Just as a reminder, though, under the ‘Overall’ group, ‘Demographically sampled’ refers to materials collected from tape recordings made by individuals in private settings, and ‘Context-governed’ to data transcribed from public speeches or other events, news or sports commentaries, lectures, classroom interaction, etc.

Both of these have their individual sections for restrictions further down the page, to allow you to narrow down your choices, based on domain information for the context-governed type, and age, social class, and sex of respondents – not the speakers, whose characteristics can be selected separately – for the demographically sampled data. For instance, for the former category, you could here choose to only select educational or business materials, depending on whether you may be investigating issues in EAP or business-oriented ESP. For the latter, you could opt for only investigating the speech of men or women, or speakers from the same or different social classes, or create subcorpora for both, and then compare the language in some way, which makes it possible to carry out corpus-based sociolinguistic analyses on the data, as do the options for restricting speakers to particular education levels, etc.

To carry out genre-based analysis, you’ll probably want to focus on making particular selections from the ‘Genre’ section, which, to some extent, overlaps with the restrictions for context-governed materials.

Provided that you don’t forget to change the option for creating a new corpus, there should be no issues in completing this exercise and creating a suitable frequency list.

Exercise 59

This exercise may have seemed rather complicated to you, as it involves working with yet another program and also many, perhaps complicated-seeming, steps. However, as we’ve already seen above, in order to get the full picture, we cannot

always only rely on one single program or interface, and frequently need to process our data further in various ways.

The first thing we can see when looking at the top few items in the list is that, apparently, BNCweb treats all punctuation characters as word types, and hence also tokens. This is odd because, although punctuation may well convey linguistic meaning in some sense, it still doesn't carry any truly semantic meaning, but its function is more pragmatic in nature. We'd thus probably want to 'throw away' all types that constitute punctuation marks or mathematical operators, which also includes some character reference entities like ‘ (left single quotation mark), — (m-dash; dash that has the length of the character m), and ’ (right single quotation mark). The latter, strangely, are still part of the text representation of the XML version of BNCweb, despite the fact that they can easily be encoded in UTF-8, which is, after all, the assumed default encoding for XML.

Likewise, when scrolling through the list, you may notice that there are some forms preceded by apostrophes, which either represent true clitics or sometimes just indicate non-standard pronunciation forms, such as *b-dropping*. The question here is whether we'd really want to keep them in this form or possibly associate them with their full, non-contracted equivalents by increasing their counts and deleting the contracted forms, or keep on treating them separately. A little further down, we also find a series of numbers, some of which represent years. However, in many contexts, number tokens are really not terribly meaningful, either, so we could opt for removing them to prune the word list, especially because most of them represent singletons, anyway.

At the end of the list, you'll also find some entries related to anonymisation in the data, where names, addresses, or telephone numbers were marked as having been elided in order to protect the anonymity of people or institutions. In many cases, these may well represent nouns, but as we have no way of identifying how many tokens they genuinely represent and which types they belong to, it may sometimes be safer to ignore and delete them. The same goes for instances of [unclear], which we again cannot interpret in any meaningful way.

Of course, any operations where we modify the data or remove items may also have an effect on further processing, as they'll affect the overall type and token counts that may, for instance, be relevant in comparing our list to lists from other domains, something we'll explore soon. Issues like this may affect both manual and automated processing, but can have more of a negative effect on automated and unsupervised analysis procedures because, there, possible issues will never be spotted, and the potential relationships identified between words may become skewed. Now, if you're worried about deleting data from the list, you can of course take advantage of the spreadsheet application in a different manner. As spreadsheet files normally consist of multiple worksheets, rather than completely deleting items you want to remove from the list for various reasons, you can copy/cut and paste them to an extra spreadsheet and preserve them there, just in case you may decide later on that you want to insert them again. That way, you also won't have to

go through the same routine to re-create the list in the online interface, should you notice that you've accidentally removed data you'll later need. And, if you're worried about not being able to get an exact token count of all the data after making modifications, the spreadsheet will also help you there because all you need to do in order to obtain this is to place the cursor in the field immediately below the count for the final token in the list and use the AutoSum function (symbolised by Σ) to automatically count the total for you.

One other thing you'll hopefully be able to see easily when looking at the list is that BNCweb has downcased all words, and that there was also no option to preserve the original distinctions between capitalised and non-capitalised forms, which now makes it difficult to distinguish between common nouns and proper names. Of course, you could fix this manually, but, since we've now separated the data from our ability to concordance on it, we'd have to go back into BNCweb to look at the original frequency list. One way in which we could have avoided this particular issue would of course have been to create the list including tags. This way, the different PoS categories, insofar as they were correctly tagged in the first place, would have automatically been counted separately. I'll leave it up to you to create such a revised list, and see how this may affect your interpretation of the data.

Exercise 60

The only two things you could possibly get wrong in this exercise are that, perhaps, you don't select all the data and get an incomplete sort, or that you don't use the right (custom) sort button and instead sort only according to one field. However, as you can hopefully see, the options we want to use here may be labelled somewhat differently in Excel/Calc, but are otherwise exactly like what we'd get if we sorted according to descending frequency in AntConc, as our secondary sort also takes the alphabetical order into account.

When you look at the results, you should also be able to notice that, similar to the results we had for the Trains corpus, most of the high-frequency function words get sorted to the top, as well as first and second person pronouns, and fillers like *erm*, etc. If you didn't delete the contraction forms in the previous exercise, these will also appear near the top. Otherwise, of course, we don't get such a clear indication of the topics as in the relatively small Trains data, which was restricted to a very narrow domain, along with a very specific task.

Exercise 61

This exercise should be relatively straightforward, provided that you pick the option 'W:essay:univ' from the dropdown list. And, of course, you need to make a decision about whether to include tags again or not, but, other than that, the steps required are just the same as before and should require no further comment.

Exercise 62

As before, there should be nothing special about this exercise.

Exercise 63

Provided that you've swapped the second and third columns correctly inside the spreadsheet application, as well as loaded the data file for the reference corpus properly in the preferences, the only real difficulty in this exercise is to remove all the items that don't look like proper words successfully, so that AntConc can load your frequency list. If you've followed the guidelines I gave above for Exercise 60, you'll already have eliminated most of the likely candidates for errors. Other things you'll need to delete are single letters followed by dots, as well as abbreviations, such as *i.e.* or *e.g.*, and any genuine words that are followed by dots, as the dots don't form part of the general token definition we're using, and most of these forms (apart from the abbreviations) are probably the results of tokenisation errors in the BNC, anyway. You should definitely also delete all numbers, unless you want to change the token definition to include those, but, as I pointed out before, numbers may take many different forms and their meaning may be difficult to identify.

Once you've created the keyword list, you should immediately be able to see that some of the words we'd previously only been able to identify through the basic frequency list after repeatedly manipulating the stop word list should now more or less automatically have 'jumped' to the top. If there are still some words in the list that seem odd, perhaps because they are part of the meta-information of the dialogues or indicate paralinguistic features, such as breathing, etc., you can easily eliminate them from the list in order to clean it up further by manipulating the stopword list, then re-creating the Trains frequency list, and re-running the keyword analysis.

Exercise 64

Even just a cursory glance at the positive keywords in the top 31 types of the keyword list already demonstrates the academic nature of the vocabulary quite clearly, containing vocabulary roughly related to scientific/psychological experiments, politics, as well as linguistics. The terms themselves may also be polysemous or refer to different domains, so it's useful to be able to use the hyperlinks behind the frequencies to generate concordances for disambiguation. For example, *subjects* predominantly refers to 'people involved in experiments', rather than the plural form of the 'syntactic role' category, while *memory* may refer to either 'brain capacity' in psychological experiments or to '(random-access) computer storage', etc.

In terms of function words, the high keyness of *the* and *this* points to the highly packed nature of written academic language, which contains many (definite) noun phrases and other types of deixis.

Looking at the negative keywords, we can easily notice that personal pronouns are much less frequent, partly due to the impersonal, non-interactive, 'objective'

nature of most types of academic writing. This becomes especially clear if we investigate the low frequency of *i* (lowercased, as all examples are automatically), which indicates a strong dispreference for authors to refer to themselves, in particular also because not even all instances of *I* always constitute tokens of the personal pronoun. Sadly, in very few cases here do we actually get a strong expression of an author's voice.

Exercise 65

Essentially, the option we just explored has relatively little to do with keywords as calculated through the options from the top part of the same page, as all it really does is eliminate all word types both corpora share, and then display whatever remains as a frequency list. Therefore, the list produced by BNCweb actually also contains one redundant column, depending on whichever list wasn't selected, and where all the type frequencies are set to 0. The idea behind seeing such words types as key is of course based on the notion that non-shared items are always key for a particular corpus, which may not necessarily be the case, even though they do help us narrow down the options for identifying true keywords without the use of statistics. The smaller the corpora compared are, the easier it'll of course become to narrow down such selections, but essentially, the technique itself is similar to creating basic stopword lists, only that, in this case, a word list from a whole corpus is used as a stopword list.

Although the comparison of such wildly different subcorpora in terms of size is, admittedly, not very useful in general, the list of unique items in the written component immediately reveals a number of interesting features of the BNC tagging and composition, or rather, the way BNCweb allows you to work with them. An extraordinarily large part of the unique types revealed as 'keywords' here constitutes items of punctuation, proper names (genuine names, abbreviated letters), cardinal numbers, dates, and measurements, many of these features that we've already previously identified as having the potential to skew our frequency analyses, and thus being able to filter these out in some way would be nice. Unfortunately, though, BNCweb doesn't offer us any direct way to do so, so we'd need to export a tagged list and then filter it ourselves in some way, for instance by excluding specific tag patterns in Excel or by writing a small program to remove words tagged with unwanted tags.

Exercise 66

As before, there are many steps in this exercise, and you need to be very diligent in moving around some of the data once it's been pasted into the spreadsheet. However, you'll hopefully soon realise how programs such as Excel or Calc make it much easier for you to calculate the values you want automatically, which saves a lot of time and is much less error prone (if done carefully) than calculating everything by hand. But of course, if you get some of the steps wrong, there's also

great potential for producing erroneous output, so we'll go through the individual steps here again and I'll try to point out potential issues or traps you might fall into, along with further opportunities the spreadsheet offers to visualise and investigate the data better.

The first part, creating the subcorpora, should be relatively straightforward, as we've practised this before. Creating the header row in the spreadsheet and saving it should also not present any problems.

The first issue could in fact arise when you paste the data from BNCweb into the spreadsheet. If you don't use the 'Paste Special...' option, which in Calc even triggers the text import wizard, you'll end up with frequency values that the spreadsheet cannot interpret as numbers because they still contain some (hidden) HTML formatting. In this case, there's simply nothing I've been able to identify that will make it easy for you to convert all the data short of editing all the cells manually ☒.

Sorting the lists (individually) should again be easy, as this time we don't even need to specify any additional sort key. Pasting the token counts from the 'Make/edit subcorpora' page also presents no problem because, this time, we only have one single piece of 'text' without HTML codes, so there's nothing to misinterpret for the spreadsheet program.

Naming the totals should not be particularly difficult, but if you should have a problem using the 'name box' above cell A1, you can right-click on the cell in Excel and choose 'Define Name...' or use 'Insert→Names→Define...' in Calc.

To transfer and align the data is probably the most difficult and time-consuming part of the exercise because it's easy enough to make mistakes when creating new rows and moving the data around, as well as adding the noughts to the relevant cells where a type doesn't exist in one of the corpora. Incidentally, the reason for why we added noughts for the ranks in the first place is that we'll later be able to sort the result table according to the rankings in either of the corpora and compare them, too, in which case all non-ranked, that is, non-occurring types will be sorted together. The same applies to sorting the data by frequency if we wanted to compare raw frequencies for whatever reason, or simply identify non-occurring types in either corpus.

Provided that you type the formulae into the formula bar correctly and select the right cells, calculating all the relative frequencies accurately and efficiently should also not present any problems, although, if you haven't used spreadsheets extensively, filling cells by dragging may take some getting used to. If you feel uncomfortable using the mouse for this, there's also an alternative way of filling the cells down, which is to click in the first cell that already contains a formula, then holding down the shift key, and using the down arrow to highlight all cells you want to fill. Once you've selected all the cells, you can press 'Ctrl + d' in Excel or use 'Edit→Fill→Down' in Calc.

What perhaps remains to be explained in order to provide you with a full understanding is how what we've just done actually works. By default, programs like Excel and Calc try to calculate formulae in adjacent cells using relative references.

For example, if you have a formula that sums two adjacent cells A1 and B1 using the syntax $=A1+B1$, and then drag down, the spreadsheet will ‘assume’ that you’ll next want to add A2 and B2 automatically, and adjust the formula accordingly. However, by naming the totals, and using the names inside the calculations for relative frequencies, we’ve effectively mixed relative references and absolute ones in one and the same formula, as a named cell always refers to a constant value contained in that cell. Thus, by dragging down the formula we created to calculate the relative frequency for *about* in cell E2 ($=C2/n_general$), we effectively changed it to $=C3/n_general$ for the next cell below, etc. In determining the ratio, however, we used relative references for both parts of the equation.

Once you’ve finished all the calculations, you can analyse and evaluate the results. As I said before, there are now a number of options for sorting according to different fields, for instance comparing the ranks in one corpus against another or, perhaps more importantly, seeing whether certain words dominate to some extent in one corpus in comparison. To do the latter, all you have to do is to sort according to the ratio. If you sort our data in descending order, all the types that only occur in the first corpus will automatically appear at the top, due to the division by 0 error I referred to in the instructions. This will then be followed by all instances where the relative frequency is higher in the first (general) corpus, and you can easily identify these ‘dominant’ words due to the fact that they’ll have a ratio above 1. However, even without sorting in this way, or sorting according to another column, the spreadsheet application would still allow you to visualise this easily by applying conditional formatting to the ratio column, where you could, for instance, highlight cells with values above zero with a green background and those below with an orange one, or any other colours you prefer. This is similar to the + and - symbols display in the keyword analysis in BNCweb, and represents just one of the ways in which we can visualise differences in the data easily for a quick overview. I’ll leave it up to you to find out how exactly this can be achieved in whichever application you’re using.

The analysis of the data, sorted according to the ratio, reveals a few interesting features, although overall it may not provide us with any deep insights. Looking at the words that exclusively occur in the general corpus, we can see some interesting types, namely *concerning* and *regarding*, that have been classified as prepositions despite the fact that they don’t look like typical prepositions because they are in fact ing-forms, that is, they clearly still retain some verbal character. Closer inspection of the original data in BNCweb further reveals that at least some of their tokens, especially for *concerning*, would perhaps better be classified as conjunctions, especially when occurring in clause-initial position. The occurrence of *v.* demonstrates that parts of the general subcorpus contain references to legal matters, where the type is one of the two possible abbreviations for *versus* (the other being *vs.*). Interestingly enough, the second form, which in fact occurs twice in the same subcorpus, is not used in a legal context, so the first form may (tentatively) be seen as a potential identifying factor for texts from the legal domain. The most striking difference in type frequency for types occurring in

both subcorpora is that for *upon*, which occurs more than 10 times as frequently in the general corpus (ratio 10.604). This can be interpreted as a difference in the levels of formality in the two subcorpora, as well as again partly to the occurrence of legal texts, which tend to be more formal (or formulaic) in nature, anyway. This would seem to be corroborated by the fact that *among* is somewhat underrepresented in the general corpus (ratio 0.712), which, however, exclusively has the alternative, and more formal form, *amongst* instead, as well as the relatively high level of occurrences of *within* (ratio 2.727) as an alternative to the less formal *in*. Of course, as with all analyses of word lists, even though the above results may lead us to assume that economics-related newspaper reportage may be less formal in style than other types of writing in economics, this assumption would still need to be substantiated further by more careful qualitative analyses based on concordances.

Exercise 68

This exercise is much easier to do than the previous one, so there's relatively little potential for getting anything wrong. Perhaps the only things that could happen here are that you either inadvertently select the wrong display option, or the wrong sub-genre from one of the list boxes. In the latter case, the interface will unfortunately not allow you to spot that error easily because it'll only refer to the two selections you've made as SEC 1 and SEC 2 in the comparison frame.

Looking at the results, we can first notice that there are specific tokenisation and tagging issues that result in inappropriate 'compound' forms. These, we can perhaps safely ignore, bearing in mind that the modals contained in them would normally contribute relatively little to the overall token counts of the genuine types.

The remaining examples can be investigated by clicking on the links referred to in the instructions, which I'd strongly advise you to do before making any conclusions about the comparison results, as at least the results for 's mainly yielded results that contained the contraction *Let's*, where the clitic isn't even a verb, but the reduced form of the pronoun *us*, apart from the fact that it's clearly debatable whether the reduced form of the auxiliary BE should ever be considered a modal. The latter also applies to *used* as part of the combination *used to*, where the sequence represents an alternative way of expressing past tense, rather than any form of modality. These two cases clearly represent issues related to automatic PoS tagging and the theoretical assumptions and rules behind it.

The next thing to observe in the lists is that we have what appear to be non-words, namely *ca*, *wo*, *sha*. However, if you call up the concordances for these, you'll soon find out that they represent the initial parts of the negative contractions *can't*, *won't*, and *shan't*, which have been separated from the negation 'clitics' in the tagging process and are being treated as individual tokens. Incidentally, the same thing also applies to BNCweb, due to the use of CLAWS in the tagging of both corpora.

Having identified these potential issues, we can now focus more on the actual genre similarities and differences. Here the most common ‘general’ modals like *can*, *will*, *would*, *may*, *must*, *might*, which tend to be more *epistemic* in nature, that is, frequently reflect instances of *hedging* or *stance*, are almost equally frequent in both sections. In contrast, though, the light-green highlighting in the table on the left-hand side clearly (along with its mauve counterpart on the right) indicates that the humanities writing contains more instances of *deontic* modals, that is, those that express *permission* or *obligation*, such as *shall*, *ought*, and *need*. The latter, which are sometimes also slightly more old-fashioned, for example, *shalt*, *shall*, or also to some extent *dare*, are apparently due to the fact that the humanities section contains religious texts that are highly formulaic in nature, as they contain biblical quotes of deontic nature, such as *thou shalt...*, etc., which are not present in the science/technology writing, which tends to be less ‘prescriptive’ in general. One particularly interesting type in the table in this context is *wilt*, which, on the surface, would appear to be an archaic form of WILL, so we’d expect to find more occurrences of it in the humanities texts. Upon closer inspection, however, it turns out that this type in most instances in both corpora in fact represents a typo, that is, the misspelt form of *will*, or in the science writing a mis-categorisation of the nominalised form of the verb *wilt* in two cases.

Sources and Further Reading

- Bondi, Marina & Scott, Mike. (Eds.). (2010). *Keyness in Texts*. Amsterdam: Benjamins.
- Hoffmann, Sebastian, Evert, Stefan, Smith, Nicholas, Lee, David, & Berglund Prytz, Ylva. (2008). *Corpus Linguistics with BNCweb – A Practical Guide*. Frankfurt: Peter Lang.
- Lee, David. (2002). Genres, Registers, Text Types, Domains and Styles: Clarifying the Concepts and Navigating a Path through the BNC Jungle. In Kettemann & Marko. (Eds.). (2002). *Teaching and Learning by Doing Corpus Analysis*. Proceedings of the Fourth International Conference on Teaching and Language Corpora, Graz 19–24 July, 2000. Amsterdam: Rodopi.
- Leech, Geoffrey, Rayson, Paul, & Wilson, Andrew. (2001). *Word Frequencies in Written and Spoken English*. London: Longman.
- Scott, Mike. (2010). Problems in Investigating Keyness, or Clearing the Undergrowth and Marking Out Trails... In Bondi & Scott. (Eds.). (2010). *Keyness in Texts*. Amsterdam: John Benjamins.

10

Exploring Words in Context

Although it's of course somewhat easier to confine our corpus-based investigation to single lexical items, single words and their frequencies aren't the only interesting things we may want to analyse with the help of corpora. This would be restricting ourselves to more or less a kind of lexical/semantic analysis that largely ignores the fact that words really only gain their 'true meanings' in context. Even if (sorted) concordance lines can already represent an extremely valuable asset in a teaching context because learners – as well as teachers – can investigate words as they're really used in authentic materials, such an analysis may be rather time-consuming. Furthermore, the fact that we may be able to find some patterns easily – and then perhaps focus on only those – may make us overlook the true flexibility inherent in language in 'playing' with these patterns to change their meaning in an appropriate and sometimes fairly subtle manner. These choices we have when using language can really only be investigated through finding ways of expressing this flexibility on the paradigmatic and syntagmatic axes in our corpus searches. On the other hand, if we move away from the analysis aspect for a moment and think about the teaching/learning side, we'll soon realise that finding out about groups of words that co-occur is of great importance in developing the learner's sense of idiomaticity.

10.1 Understanding Extended Units of Text

People who tend to ‘stray’ on the theoretical side of linguistics often appear inclined to assume that the unit of text we should deal with is the sentence, without necessarily being able to define what exactly this is. Is it simply everything from a capital letter to the next full stop, exclamation or question mark, provided that we even follow Western writing conventions? And how do we deal with embedded sentences, or co- and sub-ordination, reported speech, or parenthetical structures separated by hyphens or commas from the main proposition?

From these questions, it may be clear that it’s by no means obvious – or at least should not lightly be considered obvious – what the real unit in text could be. Certainly the ‘utterance’ *A cup of coffee(, please).*, given in response to the question *What would you like to drink?*, though definitely not grammatical in the traditional sense, should be considered a valid unit of information, even if it doesn’t contain a verb or a subject. So if it isn’t the sentence, then what should our ideal units in text be? Moving down to the level of *clauses/phrases* surely seems somewhat restrictive, even if the concept of phrase may sometimes be taken rather loosely simply as a (functional) combination of co-occurring words (cf. Nattinger & DeCarrico 1992). However, as we’ve seen in the preceding example, even a single noun phrase without the ‘politeness marker’ *please* may constitute a complete answer to a question, only that not all noun phrases actually do fulfil this function and can thus simply occur on their own and out of context.

The most likely candidate for a textual unit seems to be the c-unit, defined by Biber et al. (1999: p. 1070) as consisting of “clausal and non-clausal units [...] that [...] cannot be syntactically integrated with the elements that precede or follow them”. To develop a better sense of what this definition may in fact encompass, let’s do an exercise.

Exercise 68

Try to identify the c-units inside the following piece of dialogue by inserting whatever you think may be an appropriate punctuation mark between them in pencil. Pauses in this excerpt are indicated by #, and beginnings and ends of overlapping speech by opening ([) and closing (]) square brackets, respectively. A & B represent different speakers, where each speaker indication in the text is followed by a dot and the turn number.

Justify your choice each time, thinking about why you’ve chosen to break the unit at this point, and also why you’ve decided to use this particular punctuation mark. Was your decision based on your knowledge of writing conventions, syntactic, semantic, or pragmatic criteria, or maybe a mix of the former?

A.1: good afternoon Virgin train line Sandra speaking for which journey do you wish to purchase a ticket B.2: er Euston to Manchester please A.3: # now do you hold a current debit or credit card B.4: aha A.5: do you have a railcard at all B.6: no A.7: and how many people's travelling B.8: just one please A.9: and what date is it you're travelling B.10: the Saturday which is the third i think A.11: third of Oc... B.12: [October A.13: October] departing at what time from London Euston B.14: i'm not sure what time the trains are do you know A.15: well the trains run at 10 to the hour every hour B.16: every hour # and that'll be the 14 50 i think then A.17: arriving at 17 30 B.18: that's right yeah A.19: when are you returning B.20: er Monday A.21: departing at what time

As the above exercise will hopefully have shown you, while we may have relatively clear-cut syntactic categories/units in written language, spoken language, which is, after all, more dominant in our daily lives, is not as easily segmentable. Therefore, analysing and understanding it may require a rather different approach, as well as a need to re-think many things we may have learnt about the analysis of language in the past. What therefore seems to be even more important than recognising the c-unit as the correct unit is the fact that one should always be aware that there are certain boundaries in text that form natural borders between the units of text we may want to see as belonging together, an issue that gains in importance once we move away from looking at single words only.

10.2 Text Segmentation

As the previous exercise has shown, finding individual larger units of/in text is quite difficult to achieve. One might assume that punctuation, something we use all the time in order to delimit 'sentences' from one another, is fairly unambiguous, but, as you may also have noticed, once we start exploring all the varied functions the different punctuation marks may perform in a text, we may be quite surprised by their capacity for creating ambiguity. To illustrate this further, let's take a brief look at some of the general and special potential functions of different punctuation marks:

- full stops may indicate (declarative) 'sentences', enumerations (as in ordered lists, e.g. 1., 2., 3., or i., ii., iii., ...), abbreviations (*etc.*), act as decimal separators for numbers (e.g. 2.5; in English), or indicate dates or ordinal numbers (in German)
- commas in English may be used as thousand separators (e.g. 1,250) in numbers, whereas in continental European formats, they act as decimal separators (e.g. 3,5 signifying the same 3.5 in English)

- a colon may indicate either the beginning of some form of listing inside a sentence, the beginning of direct speech, or separate two numbers, such as a score/ratio (2:1) or time (6:45)
- even exclamation marks don't always indicate imperatives, but may occur in unusual contexts, e.g. as indicators for faculty in mathematics, as in *5!*
- some single quotation marks serve as apostrophes and may indicate different types of genuine contractions, abbreviations/shortenings involving deletions (e.g. *'cos*, *'coz*, *o'clock*), often to indicate non-standard features, such as *h-dropping* (e.g. *'ere* for *here*), or be used in scare quotes.

We already encountered some of these issues in Exercise 35, where, for instance, most of the Roman numerals in capitals were tagged incorrectly, due to not having been identified as numerals followed by a dot indicating their ordinal nature. Despite the potentially problematic practical issue of how to deal with the above options, it's important for us to try and interpret all the different uses of punctuation marks as accurately as possible if we do want to determine the correct unit for analysis, since straying too far may result in odd results when we want to handle the larger units of texts we're now going to discuss. Luckily for us, many of the potential difficulties shown above can be avoided if tokenisation is performed using appropriate regexes in most cases. We'll discuss options of how to actually mark (up) and distinguish larger units of text in Chapter 11, when we talk about annotation.

10.3 N-Grams, Word Clusters and Lexical Bundles

Moving beyond the level of single words, we usually start by looking at two-word-units (bi-grams), next three-word-units (tri-grams), and so on (see Section 10.7). When we talk of units larger than a single word, we can also refer to them collectively as *n-grams*, *word clusters*, *lexical phrases*, or *lexical bundles*, although these expressions carry slightly different connotations, depending on the context they're used in or the research conducted on them. The main thing that unifies them is that they all represent sequences of words that tend to occur directly in a row, that is, without any intervening elements between them.

N-grams may already constitute basic level collocations, such as *strong/black coffee*, etc., but certainly don't need to, as for example in the determiner-noun sequence *the green* as part of the noun phrase *the green house*. When dealing with semantically and syntactically meaningful n-grams, we need to be especially careful not to move across any c-unit boundaries because otherwise our combinations of words no longer make any sense. For example, if we have two consecutive units like *There was a green house. The house was in a terrible state.*, and we ignore the full stop at the end of the first unit, we may either produce a bigram *house the* or a tri-gram *house the house*, neither of which will really be a sensible combination, and both are therefore not worth interpreting in terms of their linguistic functions.

Incidentally, you may have noticed that the words in the n-grams shown above are all in lowercase, so that they don't really always represent the original forms. This is generally done in order to be able to group words with or without initial capitals together, just as we've seen for sorting above, and represents one of the 'shortcuts' in language processing that could potentially lead to errors of analysis if not borne in mind.

One further pitfall in analysing n-grams may also be to forget that there could actually be n-grams within n-grams. In other words, it often makes sense to look for the longest (sensible) string, which could always contain one that's one word shorter. Coming back to our example from above, the tri-gram definite noun phrase *the green house* also contains the bi-gram *green house*, which we may want to distinguish from the indefinite noun phrase *a green house*. Here, we can also begin to see the importance of retaining function words in our n-gram counts because otherwise we run the risk of losing potentially valuable distinctions, a feature that becomes even more important when we start looking at collocations or idioms, where, for example, we say that *somebody hit the* roof because they were angry, but not they *hit a* roof.

The term *cluster* can be seen as a kind of neutral 'umbrella term' for both n-gram collocations, that is, meaningful sequences, and those sequences of words that are less meaningful in terms of their syntactic or semantic functions, which are generally referred to as 'lexical bundles' (c.f. Biber et al. 1999: 990ff.) or 'lexical phrases'. Lexical bundles are prefabricated, often more formulaic, constructions that may in fact span across separate syntactic phrases, such as for example the beginnings of some questions like *would you like X*, *do you want X*, *can I have X*, certain types of hedging expressions, such as *I think that X*, *I believe that X*, to name but a few. Each time, the first part always represents a fixed, formulaic, construction, while the semantic flexibility of the overall clause is provided by the different types of *objects* required (represented as X in the examples). While the former examples of lexical bundles are relatively clearly structured, at other times, some bundles may appear less so, as in such combinations as *is based on* or *are likely to be* (Biber et al. 1999: 996), where, syntactically, we need to provide both a *subject* and a(n) object/complement, that is, two elements surrounding the bundle. To develop a better sense of what n-grams look like, and to what extent they may be meaningful, let's do another short exercise.

Exercise 69

To do this exercise, go to http://martinweisser.org/pract_cl/online_materials.html and select the link to 'Understanding Units in Text'. You should see the following text:

To develop a better understanding of what clusters may look like and how they're built up, let's take a look at an example of how n-gram lists are generated, based on the contents of this paragraph.

Click on the button that reads 'Click to generate n-gram list' and look through the results.

Next, change the number in the dropdown list to 3, 4, and 5, respectively, each time observing the output. Every time you do this, think about how meaningful/useful the resulting combinations may end up being.

The frequency of occurrence of the n-grams you'll have observed doing the above exercise can also be counted, just as in basic frequency lists. In this way, we can identify the most common clusters, and, for example, evaluate whether they may be semantically or pragmatically meaningful or useful to learn for L2 learners, or be indicative of a given textual domain, etc.

10.4 Exploring (Relatively) Fixed Sequences in BNCweb

BNCweb provides us with many different options for investigating words in context, involving a mix of the various options we've explored before, such as looking for only the (fixed) word forms themselves, this time in combination, enhancing this by restricting/expanding word forms in context by using wildcards or headword/lemma searches, as well as using regex quantification in conjunction with grouping. These queries are generally referred to as *phrase queries* in BNCweb. Additionally, we can also do contextual or so-called *proximity searches*, where it's possible to search for words that occur within certain textual units or within a certain number of words of one another, etc. The range of potential options is so great that we'll only be able to explore a small part of this to give you a 'taste' of what you can achieve if you explore them in more detail.

10.5 Simple, Sequential Collocations and Colligations

10.5.1 'Simple' collocations

We'll begin our exploration by looking at what I've termed 'simple' collocations, to contrast them with what would appear to be more complex phenomena we can only identify easily via statistical measures of co-occurrence. We'll discuss the other 'type' in more detail in Section 10.8. In general, as the name *collocation* implies, we're here dealing with a phenomenon that describes which words tend to occur in proximity (co + location) to one another because they have some kind of 'affinity' to, or 'affiliation' with, one another. To identify such affinities at the

most basic level, one can start by looking at the sequential co-occurrence of words in context. The differences in meaning lie in the ‘combinatorial options’ on the paradigmatic axes of the different words that co-occur with one another, either in relatively loose combinations, such as those involving nouns and their pre- or post-modifying adjective phrases, or in terms of relatively fixed combinations, like idioms or proverbs. In other words, the question here is to what extent the words in a particular context are actually (freely) exchangeable with other words.

In order to investigate the different options available for specifying variable sequence patterns, we’ll now do a series of exercises, beginning with an analysis of expressions of ‘voice quality’ in literary texts.

Exercise 70

Select ‘Written restrictions’ under ‘Query options’ on the main page.

To practise narrowing down your searches some more, restrict the query options for ‘Medium of Text’ to ‘Book’, the ‘Derived text type’ to ‘Fiction and verse’, and ‘Publication Date’ to between ‘1960-1974’.

Using your knowledge of wildcards in BNCweb, try to formulate a phrase query that captures the sequence *in a* + any intervening word + *voice*, starting with the most general option you can think of. Think of how to express ‘anything’.

Gradually refine the query to limit it to particular parts of speech occurring between the fixed parts. You can also use the simplified PoS tags after the underscore, and don’t even need to add any word or wildcard before it.

Extend the query to include two different intervening pre-modifiers co-ordinated by *and*.

As you’ll have seen, this approach already works quite nicely with relatively fixed phrase query patterns, such as the ones we tried above. However, sometimes you may also want to be able to restrict certain options for words, rather than using wildcards or PoS tags, or to make certain parts optional, so let’s just try a few relatively simple options for grouping, alternation and quantification:

Exercise 71

Try to re-write the last example to make the co-ordinated second modifier optional.

Extend the previous query to alternately look for *in* or *with* as the initial preposition. Can you observe any differences?

Next, try the somewhat more complex string `(in|for) (a|the) (long|short) (time|period)`.

Finally, try to create a pattern that will find one of the three prepositions *in*, *on*, and *at*, followed by up to three intervening words, and then a noun. Hint: You need to use exact regex quantification here. What do you think you can find with this and what could the results of this query be used for?

In addition to exploring the various phrasal constellations we've just investigated above, this type of flexibility also makes it possible for us to research idioms to some extent. When doing this, one important feature to verify is to see whether there are any words in such expressions that may potentially be replaceable by other words with similar meanings, for instance definite articles by indefinite ones, etc. If such variability is found, then the most interesting feature to explore is whether it's relatively free – something that is quite rare in idioms – or whether the idiom has deliberately been 'manipulated' by the speaker/writer in order to achieve a particular (pragmatic) effect.

10.5.2 Colligations

To illustrate *colligation*, that is, the co-occurrence ('linking') of specific word classes or lexical items with particular parts of speech, we can look at a special feature of BNCweb, which is the way the clitics in contractions are handled. As discussed before, contracted forms in English still tend to occur mainly in the spoken medium because the conventions of what is often deemed to be 'good writing style' continue largely to forbid their use in the written medium, despite the fact that this really doesn't make sense because contractions do in fact aid the natural reading flow. Because of this conventional restriction, we'll confine our searches to spoken parts of the BNC later.

The option to produce contractions in the strict grammatical sense, which is what we want to investigate here, rather than just any abbreviated word form, is limited to a closed set of words, belonging to few word classes, and co-occurring with an equally limited set of word classes. To be able to investigate this, we can divide our task into two separate stages. In the first, we'll use BNCweb to identify all possible forms of contractions in order not to miss any if we simply 'query our intuition'. In the second, we can then construct one or more appropriate search patterns that'll allow us to create concordances that illustrate which word classes co-occur with which type of construction.

Exercise 72

Open BNCweb and set the 'Restriction' option for a new query to 'Spoken restrictions'.

Think of possible options for contractions you know and define a wildcard pattern that'll find them. BNCweb defines certain clitics in contractions as separate words, so you need to bear that in mind.

Test your pattern and use the 'Frequency breakdown' option to identify unique 'types'. If you're not happy with the results, refine your pattern iteratively.

Investigate the individual clitics by accessing the concordances from the breakdown page and hovering over the hit to get the pop-up display of the PoS-tagged context. Opening the concordances in a new tab each time would definitely make sense here, as you then only need to close the tab again to return to the list of clitics. In addition to straightforward concordancing, though, also explore other ways of investigating the results, such as those that the sorting options/restrictions for the concordance lines offer.

The above exercise should have given you an insight into how particular clitics in contractions can combine with other word classes on the *syntagmatic axis*, although the options for each one of the parts of speech that can occur to the left of the clitic obviously represent items that are exchangeable on the *paradigmatic axis*.

10.5.3 Contextually constrained and proximity searches

Apart from the various options for using wildcards to specify optional or non-optional slots in a query, BNCweb also allows us to look for words/expressions in certain contexts or within a specific number of words of a search term. This, for example, enables us to look for search terms in specified environments, such as at the beginning or end of a paragraph, a sentence-like unit (*s-unit*), or a speaker turn in the spoken part. Furthermore, it also makes it possible to construct patterns where we're not exactly sure about the potential position of one term in relation to another, and want to be able to roughly investigate a range of possible options. What exactly this means will hopefully become clear soon.

The context options we have are indicated via XML tags (explained in more detail in Chapter 11), e.g. <p> for paragraph, <s> for s-units, and <u> for turns, etc., where 'u' apparently means 'utterance', although it may, strictly speaking, possibly consist of a number of these. Searches can either be defined to occur within paired tags, e.g. <u>(+)</u> to (theoretically) retrieve all individual turns from the spoken part, or as occurring at the beginning or end <u> + search term to retrieve items that occur in turn-initial position. Let's try this in a brief exercise.

Exercise 73

Using the pattern shown above, try to retrieve all individual speaker turns and get an overall impression of their length.

Look for the word *well*, first in turn-initial, then in turn-final, position and compare their meanings.

Proximity queries, in contrast, allow us to search for words within a certain span, left or right context, of one another, where the context may also go beyond a syntactic boundary. There are essentially two different options for doing proximity queries, one where we only say that a second term is allowed to occur within a certain number of words on either side of the search term, and the other where we specify clearly which side it's supposed to occur on. The first type is indicated by a number indicating the range surrounded by two angle brackets pointing away from the number on either side, e.g. <<3>>, and the other by indicating the position where the other term should occur through the direction in which the brackets are pointing, e.g. <<3<< for items that should occur to the left and >>3>> for items that follow on the right.

The results of proximity queries are a little more 'vague' than those of the other searches we've conducted before, especially because BNCweb only highlights the first term specified in the search. In order to interpret the results more easily, we may therefore need to sort the results according to the individual possible positions where the other term(s) may occur. Let's see what this might look like by trying to find phrasal or prepositional verbs by looking at all words that have prepositions – or rather, particles – occurring up to 3 words to their right.

Exercise 74

Based on the information given above, and your knowledge of how to look for simplified PoS categories, try to define a suitable pattern.

Once you get some results, try to explore them by sorting the results according to the three possible positions in turn and see whether you can identify some suitable candidates for phrasal or prepositional verbs.

10.6 Exploring Colligations in COCA

As we've just seen in our investigation of prepositions/particles, this PoS category may also enter into combinations with other prepositions/particles. In other

words, if we take colligation to be the co-occurrence of PoS categories, this particular class would colligate with itself in these cases. In the examples we saw earlier, though, the types we encountered usually contained one element that was part of a phrasal/prepositional verb construction. In another type, such combinations generally create a complex meaning, as in, for example, *up until* or *down below*, where the resulting meaning is a mix of the semantic properties of both elements. In some cases, though, especially in American English, we may encounter sequences of prepositions/particles that actually contain redundancy because one of the parts alone would already express the same meaning, such as in *off of*.

To investigate this feature, and to see whether such redundancy also exists in other combinations, we could now try to create lists of all multi-preposition sequences. Unfortunately, though, the BYU interface won't allow us to search for these, throwing the following error message "All of the "slots" in your multi-word search string occur more than 10,000,000 times in the corpus (e.g. [n*] [be], or [j*] [nn*]). Please re-do the query so that at least one of the slots has a frequency lower than 10,000,000". Thus, unlike in BNCweb, where we were able to run the query and then thin it down, there's in fact no way to run this query at all if we don't know how to restrict at least one element. The alternative presented for the COCA is to download n-gram frequency files by following a link presented further on in the error message. This obviously isn't as useful because there's no way to directly investigate the context by clicking a link whenever you find an interesting example, but of course better than nothing – or looking through hundreds of concordance lines generated for single prepositions in the hope of finding suitable combinations, which you could still do by using the KWIC display option and sorting according to first word on the right of the node.

Once you've downloaded the relevant offline file(s), you may now think that you can just load an n-gram file into AntConc and specify a suitable regex pattern to extract all multi-preposition sequences. Unfortunately, though, as useful as AntConc is for most purposes, since it's a stream-based concordancer it'll ignore all line breaks and match more than we want to see in our concordance, so using it here is not an option.

Theoretically, you could also import the data into a spreadsheet program and filter it according to the last two rows, but, in practice, this failed for me in both Excel and Calc as the number of rows (1,048,720 for bi-grams) unfortunately exceeds the rather high limit of 1,048,576 rows the spreadsheet can handle for each individual worksheet in it. In our particular case, filtering may still work if you cut and paste the final row into another worksheet (or delete it altogether) and insert a header row at the top of the data and then use the 'AutoFilter' function in both programs, because, essentially, all the rows that didn't get imported into the worksheet will contain combinations that start with the letter *z* and English has no preposition starting with *z*. To automatically filter here, you'd have to specify patterns where the PoS columns 'begin with' *i*.

However, the above clearly isn't a very 'clean' way of doing this, and also relatively complex. The only really 'clean' way to solve this using filtering would be to

import the data into a database, which is too advanced a topic here. A much simpler way to search the data, though, is to use a line-based concordancer, such as my own Simple Corpus Tool, where you can both look at the whole file easily once it's been loaded and also devise a suitable regex that will match two occurrences of the tag used for prepositions occurring in a row at the end of the line.

Exercise 75

Download at least the bigrams frequency list (case sensitive, with part of speech) from the BYU website at http://www.ngrams.info/download_coca.asp. Unzip it.

Download the Simple Corpus Tool from http://martinweisser.org/ling_soft.html#viewer and install or unzip the archive. On Linux or Mac OS, you can again use Wine to run the program.

Run the tool, press 'Ctrl + f' to open a single file, then select 'text files (*.txt)' from the file types list in the bottom right-hand corner, navigate to where you saved the frequency list, and open it. A hyperlink with the file path will appear in the left-hand frame.

Click on the hyperlink to open the file in the built-in editor. As it's a rather large file, it may unfortunately take a considerable time to load.

Look at the way the file is structured, think of a suitable regex that would help you match two prepositions in a row, and close the editor.

Change the 'Lines after' context to 0 on the tool bar.

Type the expression into the box next to the label 'Term 1' and press the enter key or click on .

Once the results have been loaded into the frame on the right-hand side, investigate them, always adjusting your regex, if necessary.

The previous exercise was mainly designed to show you that line-based concordancers can give you an alternative view of some types of corpus data, as well as obviously to help you get sensitised to the issue of multi-sequence prepositions a little further. However, to get some more experience in investigating some limited options for these in COCA, let's do another exercise there revolving around *off* as the first element in such a sequence. This will also allow us to see whether it may be involved in other redundant structures or belong to any of the other categories we explored before.

Exercise 76

Open the COCA interface again.

Run a search for *off* followed by any preposition.

Investigate the individual results in the KWIC view frame to see whether they may contain any redundancies, are part of phrasal/prepositional verbs, or form genuine multi-word prepositions.

To potentially distinguish between the three options if they should co-occur in the same KWIC window, you can also make use of the columns labelled A, B, or C for each concordance line. Clicking on each of these will highlight the particular row in a different colour and therefore allow you to visually distinguish between the samples at a glance.

In cases where the frequency of the colligates isn't too high, we can obviously also investigate colligations in a similar way as in BNCweb.

10.7 N-grams and Clusters in AntConc

We've already seen how relatively fixed, largely pre-defined sequences of words can be investigated using BNCweb, and to some extent COCA, but of course this only works if we want to explore interesting sequences that we're already aware of in more detail. In order to find new sequences of interest, such as lexical bundles or collocations that involve words occurring next to each other that we were unaware of before, we need to learn about some other features that are (currently) only available in AntConc.

In what follows, we'll initially experiment with features that help us to explore collocation (in a very loose sense) by working on a limited number of files in AntConc. You should, however, try to verify some of the results in BNCweb as well once we've identified particular patterns because the data you'll find there will potentially be much more representative.

The simplest way of investigating collocation in unknown sequences is to use a computer program to produce a series of n-grams or word clusters, as described in Exercise 69. The 'n' here theoretically means 'any number of', but usually n-grams are restricted to relatively short sequences, such as bi- or tri-grams, because finding n-grams can be very expensive computationally as long lists have to be kept in memory, and their size grows with increasing length of a text/corpus, and each increment of n. Thus, if you're working on a relatively sizable corpus, be prepared to wait for a few minutes for n-gram calculations to finish.

We can create two basic forms of n-grams/clusters in AntConc, one where we produce lists of sequences of all words in the text indiscriminately, which takes the longest, and another where we create clusters around a given search term. Obviously, the lists for the latter are much shorter and much more concise because they 'centre around' the search term. Let's try out both variants.

Exercise 77

Download the following texts from the Project Gutenberg website and put them in a folder called 'lit_selection': *The complete poems by Emily Dickinson*, *Frankenstein*, *Much Ado About Nothing*, *The Taming of the Shrew*, *The Tempest*, *Twelfth Night*.

Clean up the files as we practised before.

Start AntConc and open the folder containing these samples. Also move the full copy of the *Sherlock Holmes* text here. Please note that the selection we've now created is deliberately mixed, and in no way represents any balanced sample!

Make sure you set the 'Token (Word) Definition' in the 'Global Settings' to include hyphens and apostrophes.

Click on 'Clusters/N-Grams' or press **F4** to jump to the appropriate tab.

Select 'N-Grams' and note how the label next to this turns to 'N-Gram Size', too.

Set both the 'Min. Size' and 'Max. Size' under 'N-Gram Size' to 3 to produce only tri-grams.

Click **Start** and wait for AntConc to produce the list.

Select 'Sort by Range' from the 'Sort by' options and click **Sort**. This will sort the results based on how many of the documents in the corpus the n-gram occurs in, that is, the dispersion, in descending order.

Scroll through the list briefly and try to identify some interesting constructions. Also, pay attention to the overall number of n-gram types and tokens AntConc has identified.

So far, the composition of our small 'corpus' has been fairly heterogeneous, which has had a clear effect on our results in making it difficult to identify any interesting recurring 'themes'. However, if we change the composition of the corpus to make it more homogeneous, this ought to change very quickly.

Exercise 78

Select all non-Shakespeare texts in the 'Corpus Files' window and close them via the 'File' menu.

Create a new n-gram list and explore the results.

As you'll have seen from the first two exercises, n-gram analysis will often highlight typical grammatical or interactional structures, but possibly also some author- or period-specific language or recurrent semantic themes. The more limited the

topic contained in the corpus, the easier it'll become to identify the latter. So, for instance, if you'd run this same n-gram analysis on our Trains data that contains task-oriented dialogues of a highly limited nature, you'd have been able to identify some of the key terms and expressions used there much more easily through n-grams than through the single-word lists we analysed above.

If you're interested in identifying the 'neighbours' of individual specific words, that is, potential collocations or colligations, it's far more useful to be able to specify the search term and investigate its behaviour with regard to other words that may surround it. Let's try this, using the 'all-time favourite' *fair*.

Exercise 79

Add the remaining literary files again. It's probably easiest to close all files first and then open the directory containing all the literary samples again.

Uncheck the box for 'N-Grams', and type in *fair* as your search term.

Change the 'Cluster Size' option to 2 for the minimum and 3 for the maximum. Also ensure that the option to 'Treat all data as lowercase' in the relevant 'Tool Preferences' is unchecked and that the 'Sort by Freq' option is selected.

Click **Start**.

Scroll through the results, at least some of which should immediately make some sense to you.

Pay particular attention to the length of the clusters and their frequency. Can you identify a correlation?

Experiment with the two additional options for 'Search Term Position', 'On Left' and 'On Right'.

Try to identify in which way the different files making up the selection influence the meaning of the word *fair* in our results.

Having just covered the ways in which we can identify collocations 'manually', we can now turn towards exploring methods that involve statistics to try and 'predict' which words collocate more frequently and/or more strongly with one another.

10.8 Investigating Collocations Based on Statistical Measures in AntConc, BNCweb and COCA

10.8.1 Calculating collocations

As you'll have seen from our two previous exercises, n-grams/clusters may sometimes already provide us with a fairly good measure of collocation. However, they

most easily show us only words that co-occur in a relatively *fixed* and *linear order*, and frequently also include purely grammatical elements that may form a part of lexical bundles, but which may tell us rather little about the actual content of texts or the specific semantic content of the search terms we want to find collocations for. Another approach, which usually looks at a larger *span* of potentially *discontinuous* words, is that of calculating the degree of relatedness of words that occur near a particular *node* word. The following illustration shows how this relationship may be represented by referring to the negative (left-hand) or positive (right-hand) positions relative to the node.

It	is	only	<i>fair</i>	to	them	to
-3	-2	-1	node	+1	+2	+3

Figure 10.1 Illustration of a collocation span

Figure 10.1 only illustrates a span of 3 words to the left or right, but the calculations can be, and are commonly, also carried out taking larger spans into account. Due to the increasing number of calculations, though, this will usually also prolong calculation time. Calculating the degree of relatedness may be carried out using a variety of different statistical measures, most of which have advantages and disadvantages, the latter because they usually assume that words are randomly distributed, which is rarely the case in language (cf. Zipf 1949; Stubbs 1995). The basic mechanism is usually to compare the ratio of *observed frequency* (O), that is, how often two words have actually co-occurred within the span and relative to the overall size of the corpus, to their so-called *expected frequency* (E), that is, how often the two words have really occurred independently in the corpus, again relative to the size of the overall corpus. Therefore, the calculations are:

- for observed frequency (O), i.e. relative frequency of joint occurrence:
frequency of joint occurrence/number of words in corpus
- for expected frequency (E), i.e. relative frequency of node * relative frequency of collocate/square of corpus size:
frequency of node/number of words in corpus * frequency of collocate/number of words in corpus/square of corpus size.

Here, we'll only discuss two of the individual measures that are implemented in both AntConc and BNCweb, *MI* and the *t-score*, and briefly evaluate their relative merits. For a very interesting and much more in-depth discussion of these two measures, see Hunston (2002: 67ff.).

MI is a measure that tries to compare the ratio of O to E directly using the formula $\log_2 O/E$, and, according to Hunston "is a measure of strength of collocation" (2002: 73) that "tends to give information about its lexical behaviour, but particularly about the more idiomatic ('fixed') co-occurrences" (Hunston 2002: 74). However, one of its weaknesses is that it's fairly sensitive to corpus size, so that values/results can be relatively unreliable for small corpora. A general *cutoff*

point for MI values is 3, that is, any results with a lower value should probably be ignored.

The t-score, on the other hand, mainly takes into account the frequency of co-occurrence of node and collocate. It's therefore less sensitive to corpus size, and hence probably more reliable for smaller corpora. According to Stubbs (1995), it “provides confidence that the association between n[ode] and c[ollocate] is genuine”, and Hunston furthermore states that it “tends to give information about the grammatical behaviour of a word” (2002: 74). The general cut-off point here is 2.

BNCweb adds further options for calculating collocations. For a list of these, see the illustration of the dropdown menu in Figure 10.2:

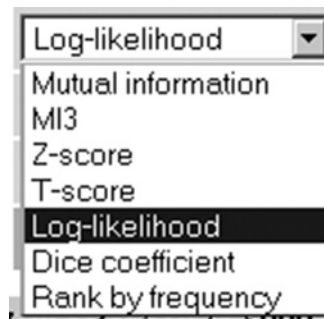


Figure 10.2 Options for statistical collocation measures in BNCweb

The relative merits and demerits of these measures are discussed extensively in Hoffmann et al. (2008: 149ff). The default for BNCweb is the LL (log-likelihood) ratio that we've previously encountered for keyword calculation.

10.8.2 Computing collocations in AntConc

Let's now test MI and the t-score in AntConc, using the complete set of literary data again, to see whether we may be able to observe some of the features discussed above.

Exercise 80

Open AntConc and select the folder containing the literary data, but restricting your selection to the Shakespeare plays again to make the selection more homogeneous.

Select the ‘Collocates’ tab, type in *fair* as your search term and set the ‘Min. Collocate Frequency’ option to 2, in order to avoid one-off constructions, also known as *singletons* or hapax legomena.

Set the ‘Window Span’ options to 4L and 4R respectively.

Make sure that you have the collocation measure in the ‘Collocates Preferences’ set to ‘MI’ initially and that the ‘Sort by Stat’ option is selected.

Start the analysis. If you haven’t previously created a word list, AntConc will inform you that it needs to do so and automatically invoke the corresponding tool.

Since we want to compare the results obtained through both measures, we first need to store the results of the MI analysis, then produce the list for using the ‘T-Score’, and save that, too. The best way to do this is to save both sets of result to text files as we did earlier for our concordancing results. However, just to get a quick impression, you can also use the quicker, easier and less permanent option produced by the [Clone Results](#) button, which will create the extra window containing the MI results that you could then keep open and then re-run the analysis using the ‘T-Score’ statistic to compare the results side by side.

Open the text files in your editor

Next, either simply separate the lines that contain scores above the cut-off points by spaces or some other marking from the rest of the results, or even delete all results below the cut-off points.

Finally, compare the results, and try to understand why the items appearing in the list may be considered collocates of *fair*, and, based on the information provided above, how they may be influenced by the statistical measures we used.

As you’ll hopefully have noticed, apart from simply giving us the option to look at/sort by the statistic or frequency of the collocates, AntConc also allows us to see or sort by whether the results occur on left- or right-hand side of the node, and with which frequency. I’ll leave it up to you to explore the usefulness of this feature.

10.8.3 Computing collocations in BNCweb

As we’ve seen earlier, BNCweb offers a much larger choice of collocation measures than AntConc, with its default set to log-likelihood. Unfortunately, though, there’s no facility for creating n-gram lists, probably because these could potentially get very large, working with such a big corpus. All collocational analyses have to be conducted by running a query on the node word first. Once you have the results of the query, the dropdown list on that page will then allow you to select

‘Collocations...’ to produce a list of collocates. Let’s try this using the same word we used above.

Exercise 81

Run a query on *fair*.

Select ‘Collocations...’ from the dropdown options list and click on **Go!**.

Investigate the settings on the next page briefly, see whether you might want to change any of them, and then click **Submit**.

Look through the list of collocates returned and identify the ones you’re already familiar with and the ones we haven’t encountered before. To get a concordance for the collocate, this time you need to follow the hyper-linked frequency in the column for ‘Observed collocate frequency’, while the link in the ‘Word’ column provides a breakdown of the statistics for the word form, including scores for the other statistical measures available and distribution of collocates within the given span.

Experiment a little with the options for changing the parameters to see whether you can understand what kind of an effect they have, and how the results will change/look different based on this.

When you try the option(s) for collocations on ‘POS-tags’, especially take a look at the results for punctuation to see whether these make sense to you.

If you want to compare results produced on a corpus analysis in AntConc with those drawn from a subcorpus of BNCweb, you should definitely select either the ‘Mutual Information’ or ‘T-score’ options there, depending on the size of your samples. However, Hunston (2002: 73) states that, while MI scores are directly comparable, this is not possible for raw T-scores and one should compare t-score ranks instead in this case.

10.8.4 Computing collocations in COCA

Calculating collocations in COCA works in a relatively similar way to that of BNCweb, so I’ll just introduce this briefly here without any additional exercise. Perhaps the main positive difference is that you can control some of the options directly from the display frame of the BYU interface, rather than having to run a query on the search term first. Thus, in order to investigate the collocates of *fair*, you simply type `fair` in the box next to ‘WORD(S)’, and then click on ‘COLLOCATES’. Doing so will add another textbox, as well as two dropdown lists to the interface, which will allow you to select the left and right span size. If you enter a specific word in the search box, the interface will allow you to calculate an MI score for the node and that particular collocate, while selecting from the

‘POS LIST’ options will allow you to look for colligations directly. If you don’t put anything in the textbox, a * will be added automatically when you run the query, and you essentially get the same effect as in BNCweb, where the query will simply find all potential collocates.

As with other queries in the interface, you can also restrict your collocations search to particular genre categories from here directly, and, of course, once you’ve calculated the results for the COCA, you’ll immediately be able to compare them to other corpora again.

Perhaps the only major limitation here, though, is that the BYU interface only provides a single collocation score measure, which is MI.

Solutions to/Comments on the Exercises

Exercise 68

For convenience sake, I’ve included one possible solution to the exercise below, where all the units, as I perceive them, are numbered, so that it’ll be easier to discuss them. As before, we won’t discuss the whole solution though, but I’ll simply get you started on the most relevant points.

A.1:

- 1 good afternoon.
- 2 Virgin train line.
- 3 Sandra speaking.
- 4 for which journey do you wish to purchase a ticket?

B.2:

- 5 er.
- 6 Euston to Manchester please!

A.3:

- 7 # now.
- 8 do you hold a current debit or credit card?

B.4:

- 9 aha.

A.5:

- 10 do you have a railcard at all?

B.6:

- 11 no.

A.7:

- 12 and how many people’s travelling?

B.8:

- 13 just one please!

- A.9:
14 and what date is it you're travelling?
- B.10:
15 the Saturday which is the third i think.
- A.11:
16 third of Oc...
- B.12:
17 [October.
- A.13:
18 October].
- 19 departing at what time from London Euston?
- B.14:
20 i'm not sure what time the trains are.
- 21 do you know?
- A.15:
22 well.
- 23 the trains run at 10 to the hour every hour.
- B.16:
24 every hour.
- 25 # and that'll be the 14 50 i think then?
- A.17:
26 arriving at 17 30?
- B.18:
27 that's right.
- 28 yeah
- A.19:
29 when are you returning?
- B.20:
30 er.
- 31 Monday.
- A.21:
32 departing at what time?

Splitting the text may initially have been hard for you, as the units of text in the extract don't conform with the normal rules of syntax in terms of complete clauses, etc., that you're probably used to from school, or maybe also badly written textbooks that represents spoken language as if it were the same as written language. For instance, the two short, verbless, units in the very beginning, *good afternoon* & *Virgin train line*, are nothing like the sentences we know from written grammar, although we're all used to hearing the initial greeting on a regular basis. Even so, it may be difficult to decide whether the punctuation mark we should

assign to the greeting formula ought to be a full stop or an exclamation mark. Clearly, it's formally an expression of well-wishing that is similar to a polite exclamation. On the other hand, we normally also associate exclamations with imperative structures, that is, commands, for which it may be more appropriate to use the exclamation mark to indicate their force, so in this case, I've opted for the full stop.

Finding a suitable punctuation mark may be easier for unit 2, which simply consists of a single noun phrase, and clearly is a kind of statement that just provides the name of a company. This, of course, grammatically doesn't correspond to a well-formed statement, as it's elliptical and doesn't even contain a verb. Its 'well-formed version' would probably be *This is Virgin train line*, with a subject, finite verb, and subject complement.

In contrast, unit 3 does at least contain a verb, even if it's non-finite, and we'd probably expect a completely well-formed version to look similar to unit 2, only that the verb in this case consists of a split construction of auxiliary + non-finite ing-form, that is, *This is Sandra speaking*.

Unit 4 is obviously a question, so you'll probably have chosen a question mark like me. Normative grammarians may frown upon the preposition preceding the wh-word, though.

My having split off the filled pause/hesitation marker *er* as unit 5 may come as a surprise to you, but as it occurs at the beginning of a turn and has the function of indicating hesitation on the part of the speaker, this is justifiable, although, admittedly, not everyone involved in transcription may agree with this practice, and some people might prefer to integrate it with what I've numbered unit 6, which is clearly a directive, as indicated by *please*, that is, a mild form of command that justifies the use of an exclamation mark.

In turn 3 by speaker A, unit 7 is a discourse marker that indicates that a new stage in the dialogue is beginning. As such, it's prefacing the yes/no question in unit 8, and clearly has an independent function that justifies treating it as a separate unit.

Unit 9 is a short acknowledgement, the equivalent of a yes-answer, so clearly a special form of statement, which warrants the use of the full stop here. The same goes for the negative response *no* in unit 11.

As you'll have seen from the above discussion, throughout the extract we encounter a number of units that'll look familiar to us from traditional grammar, but also a considerable quantity of elliptical structures that range from single words as signals of acknowledgement, agreement, negation, or hesitation, to more phrase-like structures that may even contain incomplete words, such as in unit 16, where speaker A begins to ask for a confirmation of what she assumes to be the month of travelling. However, before she can actually complete her question, speaker B completes this name for her collaboratively. At other times, such as in unit 24, a speaker may simply repeat part of the information provided by the previous speaker, possibly in order to confirm that they've heard and absorbed the information correctly.

Exercise 69

As you should be able to see from experimenting on the web page, when generating n-gram lists, each time a window of n words is selected from the text, then the position shifted to the right by one, extracting the next n words, and so forth. This seems to be a relatively straightforward process, involving no particular issues. However, if you look closely at the example sentence, you'll realise that it actually contains a number of clauses, with the main (finite) one being "let's take a look at an example of how n-gram lists are generated". Now, based on the experience developed in Exercise 68, you'd probably want to assume that this, as well as the other two, dependent clauses, are also independent c-units, so that, technically, we're crossing over a c-unit boundary as soon as we create the bi-grams *up let* and *generated based*. And when crossing over a boundary for a new unit of sense, we'd probably want to avoid assuming that there's a close association between the final word of the first unit and the first one in the second unit, although, of course, there may be some cohesion between some units, for example, when the first unit ends in an object NP and the second unit picks this up as a subject in the form of a pronoun. At any rate, even in the latter case, we'd have an association between the NP and the pronoun, but not necessarily the final word in the NP and the beginning of the next unit. Unfortunately, at the moment, most corpora generally don't indicate c-unit boundaries, so that, at best, we'll generally be able to analyse texts according to the 'sentences' marked up for them. An exception in this respect is the SPAADIA corpus (see Leech & Weisser 2013).

Exercise 70

Selecting the restrictions for this exercise should be quite straightforward again, so I won't say anything further about this here.

The most basic form of this query would be `in a * voice`, where the * stands for any word or character. However, this wouldn't catch any intervening words that start with a vowel letter, so we'd have to re-write this slightly to make it `in a[n,] * voice` to include the optional *n*.

As the whole phrase is a prepositional phrase, in its basic form the intervening parts of speech could minimally be an adjective or an adjective pre-modified by an adverb, so we could extend the query by either saying `in a[n,] _{A} voice` or `in a[n,] (_{ADV})? _{A} voice`, where the element for the adverb slot has been made optional by including it in round brackets and using the ? quantifier. This should result in 899 hits which, if you consult the frequency breakdown, will tell you that *in a low voice* is by far the most frequently used expression, accounting for 157 hits (17.46%), while the frequency for the next two expressions, both with 36 hits (4%), *in a small voice* and *in a loud voice*, is already considerably lower. We can therefore conclude from this that *low* collocates with *voice* most strongly in descriptions of voice quality.

Extending the query to again include optional adverb results in `in a[n,] (_{ADV})? _{A} and (_{ADV})? _{A} voice`. This query, involving a phraseological pattern with more complex and longer pre-modification, now only yields 16 hits, none of which are repeated. This goes to show that, the more specific we want to be in characterising different features of NPs, the less we can rely on ‘ready-made’ chunks, such as the ones collocations offer us.

Exercise 71

This exercise draws on much of the knowledge you gained in previous sections on BNCweb, as well as regex quantification, but extends the scope of our queries to considerably more variable and longer phraseological units. Re-writing the last example from the previous exercise should be fairly easy, only that now the optional element contains not just a single word, but an AdjP that is (still) optionally pre-modified by an adverb, and which should be written like this: `in a[n,] (_{ADV})? _{A} (and (_{ADV})? _{A})? voice`.

The extended query string including *with* along with *in* should be `(in|with) a[n,] (_{ADV})? _{A} (and (_{ADV})? _{A})? voice`. When you do a frequency breakdown of this, you should be able to notice that there are far fewer examples starting in *with*, and that most of those tend to describe fixed vocal characteristics of people, rather than specific ways in which people are reported to be saying things.

In investigating expressions related to durations, using the query string `(in|for) (a|the) (long|short) (time|period)`, close observation should show you that, while both constructions have relatively similar meaning, those with *for* are used much more frequently. This may indicate a slightly stronger preference for such expressions to collocate with *for*, even though this would of course require more in-depth investigation, including potentially further expansion of the nouns expressing durations, such as for example *duration* itself, which I leave up to you for additional practice.

The final part of the exercise focusses more on identifying general differences between the three prepositions *at*, *in*, and *on* regarding their usage. The string `(at|in|on) (*){1,3} _{N}` finds a very large number of mainly prepositional phrases based on the three prepositions. To be able to do a frequency breakdown here, you’ll need to reduce the number of examples, which is referred to as ‘thinning’ in BNCweb, as the interface won’t allow you to work with the whole set of data, due to its size, and the computation time and processing power involved. You can reduce the number of results by selecting the ‘Thin...’ option from the dropdown menu after running a query. In most cases, you’ll probably want to keep the default setting set to ‘random (selection is reproducible)’ and then go for a percentage of 5%–10% of the data, at least for the above exercises. Thinning down the results to a manageable amount (randomly) and extracting suitable items from them can then yield useful samples that may for example be used to investigate differences in their usage for creating improved teaching

materials, based on real-life data, or directly in the classroom for awareness-raising activities.

Exercise 72

For this exercise, you may again need to thin the query; 10% should do. As the thinning is random, of course the results you obtain may look somewhat different from the ones I'll describe, as well as from those of your classmates, if you're doing this exercise in a classroom.

Your attempts at creating a wildcard query will hopefully have led you to define a contraction as anything that contains an apostrophe followed by a one or more characters, i.e. *' +, or possibly also *'*. A more general wildcard pattern that may only be looking for '+' (or '*') will fail here because it can only identify word tokens that actually consist of the apostrophe followed by any number of characters, but will not include anything preceding the apostrophe, due to restrictions of the wildcard syntax. It will thus, for instance, find forms like 's (representing multiple PoS) but fail to identify any instances of the negation particle (tagged XX0), as doing a 'Frequency breakdown of word and tag combinations' will easily confirm. Likewise, if you postulate that something has to occur in front of the apostrophe by using +' +, you won't find any of the tokens where the clitic is treated as being independent.

Once you've run the appropriate query, and switched to the word + tag breakdown, you should at least be able to identify the following options for PoS categories involved in genuine contractions as the right-hand element, discarding other abbreviated forms or representations of 'non-standard' speech:

- be ('s_VBZ, 'm_VBB, 're_VBB), have ('s_VHZ, 'd_VHD, 've_VHI), modals ('ll_VM0, 'd_VM0)
- negation (XX0)
- personal pronouns ('em_PNP)
- possessive ('_POS)

Strangely enough, you'll probably also find occurrences of *let's* tagged as modal verbs, where apparently the tagging routine has failed to tokenise properly into verb + pronoun clitic.

Now that we've identified the right-hand elements, all that remains to be done is to investigate the left-hand options. Starting with 's_VBZ from the top, and sorting the concordance according to 'Position: lLeft', we can easily see that this particular clitic, representing the 3rd person singular contraction form of BE, colligates with nouns or other constructs that can constitute NPs, such as pronouns, where the first occurrences you'll probably find of the latter will not be the definite singular 3rd person subject pronouns you may have expected (i.e. *he, she, it*) but instead the indefinite ones *everybody, everyone, and everything*. Along with

these genuine colligations, you'll unfortunately also find a few erroneous possessive markers or even cases where the transcribers erroneously inserted apostrophes in places where they don't belong, such as, for example, for *gets*, which occurred transcribed as **get's* three times in my results. Please note, though, that this transcription error was also not caught by the CLAWS grammar, which should have rejected the combination of tags VVB + VBZ where the right-hand element is a clitic, which should only be allowed to occur in very few cases in spoken language. What exactly those are, I'll leave up to you to explore again, as you should know how to do this by now ☺

As going through the rest of this particular list is a bit tedious and time-consuming once you reach colligations with *he*, you'll probably be tempted to assume that, from now on, you'll most likely only find pronouns, anyway, so you might be tempted to stop here, even though you shouldn't really. However, because we can assume that pronouns will make up a major part of the colligations, we can use an alternative way of investigating those colligations other than pronouns. To do so, we can use the 'Tag restriction:' feature on the sort page, select 'any pronoun' from the dropdown list, and then check the box next to 'exclude'. This would then at least have shortened the list by removing everything tagged as a pronoun, although you'd still have to click your way through lots of occurrences of noun colligations and other noun-like tags, such as those for cardinal numbers, which may act like nouns syntactically. The ideal situation would of course be that we could exclude tags based on regexes, but unfortunately BNCweb currently doesn't offer this functionality.

If we start by excluding pronouns, the first word class we find when we start ignoring everything 'nouny' is adverbs (although some of them may have been mis-tagged). Having realised that some adverbs may in fact be colligates of this clitic, we can now go the other way and, instead of excluding a particular word class, restrict our sorting to display only adverbs (using 'any adverb') to the left of the clitic. One of the possible adverbs you'll probably find here is *else* (tagged AV0), although this may be a doubtful classification, as it generally appears to have a nouny function again before clitics because it normally occurs with indefinite pronouns like *someone*, and the two elements together constitute an NP. Furthermore, you'll find deictic adverbs like *here*, *there*, *today*, *tomorrow*, and *tonight* (AV0), the cohesive conjunction *so*, also labelled as a general adverb (!), and the question words *how*, *when*, *where*, *why* (tagged AVQ). Regarding the latter, you may now, quite rightly, expect to find at least the other two question words *what* and *who* in the same concordance list, as they can certainly be followed by the same clitic, but, due to the tagging rules of CLAWS, these are in fact classified in different ways from the other question words. As before, I'll leave it up to you to find out how and also reflect on why this was done, using the query string (what | who) 's...

Going through the rest of the items of clitics that are parts of genuine contraction should now be relatively straightforward, as I've already pointed out the various issues you may encounter, and ways of efficiently identifying particular

occurrences of lexical items for a given PoS category. The only interesting thing to perhaps point out to you here regarding contractions involving *n't* is the high frequency of occurrences of *ain't* you'll encounter, which clearly reflects the non-standard language occurring in the spoken sections of the BNC. What's also interesting here, but from a tagging point-of-view, rather than a sociolinguistic one, is that all left-hand elements of this contraction have been tagged as 'unclear' (UNC), probably due to the fact that the CLAWS grammar couldn't disambiguate between the different forms it may represent in standard language, such as *isn't*, *hasn't*, etc.

As with Exercise 71, the usefulness of this exercise doesn't only lie in the area of pure linguistics research, although it may have highlighted a few potential issues in the realm of grammar-writing for general and tagging purposes or PoS classification, but may of course again also yield interesting examples for a) raising language awareness in more advanced students who can use concordancers on their own, b) producing pre-prepared filtered concordance lines for use in the classroom with less advanced students, and c), again, providing diverse real-life materials for textbooks.

Exercise 73

The first part of the exercise shouldn't prove particularly difficult, at least not as far as retrieving the units is concerned. However, if you switch the display from random to corpus order, you'll notice that, apparently, not all u-units are in fact retrieved because the KWIC display actually starts with the second unit. Therefore, the number of hits (699,885) reported in the info bar at the top is most likely unreliable. However, for the purpose of this part of the exercise, this isn't really relevant because you can still get a rough impression of the length of the individual turns, especially if you switch from 'KWIC view' to 'Sentence View'. Here, what I specifically wanted you to notice is that, unlike in a paragraph in written language, the number of units in a speaker turn can be quite low, and turns often consist of individual c-units, which, again, may only consist of individual words.

Bearing in mind that some occurrences of *well* may represent both the beginning and end of a term at the same time, that is, in cases where *well* constitutes the only c-unit in a turn, you should be able to observe that the word form can have two rather distinct meanings. In initial position, the meaning and function is that of a *discourse marker* (DM), and either tends to indicate the beginning of a new sequence in the spoken interaction or the beginning of a response on the part of one speaker, where that particular speaker wants to preface this response by indicating that they don't agree fully with what has been said before. In final position, in contrast, *well* represents the genuine adverb counterpart of *good* or possibly also part of the MWU *as well*. Unfortunately, the CLAWS tagging simply 'lumps' all these meanings together, using a single general adverb tag for all of them, which again proves the point that taggers like CLAWS are really optimised

for written language, but often still have a number of problems when it comes to dealing with spoken language appropriately.

Exercise 74

The basic query that allows us to look for the relevant patterns is `_ {v} >>3>> _ {PREP}`. As this'll produce a rather large amount of hits, though, you'll need to thin it down in order to be able to perform any sorting operations. To get a wide range of examples, I'd suggest that, this time, you thin to the maximum allowed number of hits, 250,000.

Starting our investigation with particles occurring one position to the right of the node verb form, we can choose the position '1 Right' and set the restriction to 'any preposition'. This will immediately return some fixed collocations with verbs such as *think*, *talk*, *know*, *say*, or *ask* in combination with *about*. Please note that, essentially, all these verbs are semantically relatively empty, that is, would largely be considered *de-lexicalised verbs*, and often only acquire a specific meaning in conjunction with such a preposition. Of course, you may also encounter atypical combinations, such as *consider about* in "PRACTICAL POINTS TO CONSIDER ABOUT_PRP-AVP A HOME BURIAL" (ACM 711), as is in the nature of corpus data. As you'll see from the typical examples here, the verbs involved in this construction are predominantly verbs of saying, perception, and mental reflection. This would probably be even easier to see if BNCweb allowed us to do a secondary sort, but unfortunately, it doesn't do so. In rarer cases, we may also find even more idiomatic collocations that do involve verbs of action, as in the example "Anne said that she would like to come up with him, potter about_PRP Dundee while he's having his medical [pause] then they will come" (KP8 2581), where of course *about* has the meaning of 'around', rather than specifying an object or expressing an approximate value via a prepositional phrase. The same also goes for examples involving *look about*, etc. As it might get a little tedious again looking at endless examples of *about*, you can also skip ahead a few pages by typing in a page number next to the **Show Page:** button in the navigation bar above, and clicking the button. From about page 24 onwards, I managed to find examples involving *above* this way, but remember, as we've thinned the examples randomly, your frequency distribution may be somewhat different. For *above*, one thing that again becomes immediately clear is that certain verbs of 'forced' or independent 'movement', such as *rise*, *raise*, *increase*, clearly collocate with this in establishing a sense of directionality, while others, such as *whirl*, *billow*, *tower*, or *perch* signal descriptions of positions. I'll leave the remainder of this sub-part up to you to complete and learn from, or use as a basis for creating suitable materials for teaching.

While investigating position '2 Right' with the same restriction, you'll have to ignore many examples where the node verb form may in fact be an auxiliary, followed by a finite verb and then the preposition, as these examples are really only similar to what we just investigated. At other times, they may illustrate collocations that are similar again, but allow for intervening objects (e.g. in "that man

in London you *told us about_PRP*"; HA7 3272), adverbs (e.g. in "It also follows that one cannot *talk specifically about_PRP* 'marked theme' in FSP theory"; FRL 1446) or adjective complements (e.g. in "But I have nothing to *feel guilty about_PRP-AVPP*"; JXS 1530) to occur in between the verb form and the preposition. Things do get even more interesting in terms of the collocations when we start encountering verb + multi-preposition constructions, as in, for example, "That is what Robert and my agent *are on about_PRP-AVP*" (ADA 596), or "What're those dogs *goin' on about_PRP?*" (J13 1864), "What are you so *fed up about_PRP?*" (C8E 655). As before, you'll probably have to skip ahead quite a few pages until you find examples of other prepositions, or try to use the 'Starting with letter:' option to isolate a few more interesting examples other than those where the preposition starts with *a*. Alternatively, you could of course also thin the original query down to a manageable size and hope that the random selection won't give you too many of the most frequently occurring prepositions.

Investigating the '3 Right' option will be even more time-consuming, due to the relatively longer span, which leaves even more room for options of syntactic constructions occurring in between. Nevertheless, in some cases, it may well be necessary and rewarding to carry out in-depth investigations of this kind as it's well known that particles may occur that far apart from the verbs they go with. However, we'll soon investigate other ways that may make it easier to check these co-occurrences. And, of course, you could always also use smaller corpora (or parts of the BNC itself) and investigate these in a concordancer like AntConc, where the sorting options make things easier for you.

Exercise 75

Downloading and unzipping the frequency files and the program should hopefully have presented no problem, as should loading the file in the built-in editor. As you may have noticed, this editor is really not optimised for handling large files, so to just view the frequency list without the ability to concordance on it, a dedicated editor, such as Notepad++, would be far better. On the other hand, using such an editor would only allow you to search through, but not concordance, on the file.

The regex that should allow you to extract only occurrences of two prepositions should be something like `\bi.{1,3}\b\s{i.{1,3}}$`. If we were sure that we could exclude ditto tags, we could also have used `\bi[^d]{1,3}\b\s{i.{1,3}}$`, but in our case, this would actually have excluded *off of*, which happens to be marked with a ditto tag.

I'll leave it up to you to find some interesting combinations, but remember, whenever you identify something here that may be interesting, you need to go back into the BYU interface and investigate it there. And if you find that it's too difficult to spot potential candidates for investigation, due to the limited context, you can of course also download the files for larger n-gram combinations (up to 5) and adjust your regex accordingly.

Exercise 76

Although this exercise will quickly present you with a wide range of choices representing combinations of *off* followed by another preposition, provided that you used the right query, `off [i*]`, very few of these will actually contain any examples of the kind of redundancy described in the exercise above. Probably the only real examples are *off of*, as well as *off off*, where perhaps some of the examples are used jokingly. However, not having found many other examples here still doesn't invalidate the observation that this feature does occur in American English, as we've shown the redundancy to exist. And, of course, it may not be limited to instances involving *off*. On top of that, the exercise has helped us to distinguish more between the different types of multi-item prepositions, which has hopefully helped you understand better how they work.

One thing that still remains to be done, though, is to verify the question whether this may be a typical feature of American English only. Obviously, as one of the strengths of the BYU interface is that it allows us to compare different corpora, this is something that can easily be tested by running a comparison with the BNC, where it turns out that the combination *off of* is nearly 4.5 times as frequent in the COCA, while the frequencies for *off off* are too small for a genuine comparison.

Exercise 77

As you'll probably notice while doing this exercise, n-gram lists, although they can produce very interesting results, may take a while to interpret properly. Furthermore, instead of revealing interesting combinations of *content words*, you'll often find more grammatical constructions or combinations of *function + content words*, especially if the corpus is not very homogeneous, as in our case. With this particular selection, what you'll probably be able to identify at best is some recurring grammatical constructions, such as the beginnings of statements or questions, as well as some occurrences of 'archaic'/'literary' usages. For instance, there's a relative overabundance of constructions with *shall*, or occurrences of *I will not* instead of *I won't*, where the latter would be much more common in modern drama/literature. Other things you can identify are *I pray you/thee* vs. *I prithee*, instances of reported speech (*said X*, etc.), as well as a few proper names, such as (*Mr.*) *Sherlock Holmes*, etc. The overall number of types (194,570, at least based on my token definition) is also fairly high, reflecting the variability of expressions.

Exercise 78

The results of the new trigram list should now have become much easier to interpret, and you should be able to identify many more archaic usages in terms of expressions, such as *by your leave* or *didst thou not*, absence of contractions, etc., much more easily than before. Other things that will crop up frequently are bits of

information related to the plays that form part of this ‘corpus’, such as references to the author, to acts and scenes within the plays.

Exercise 79

This exercise, as deceptively simple as it may look, can actually demonstrate a number of important features of how to identify and analyse collocations to you. First of all, keeping the ‘Search Term Position’ set to ‘On Left’ will help to mainly identify noun or NP collocates, while changing this to ‘On Right’ reveals other types of combinatorial options, such as combinations of determiners, possessive pronouns, intensifying adverbs, etc. with the node.

In terms of the length of clusters, it should immediately become obvious that, as soon as we move from bi-grams to tri-grams, the frequency drops, as does the range, in fact. This is not surprising because the longer a cluster gets, the more specialised its composition, along with its meaning, also gets.

In terms of the composition of the corpus and its relation to the individual clusters, you’ll hopefully notice very quickly that, with collocates occurring on the right, our results contain a relatively high number of proper names. This is due to the imbalance in the data, where the three Shakespeare texts still clearly illustrate the Elizabethan concept of ‘fair’ predominantly related to the appearance and beauty ideal of women, where light-coloured skin was not only one of the most valued properties, but also associated with good character, honesty, and virtue. This feature still remains ‘alive’ in modern English, but in a highly limited way, where *fair*, related to appearance, still collocates with nouns like *hair* and *skin*. You can see the difference between the texts from the Elizabethan period and the somewhat more modern other texts more clearly if you remove all the Shakespeare texts from the selection. Then, even though the ideas of beauty and virtue will still occur in a number of the examples, the modern meaning of ‘fair’ as ‘just and equal’, ‘unbiased’, or ‘considerable’ will become more and more apparent, even despite the fact that much of the language in the remaining texts is still more old-fashioned and/or poetic.

Exercise 80

In doing this exercise, it’s perhaps even more important to follow all the steps carefully than before because relatively minor changes may well influence the results. As we saw earlier, the original literary selection was (deliberately) very heterogeneous, which did allow us to identify features related to language change nicely. However, this very heterogeneity could skew the results of our general collocational statistics rather strongly, especially in such a small corpus. Thus, it’s important that we try and compare ‘like with like’ in restricting the selection to a single author/period now to get some more specific information about the collocations that are more characteristic of the actual texts.

Also, if you forget to set the minimum collocate frequency to 2, you may get a number of results related to singletons, which we clearly want to avoid, since

generalising from single examples has no genuine predictive or explanatory value. Yet another potential source of ‘error’, or rather confusion, could be that you might somehow inadvertently have set the sorting option to ‘Sort by Freq’. This is, of course, one valid way of looking at the data, but unfortunately doesn’t illustrate the differences in the use of statistics very clearly. This is because the words then wouldn’t appear ranked according to their relative importance according to the statistic used, but we’d have to look very carefully at the values in the ‘Stat’ column to compare the difference and judge whether it may be significant.

If we trust the cut-off points given in the literature, 3 for MI and 2 for the t-score, we should in fact prune the lists before comparison, which leaves us with a list of around 60 collocate candidates for the former, but only 15 for the latter, although of course the same number of collocates originally gets identified by AntConc since they all occur within our span. Perhaps the very first thing to notice here is that, as we saw in the quote from Hunston earlier, the t-score does indeed seem to provide information about the “grammatical behaviour” because most of the words in the pruned list are definitely function words. In contrast, the majority of words MI identifies are content words, such as proper names, common nouns, adjectives and verbs. Here, especially the high number of proper nouns is, as we saw earlier, highly characteristic of the Elizabethan English used in plays such as Shakespeare’s. And, if we try to interpret the MI list further, most of the words that occur at the top turn out to be words that either refer to or characterise people or general nouns – in other words, word classes that form parts of noun phrases. Having identified the potential significance of the MI results for our data, we now need to think about how the t-score results may relate to these. Starting from the very top, with *and*, we can observe that some of the instances are simply due to issues of cohesion, that is, they are co-ordinating conjunctions that represent the beginnings of new clauses, in which case it would be nice if we had some way of preventing collocations from being identified across syntactic unit boundaries to improve the precision of our analyses (see Exercise 81) because those aren’t really collocates at all. In a number of instances, though, *and* co-ordinates adjectives in predicative structures that, again, help to characterise people, for example in *fair and virtuous*, which actually occurs twice in the data, each time characterising a different woman, Katherina or Bianca, or *fair and fresh and sweet*, which once more represents an epithet of the former. The definite and indefinite determiners, as well as to some extent the possessive pronouns *my* and *thy*, again sometimes form parts of characterising noun phrases, where *fair* is used attributively in these cases. Therefore, essentially, all the cases of collocations identified through the t-score stat that I’ve just discussed, strictly speaking, represent cases of colligation.

Exercise 81

When you look at the ‘Collocation Settings’ page, you’ll notice that the option to ‘Calculate over sentence boundaries’ is, very sensibly, set to ‘No’ by default. In BNCweb, even being able to do this is only possible because the whole corpus is

marked up for s-units. As pointed out before, though, in very rare cases where we want to investigate cohesion, looking across boundaries may be justifiable. Unfortunately, when working with most corpora, and in most concordance programs so far, the option for handling data involving a measure of the syntactic units they occur in is still absent, something we just saw in the Exercise 80. In addition, as Hoffmann et al. (2008: 142) also state, it's generally advisable to treat word forms separately in collocation analysis, because the use of a different form generally involves a specific function, too, so here it's again best to stick to the default of 'No'. And since we ideally want to observe results that are comparable to what we generated before, setting the 'Maximum window span' to 4 would also make sense here.

Starting from the top collocates, you should be able to see that *enough*, *trading*, *share*, *amount*, *fair*, *play*, *say*, *trial*, etc. all relate to the meaning of 'just and equal' we identified before as being perhaps the most common contemporary meaning associated with the word, whereas the one associated with *isle* still expresses the idea of beauty left over from Elizabethan times. The collocate *hair* is also already known to us, and related to the older meaning, but *antiques*, *Hannover*, *trade*, *Frankfurt*, and *CeBIT* (some already corrected in capitalisation here) all refer to a more modern institution, that of the 'trade fair', while *Vanity* of course is part of the title of the book *Vanity Fair* by Thackeray, where the meaning of *fair* represents a slightly older variant of the modern 'trade fair' and probably closer to the modern meaning of 'fun fair'.

Experimenting with the options should allow you to find that BNCweb also lets you list the PoS categories that collocate with the node by selecting 'collocations on POS-tags' next to 'Information'. In other words, what we're really looking into here is a form of colligation, which, however, is probably a little too fine-grained for most purposes. Changing the minimum frequency options for either the co-occurrence of node and collocate, or the collocate only, theoretically makes it possible to thin down the results further, but in practice probably only works for relatively high values, and may thus impose artificial restrictions. The 'Filter results by' options allow you to either quickly select a word form from the list of collocates produced, or even calculate a statistic for a form that hasn't been identified during the analysis. In addition, you can also restrict the output to a given PoS tag category to either disambiguate a grammatically polysemous word form provided on the left, or to filter the list of collocates by PoS.

When changing the options for the statistical measure, you'll probably have observed some similarities and differences across the measures. While LL and most of the other statistics tend to emphasise the content word collocations in our example, two of the measures, 'T-Score' and 'MI3' sometimes rank function words a little higher, while a raw frequency ranking places predominantly function words at the top of the list, as is to be expected.

The final part of this exercise is designed to raise your awareness of one of the issues in the handling of types and tokens in the BNCweb interface, as well as the underlying CQP architecture. Here, sadly, the designers of the architecture have

introduced a serious flaw in the system that may well affect the overall calculations of the collocation statistics very strongly, which is to treat punctuation tokens (and their types) as equivalent to words. In doing so, they've effectively introduced two sources of error that affect different parts of the interface, and hence the quality of the results. At the level of collocations, the very fact that punctuation occurs with a relatively high frequency in any orthographically transcribed corpus like the BNC almost guarantees that it'll be treated as collocating with genuine word types, something that simply doesn't make sense because the semantics and pragmatics of punctuation are very different from, and completely incomparable to, those of ordinary words. At the same time, the high frequency of punctuation tokens will affect the calculations of relative and normed reported frequencies throughout the whole corpus, which will again have an effect on the calculations for collocations, too. Unfortunately, even pointing out this issue to the designers/developers of CQP so far hasn't led to any changes in this respect ☒

Sources and Further Reading

- Barnbrook, Geoffrey. (1996). *Language and Computers: A Practical Introduction to the Computer Analysis of Language*. Edinburgh: EUP.
- Hoffmann, Sebastian, Evert, Stefan, Smith, Nicholas, Lee, David, & Berglund Prytz, Ylva. (2008). *Corpus Linguistics with BNCweb – A Practical Guide*. Frankfurt: Peter Lang.
- Hunston, Susan. (2002). *Corpora in Applied Linguistics*. Cambridge: CUP.
- Manning, Christopher & Schütze, Hinrich. (1999). *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press.
- Stubbs, Michael. Collocations and Semantic Profiles: On the Cause of the Trouble with Quantitative Studies. *Functions of Language*, 2(1): 23–55.
- Zipf, George. (1949). *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*. Cambridge, MA: Addison-Wesley Press.

11

Understanding Markup and Annotation

We've already looked at more basic forms of enriching our data in earlier sections, as well as at PoS tagging as a fairly important and advanced one, but now, in this chapter, want to turn towards developing both an understanding of, as well as get some practice in, using relatively sophisticated ways of making linguistic data more useful than it would be in its raw form. As we've seen before, this makes a lot of sense because it not only allows us to distinguish features on different linguistic levels more easily, actually making them countable, but also to possibly exclude some parts of the data from our specific analyses, for instance by ensuring that we don't perform n-gram/collocation analyses across syntactic boundaries. Before we can take any detailed look at the technical aspects of such an advanced enrichment process and the best type(s) of format for this, though, we still need to deal with a few terminological distinctions, and try to understand the historical developments and motivation underlying the different formats.

As in the heading of this chapter, we frequently encounter two different terms when talking about the process of storing and enriching linguistic data, *markup* and *annotation*. While the term markup is sometimes used to indicate the physical act of marking specific parts of a text using specific symbols, and, in contrast, annotation may often refer to the interpretative information added, the two may also be used synonymously. This is also roughly the way in which we'll use them here. As such, they can refer to either a particular *format* for enriching a text, so as to

- categorise parts thereof, or
- make these parts more salient,

or to the actual *process* of applying this form of annotation.

We'll see some examples and different ways for applying markup to text later on, but for now, we're more interested in the reasons that exist for applying markup to our data in the first place, and roughly which different types of linguistic annotation currently exist. Despite a somewhat strong over-emphasis on morpho-syntactic annotation, Garside et al. 1997 provide a fairly good, though perhaps now somewhat dated, overview of different types or levels of annotation and their development over the years. Table 11.1, based very loosely on Leech (in Garside et al. 1997: 12), summarises the levels of linguistic information they identify, but also adds some. To make it easier to understand what the base-level representation formats and the enrichments are, I've added the two sub-categories for *representation* and annotation.

Table 11.1 Annotation types listed in Garside et al. 1997

<i>Level</i>	<i>Sub-category</i>	<i>Information Provided</i>
orthographic	representation	orthographic rendering of written or spoken language, with or without punctuation
phonetic/phonemic	representation	phonetic/phonemic rendering of spoken language
prosodic	annotation	tone/intonation units/breaks, stress, loudness, possibly also key
morpho-syntactic tagging	annotation	PoS tags, associated with morpho-syntactic units in various ways
syntactic annotation	annotation	skeleton parses or treebanks
semantic annotation	annotation	information about lexical fields, synonymy/antonymy
discourse annotation	annotation	discourse relations, cohesion
pragmatic annotation	annotation	speech acts, adjacency pairs, sometimes also co-reference (see above)
semantico-pragmatic annotation	annotation	information about illocutionary force indicating devices (IFIDs)
stylistic annotation	annotation	direct vs. reported speech, narrative vs. thought, etc.
dysfluency annotation	annotation	information about discourse phenomena, such as false starts, repetitions, filled pauses, etc.
error coding	annotation	information about grammatical & stylistic errors produced by learners in their writing or speech

As Table 11.1 indicates, perhaps the most important distinction here is whether the physical representation of the data is based on orthographic or phonetic/phonemic transcription. All other types add information to these base-level representations in various forms and to various degrees, also with potential overlap between types. As we're already familiar with some forms of morpho-syntactic

tagging, let's just take a brief look at an example of syntactic annotation in the form of a skeleton parse from the *Lancaster Parsed Corpus*.

```
A01 3
S&[N \0Mr_NPT Michael_NP Foot_NP N][V has_HVZ put_VBN V][R down_RP R]
[Na_AT resolution_NN [P on_IN [N the_ATI subject_NN N]P]N][S+ and_CC
[Na he_PP3A Na][V is_BEZ V][Ti[Vi to_TO be_BE backed_VBN Vi][P by_IN [N
\0Mr_NPT Will_NP Griffiths_NP ,-, [N \0MP_NPT [P for_IN [N Manchester_NP
Exchange_NP N]P]N]N]P]Ti]S+] ._. S&
```

As you can see in the above example, the syntactic annotation actually starts out from PoS information and then adds *labelled bracketing* for the syntactic categories. For reasons of space, we won't look at further examples of other types of annotation here now, but will later work more extensively on a sample that combines some of the different levels. Before doing so, though, we'll first try to develop an understanding of how the different formats that are mainly used today have come to exist and developed into their most up-to-date forms.

11.1 From SGML to XML – A Brief Timeline

In the 1960s, first attempts at standardising markup for information exchange began to be made. This was deemed necessary in order to be able to exchange various types of documents efficiently and without the need for extensive reformatting of the partly idiosyncratic/proprietary coding used by different authors, manufacturers, or publishing houses. However, it wasn't until 1986 that the SGML (Standard Generalized Markup Language) standard was in fact ratified by the International Standards Organisation (ISO). This first official type of markup language, however, as we'll see further below, was relatively complex and also had a number of serious drawbacks.

After Tim Berners-Lee had invented the *World Wide Web* (WWW) in 1989, it was necessary to develop a new, and simpler, markup language in order to take full advantage of the new hypertext medium. Thus, in 1992, HTML (*Hypertext Markup Language*) arrived on the scene and became popular very quickly. It's since then gone through various stages of development, the most recent versions being HTML 5 and XHTML. HTML in its more modern variants represents a simplified version of SGML, where many of the unwieldy features and shortcuts that made the processing of SGML rather difficult (see further below) were eliminated. In recent years, it's also further been enhanced by the addition of *Cascading Style Sheet* technologies (CSS1, 2 & 3) that make it possible to separate form and content better.

Because the formatting and linking capabilities offered by HTML were not always sufficient for all needs in document handling, and SGML proved too unwieldy and error-prone, a new hypertext format, XML (*eXtensible Markup*

Language) Version 1.0, was eventually created and released by the W3C (World Wide Web Consortium) in 1998. As can be seen from its name, XML allows users to create their own extended markup frameworks, and thereby makes it possible to increase the separation of form from content even further than was already possible by using HTML and CSS. Furthermore, XML can not only be used in conjunction with CSS, but also has its own stylesheet language XSL, which far exceeds the capabilities of CSS in that it also provides mechanisms for transforming documents from XML into other forms of XML, as well as various other types of formats, for display and processing. It also offers further improved processing and hyperlinking facilities in the form of XPath and XPointer technologies. Due to these advantages, we'll explore the use of XML further in Section 11.2.

11.2 XML for Linguistics

11.2.1 Why bother?

So why should we actually be tempted to 'mess around' with our nice and clean data and possibly go through a lot of trouble in adding markup? Well, for one thing, markup helps to structure information, for example, in separating documents into appropriate sections/divisions with headings, sub-headings, paragraphs, etc. We can also include certain types of meta-information that we talked about before, such as, for example, information about when and where the material has been collected, who has edited it and in which way, etc. The best examples we've seen for this were the meta-textual choices BNCweb allowed us to make for selecting specific parts of the BNC for different analysis purposes, which were all made possible by the fact that the BNC (in its most recent version) is marked up in XML, albeit with somewhat over-elaborate header information, as we'll soon discuss. On the other hand, adding annotations also allows the annotator to highlight interesting or important phenomena by using colour coding or other visualisation techniques, or simply helps to render a historical document in a relatively faithful and searchable manner online.

11.2.2 What does markup/annotation look like?

As pointed out in Section 11.1, all the formats we'll be discussing here are essentially plain text-based, and thus constitute 'human-readable' formats where the text itself contains different types of additional information, sometimes related to its structure, and sometimes to its linguistic content. Some of these formats are rather constrained in the types of information that can be added to a document, while others are more flexible, but sometimes even the less flexible ones can be 'coerced' into allowing us to add suitable types of annotation.

Although there are actually many different ways of marking up a document, one fairly standard method is the use of the kind of *tags* that we use for writing

HTML documents. These are actually more appropriately referred to as *elements*. Elements, in their most basic form, are generally represented in markup languages by pairs of opening and closing angle brackets, i.e. `< & >`, with the name of the element appearing in between the two. There are three different forms of such element tags:

- 1 opening tags (`<element_name>`), e.g. `<p>` for the start of a paragraph;
- 2 closing tags, where the name of the element is preceded by a slash (`</element_name>`), e.g. `</p>` for the end of a paragraph;
- 3 and ‘empty’ tags, where the closing bracket is preceded by (a space and) a slash (`<element_name />`), e.g. `<pause />` to represent a pause of undefined duration.

The first two are used to ‘bracket’, that is, enclose, elements that contain some form of textual content, such as the sentences inside a paragraph, the words inside a sentence, etc. The third type usually either specifies formatting instructions, such as line breaks, etc., contains links to external resources, such as a style sheet that specifies the layout and formatting options, or can be used to include other information that doesn’t require a containing element, such as, for example, a *comment* on a specific piece of data. We’ll see more examples of these later.

Opening and empty tags can also contain *attributes*, which are typically specified as attribute–value pairs, joined by an equals sign (=). Usually, these days, the value also needs to be quoted, using either single or double quotes. Attributes often specify sub-types, identifiers (commonly labelled *id*), counters (typically labelled *n*), or other features associated with a certain element. Therefore, a paragraph with the number 5 may be represented as `<p n="5">...</p>`, where the ellipsis (...) stands for the text contained inside it, or as `<para n="5">...</para>` or even `<paragraph n="5">...</paragraph>`, if you want to be even more explicit about it being a paragraph. The following is a brief sample paragraph that illustrates what a paragraph annotated in XML might look like in terms of elements and attributes.

```
<paragraph n="01"><unit n="01"><word n="01" pos="DD1">This
</word> <word n="02" pos="VBZ">is</word> <word n="03"
pos="AT1">a</word> <word n="04" pos="NN1">sample</word>
<word n="05" pos="NN1">paragraph</word> <word n="06" pos="VVG"
>illustrating</word> <word n="07" pos="DDQ">what</word>
<word n="08" pos="NP1">XML</word> <word n="09" pos="NN1">
formatting</word> <word n="10" pos="VM">may</word> <word
n="11" pos="VVI">look</word> <word n="12" pos="II">like</word>
<punc type="stop" /> </unit> </paragraph>
```

Exercise 82

Go through the above sample paragraph and make a list of how many elements there are and which type they belong to.

Next, also count the attributes and try to explain what they're used for.

11.2.3 The 'history' and development of (linguistic) markup

Although we've already talked about the history of markup languages before when I presented a brief timeline, we now want to return to this briefly and illustrate the differences, advantages and disadvantages of the different types.

We'll begin our little survey by presenting a short sample of SGML (Figure 11.1) and discussing some of its features.

```
<div2 complete=Y org=SEQ type=paragraph> <head type=MAIN>
<s n=0003> <w NP0>STATES <w CJC>AND <w AJ0>RELIGIOUS <w
NN2>INSTITUTIONS </head> <s n=0004> <w AT0>The <w AJ0>chief
<w NN1>purpose <w PRF>of <w AT0>the <w NN1>chapter <w VBZ>is
<w TO0>to <w VVI>outline <w AVQ>how <w AT0>the <w CRD>two <w
AJ0>opposing
```

Figure 11.1 A brief SGML sample

As we can see from Figure 11.1, SGML uses a fairly standard notation for opening tags, but isn't always consistent in indicating the ends of textual elements, so often the start of a new element simply has to be taken as a signal that the preceding element is now to be considered closed. Thus, although the heading inside the rather misleadingly labelled <head> element above has both a start and an end tag, the <s> (sentence) elements don't. This type of 'shortcut' makes processing SGML much more difficult than it needs be, and also much more error-prone.

In our example, we can also see that all attributes occurring in start tags are not quoted, but at least those can be handled relatively easily because they occur inside a tag and are also clearly marked by the equals sign, at least where there's an explicit attribute-value pair(ing) present, such as for "complete" in the "<div2" tag. The lack of any explicit attribute inside the <w> elements, indicating words, however, fails to clearly label the attribute name for the values specified inside each tag, which we therefore need to infer to be something like *pos* because the morpho-syntactic tags indicate this.

SGML also has two other major disadvantages compared to HTML or XML, the first being that it absolutely requires a DTD (*Document Type Definition*) specifying its structure, in order to allow any type of serious processing, and – last but not least – that it's not supported by any 'standard' (browser) software.

One big advantage, at least in comparison to HTML, is that a large set of tag definitions/DTDs for linguistic purposes, such as for the TEI (*Text Encoding Initiative*), were originally designed for SGML, although more and more of these have been or are being ‘ported’ to XML these days, too.

Although HTML is a direct descendant of SGML, it only provides a limited set of tags, which, on the one hand, makes it less flexible than XML, but, on the other, also much easier to learn. It’s widely recognised by standard (browser) software and the DTD(s) are already built into these browsers, although they can also be explicitly specified.

HTML itself defines a rather limited set of options for specifying structural properties of documents, such as elements for paragraphs (<p>), different levels of headings (<h1> – <h6>), larger textual divisions (<div>), smaller – inline – textual sequences (), different types of lists, certain types of formatting options (e.g. **bold**: , *italic*: <i>, etc.), as well as some processing instructions, such as for line breaks (
).

HTML itself is largely standardised and also technically extensible via CSS to some extent, so that it’s already quite useful for the presentation and visualisation of some linguistic content. The option to include dynamic content via client-side scripting, which can be achieved with relative ease, can also be seen as an advantage, although cross-platform development for different operating systems and browsers is somewhat hampered by inconsistent or incompatible implementations of the DOM (*Document Object Model*).

XML, however, is much more versatile than HTML because it was designed to be extensible by the user in providing the ability to completely define one’s own tags. It’s much easier to process and far less error-prone than SGML because some of the shortcuts we’ve seen before are no longer allowed.

All XML documents minimally have to be *well-formed*, that is, no overlapping tags (as in HTML, e.g. ...<i>......</i>) are allowed. Furthermore, end tags are required for all ‘non-empty’ elements, so they no longer need to be inferred. Empty tags also differ from their SGML/HTML equivalents in that they have to contain a slash before the closing bracket, i.e. <element_name />. Unlike in older forms of HTML, where case did not matter, XML is case sensitive, so that linguistic tags like <turn>, <Turn> & <TURN> for representing individual speaker *turns* in dialogues are all treated as being different from one another.

XML has been designed to be Unicode-aware right from the very beginning, so as to allow for markup using different character sets, also within one and the same document. If no encoding is specified, it always defaults to UTF-8, so that all basic ASCII characters occurring in English documents are always displayed correctly, even without explicitly having to convert existing ASCII encoded documents to UTF-8, as the basic code points are the same.

XML describes content (like SGML), rather than layout (like HTML), so that the exact rendering of a document needs to be specified via a style sheet because otherwise the browser/application displaying it wouldn’t know how to achieve

its task. If no style sheet is explicitly provided, most browsers will try to render the XML content using their own default style sheets, though. At the time of writing, the only major browser that didn't do this was Safari, so if this is your default browser, as for example on Mac OS X, you'll unfortunately need to install an additional one to be able to display the results of the exercises further down. I'd suggest that you use Firefox in this case, following the instructions at <https://support.mozilla.org/en-US/kb/how-download-and-install-firefox-mac> for installation. XML-aware browsers at least frequently attempt to represent the hierarchical tree structure, and often allow the user to expand and collapse nested structures. Other applications, such as editors, can at least display the plain text, provided they support the given encoding.

Apart from the well-formedness criterion described above, the document structure of an XML document can also be more rigorously constrained by specifying either a DTD or a *schema* that it needs to conform with. If an XML document conforms with one of these two types of specification, we talk of a *valid* document. We won't go into issues of designing DTDs or schemas here because they're fairly complex, but will at least have a look at some of the rendering options for XML documents using style sheets.

11.2.4 XML and style sheets

Style sheets in general allow the author to present/publish material in a visually more appropriate and/or appealing format, that is, specifying line spacing, indentation, positioning, etc., very similar to the formatting options you see in the text you're currently reading. Style sheet languages come in different flavours for different markup languages, namely DSSSL for SGML, CSS 1, 2 & 3 for HTML, and both CSS & XSL for XML. In our discussion below, as well as for our exercise(s), we'll concentrate on the essential aspects of CSS because it is simpler and easier to learn than XSL.

CSS1 already allowed the user to apply relatively 'low-level display formatting' such as layout, colours, positioning, drawing boxes around elements, etc. CSS2 & 3 expanded on these features by adding the ability to specify formatting options for publishing to different media – i.e. screen vs. printed paper, etc. – and advanced selection rules/mechanisms.

At the most basic level, CSS works by pairing the name of an element with different styling properties that the browser is supposed to associate with it. This is best done inside the style sheet definition, but may also happen inside the XML file itself. Such a pairing is similar to the attribute–value pairings we've seen above, only that in CSS, each definition starts with the name of the element defined and this is then followed by a listing of its individual properties inside a set of paired curly brackets (`{...}`). These brackets may in turn contain a number of attribute–value pairings, where the attributes are separated from their value(s) by a colon, with each property definition ending in a semi-colon. Therefore, for example, to define a basic paragraph layout, we could write a definition like the following,

which has already been styled to look like the resulting paragraph it may be applied to in an XML (or HTML) document:

```
p {
  display: block;
  background-color: #80ff00; color: red;
  font-weight: bold; font-style: italic;
}
```

Figure 11.2 CSS sample paragraph styling

In the simple example in Figure 11.2, you can already see a number of important styling concepts/properties provided by CSS style sheets, such as being able to apply different background (‘background-color’) and font/foreground (‘color’) colour definitions. In our case, the hexadecimal value provided for the background is a rather striking neon-green. The properties that contain the word *font* in them are responsible for making the font bold (‘font-weight’) and italic (‘font-style’), respectively.

While most of these properties used above should be relatively self-explanatory, the very first one, ‘display: block;’, may require some explanation: essentially, in defining display properties, we can make a distinction between *block(-level)* and *inline* display, where use of the former causes the element that is rendered to be separated from the surrounding content by clear spacing, breaking up the flow of the content, giving the visual appearance of separating it from the surrounding (con)text, as in the paragraphs in this book, while the latter means that the element remains embedded in the context, that is, does not interrupt the flow, such as in *this highlighted part* in this paragraph.

When it comes to specifying colour-values, there are two options for doing so, both of which are demonstrated in Figure 11.2, too. The best is to use a hash mark/pound symbol (#), followed by a hexadecimal code, because this defines an unambiguous colour value and provides more fine-grained options for specifying the exact colour. The second is to use a more basic-level colour term, which may then be interpreted by the browser in different ways, but usually providing something close to what we wanted. Please note that it may still be safer to use the American spelling variant(s) when referring to colours in CSS, although by now most browsers will probably also understand the British one ☺

CSS also provides us with numerous other options to refer to different elements or attributes in order to style documents in an appropriate and useful manner for colour-coding/visualising relevant linguistic information. We’ll soon encounter more of these formatting properties and options when we do our annotation exercises.

XSL (eXtensible Style Sheet Language) provides similar options to CSS for formatting XML display, but also offers much more complex selection mechanisms, as well as allowing reuse of ‘text objects’, for example, for producing tables of contents, etc., via so-called XSL Transformations (XSLT). Layout design for

other (printed) media is also supposed to be enhanced through XML Formatting Objects (XSL-FO).

Of course, XML doesn't need to be viewed inside a browser or transformed in any way, but can either be viewed, or even – at least to some extent – meaningfully searched, in one of our basic editors, or manipulated in other ways through programming languages, too. Furthermore, concordancers, such as AntConc, generally also allow us to search for XML tags because they simply represent plain searchable text, but may also sometimes provide options to hide them when we don't want to display them. Thus, it's not always necessary to use a style sheet, etc., to generate a specific kind of display format for a browser, but being able to use a browser for display simply provides one convenient option for exchanging and viewing our data using a specific formatting.

11.3 'Simple XML' for Linguistic Annotation

In this section, we want to explore how to use a form of XML that I refer to as 'Simple XML' in order to do annotation on multiple levels. The following exercise is designed to provide you with an introduction to this which is broken down into a series of steps that all successively allow you to deal with a particular XML-related or linguistic issue in creating a well-formed XML document.

Exercise 83a

Download our practice file from http://martinweisser.org/pract_cl/texts/practice_dialogue.txt.

Open the file in your plain-text editor.

Add the line `<?xml version="1.0" encoding="UTF-8" ?>` at the very beginning of the file.

Save the file as 'practice.xml', ensuring that the encoding is 'UTF-8', and without *Byte Order Mark* (BOM).

Keep the file open.

The line we just inserted identifies the document as being in XML for any application that's capable of displaying it, at the same time specifying its encoding as UTF-8. The question marks at the beginning and end identify the tag on this line as a processing instruction, in our case specifically the one referred to as the *XML declaration*. Please note that, so far, what we've done has not turned the file into valid XML yet as there are still some parts missing, so we're really just 'pretending' that the file is XML.

The next step is to create a 'container' element for our dialogue. This container element is also known as the *root element*, and every well-formed XML document

needs to have one. We'll label the root element for this document 'dialogue' in order to identify our text as a dialogue, and we'll also give it two attributes with the names 'corpus' and 'id' and corresponding values of 'test' and '01' respectively.

Exercise 83b

Wrap the whole text following the XML declaration into the container tag. Add the attributes specified above and their values.

By wrapping our text in the container element, we've effectively created the first (top) level of a hierarchy or categorisation, as well as a file that is now, at least technically, valid XML, despite the fact that it hasn't been sub-divided into meaningful parts yet.

We can now proceed to subcategorise the elements contained within the dialogue. If you look at the text again, you'll notice that the next level below the dialogue in our text is the speaker *turn*, so it would be quite logical to use this label for a tag surrounding each turn. As our turns have already been transcribed with a speaker *id* (*A* or *B*), followed by a number for the turn, we can identify these as suitable attributes characterising the turns.

Exercise 83c

Wrap a 'turn' element around each turn, keeping tags and text separate. If you feel adventurous, you can also try and save a fair amount of time here, experimenting with using regex replace operations and backreferences in your editor. Always remember, when something goes wrong, you can usually undo it, in the worst case simply closing the file without saving, then reopening it and starting all over again.

Save the file again, and open it in your browser to see what it looks like there. Keep the text editor open while you do this. That way you can switch between editing and viewing the results.

Don't despair if you end up with lots of errors in the beginning, as this is bound to happen quite frequently. Hopefully, though, you'll soon learn from any mistakes you make 😊

So far, we've already marked up all the individual speaker turns, that is, made it clearer who contributes what at which point, so we can now proceed to identifying the individual c-units within each turn.

Exercise 83d

Based on your knowledge of syntax, mark up all stretches of dialogue that you can identify, using the element names <decl>, <q-wh>, <q-yn>, and <imp> for the ‘traditional’ syntax types. For simplicity, and to improve readability, always separate each c-unit text from the tags by line breaks again, so that both container tags and text end up on separate lines.

In addition, mark up areas of text that represent terms of address (e.g. *Sir* or *Madam*), *discourse markers* (e.g. *well*, *ok*, *right*, *aha*, etc.) and yes-and no-like answers, using the tags <address>, <dm>, <yes>, <no>, as well as syntactically incomplete/ungrammatical units as <frag> for *fragment*.

Save the file and view it in your browser.

Now that we’ve finished marking up the syntactic level, we can move on to the pragmatic one. Here, we want to add speech-act attributes (*sp-act*) to the syntax elements to try and express what the intentions of the speakers are. The potential options you could use for speech acts, based on the syntactic categories, are:

- q-wh: reqInfo (request for information), reqDirect (request for directive/instruction)
 - q-yn: reqInfo, reqDirect
 - decl: state (stating/informing), agree, reqInfo, reqConfirm (requesting confirmation)
 - imp: direct (giving a directive/command), suggest (making a suggestion)
 - dm: init (initiating/initialising a new (sub-)topic), acknowledge (acknowledging)
 - yes: agree, accept
 - no: negate, refuse
 - address: refer
- frag: greet, intro (introducing oneself or a third party), thank, bye (saying goodbye), refer (providing deictic information)

Exercise 83e

Add what you think may be the appropriate label for each speech act to the individual syntactic units. When you do this, please remember that our interpretations may sometimes be a little subjective, so it’s best to compare your results with a colleague to see how much consensus you can reach.

As before, check the result in your browser.

Also think about whether your tags may warrant the use of attributes to express varying other features of your categories, such as polarity, semantic information, etc.

So far, we've now dealt with all the information that needs to be placed in container elements, but in our dialogue, you'll also find strange types of non-XML markup, like # and #{9s}. These represent pauses of undefined and defined length, respectively, and are not really part of our textual hierarchy. It therefore makes sense to convert them to empty elements. As our tag name, we'll use <pause> and as an attribute, we'll include information about the length of the respective pause, if any is indicated.

Exercise 83f

Replace all pauses in the dialogue by empty tags, save and review the document. In cases where there are two pauses marked in a row, only keep one of them.

Find the instance in the text that is marked as “unclear” in curly brackets and also replace this by an appropriate tag, including an attribute.

Delete any remaining comments inside curly brackets.

Next, try to find a good solution for representing *overlap*, which is indicated by / for the *starting* position and / for the *end* position, respectively, so there are essentially two *types*.

Finally, also replace any content in round brackets by <backchannel/> tags with ‘content’ attributes.

Save and test again.

Once you've marked up/annotated all the above and tested the result in your browser, you'll essentially have annotated your very first dialogue in XML successfully. You would now, for example, be able to load this dialogue into AntConc and search only for all discourse markers, or all requests for information, etc., in order to investigate them more closely, and possibly to come up with new ideas for marking up such content further in order to (re-)classify it more precisely. If you had a number of such dialogues, you could also search for only turns produced by speaker A in order to investigate the language use of call centre agents. For a real-life example of how you can work with this type of pragmatically annotated data, you can download the SPAADIA corpus (see Section 2.4.1.3), from which the dialogue we just worked with was taken, and which is fully annotated, including a few further levels of annotation.

Sometimes, however, we may simply want to form a quick impression of certain features and this is where plain XML, as it is displayed by the browser, isn't nearly as useful as being able to identify important features of dialogues visually via appropriate colour coding, which is something we'll practise in Section 11.4.

11.4 Colour Coding and Visualisation

In this section, we want to explore some of the basic and slightly advanced features of CSS, which will enable us to style our XML file in a useful manner, based on its elements and sometimes even specific properties, such as speech acts, etc. We'll begin by defining some display properties for the whole dialogue, and then gradually move on to define options for the relevant elements and attributes.

Exercise 84a

Create a new file in your editor.

Type in `dialogue`, followed by a set of paired curly brackets. The first properties we want to set here are the background colour (`background-color`), the font size (`font-size`), and the left margin (`margin-left`), so you can add each one of these within the curly brackets, each time followed by a colon, a space, and a semi-colon, as illustrated in our discussion of CSS earlier on.

Now, set the properties as shown in Table 11.2:

Table 11.2 CSS properties for XML visualisation exercise

<i>Property</i>	<i>Value</i>	<i>Explanation</i>
<code>background-color</code>	<code>#ffffdd</code>	This hexadecimal value sets the background to ivory.
<code>font-size</code>	<code>1.25em</code>	This sets the font size to 1¼ times the original font size, which is usually around 10 points in the browser. As <code>em</code> is a relative value, this allows you to increase or decrease the display size in the browser in relative steps by pressing <code>Ctrl+</code> or <code>Ctrl-</code> , respectively.
<code>margin-left</code>	<code>2.5%</code>	This sets a left margin/indent for the page, mimicking the margin on a printed page, so that the text isn't squashed against the left side of the display window.

Save the file as 'dialogue.css' in the same folder as the dialogue file.

So far, we've only created the basic style sheet, but we also need to 'let the browser know' that it's supposed to use it with our dialogue file, so we need to create a link between the two.

Exercise 84b

Open your XML dialogue file again and insert the line `<?xml-stylesheet type="text/css" href="./dialogue.css"?>` between the XML declaration and the dialogue tag, then save it.

Load the XML file in your browser and view the result.

You may now be surprised because, apart from the few general formatting options we've just set for the dialogue itself, all the levels in our hierarchy that we've so painstakingly set before will have disappeared and the text simply runs on without any indication of where one turn or syntactic unit starts and ends. This is the case because the browser now assumes that, since you've 'told' it that you want to style the XML yourself, it should no longer apply its own built-in style sheet, but instead leave all the styling up to yours. Therefore, in the next few steps, we'll have to learn about some further styling options that'll allow us to do just that. We'll start by (re-)formatting the turns.

Exercise 84c

Go back to the style sheet and create a new definition for turns, setting the properties for `display`, `color`, `margin-left`, `line-height`, and `text-indent` to `block`, `black`, `5.5%`, `1.5em`, and `-3.5em`, respectively. What these mean should hopefully be relatively self-explanatory, based on what we learnt earlier.

Now, save the style sheet, go back to the browser, and refresh the page to see the result.

Each turn should now appear in a block of its own, but we still haven't got any indication of the speaker id, so we don't really know who's talking when. Although it could potentially be an interesting exercise for students in classroom activities to identify this, it's not really what we want to see here, so we need to find a way to access the speaker attribute information to use it in our style sheet. Luckily for us, in our style sheet definitions, we can not only refer to elements, but also to attributes, so we can easily style turns by speaker A (i.e. the agent) differently from those by speaker B (the caller). In order to do so, we'll not just make use of the speaker attribute, but will also exploit another feature of CSS (2/3), which is that we can generate content *before* or *after* each element automatically.

To be able to access information about the speaker attribute, we need to use an attribute-value pair with an equals sign, similar to the way we define attributes in XML, but without the quotation marks around the value. This needs to be enclosed in a pair of square brackets, so writing `turn[speaker=A]` will allow us

to refer to all turns produced by speaker A, which we could already exploit in order to style these turns differently in terms of background colour, etc., if we wanted to. However, instead of changing display properties like this, we'll manipulate the actual textual content by prefixing each turn with the contents of its 'n' attribute, as well as with information regarding the roles of the speakers. To achieve this, we need to access the so-called *pseudo-class* called 'before' of the relevant turn, which is appended to the earlier definition we created using a colon as a connector, thus yielding `turn[speaker=A]:before`. Knowing this, we now only need to learn how to manipulate/refer to the text we want to pre-pend to the turn. This can be achieved via its 'content' property, which, in its most basic form, is simply a string of text enclosed in double quotes, so we can write `turn[speaker=A]:before {content: "Agent:";}` to make the word *Agent* followed by a colon appear before each turn produced by speaker A.

Exercise 84d

Try adding the definition for speaker A, the *Agent*, to your style sheet.
Do the same for speaker B, the *Caller*.
Test the result in the browser.

The only thing left to do now is to incorporate the information from the 'n' attribute, that is, the turn number, into the pre-pended content. To refer to the value of the n attribute within the content property, we can use the general style sheet syntax for accessing such attribute information inside property definitions, which is to use `attr`, followed by the name of the relevant attribute in round brackets, i.e. `attr(n)` in our case. This should be placed outside the quotation marks in the appropriate place, and thus we'll write `turn[speaker=A]:before {content: attr(n) " - Agent:";}` to complete our definition, where we've also modified the content inside the quotation marks slightly to separate it from the turn information.

Exercise 84e

Amend your prior definitions for both speakers and test this inside the browser again.

Having formatted the turns, we can now move on to dealing with the syntactic elements occurring inside them. Unlike the turns, which we'd styled as block-level elements, we'll format these as being `inline` and with a left margin of `.5%`, for which you should already have enough knowledge to write the individual definitions. As this is a kind of formatting that we'll equally want to apply to all such units, it would be cumbersome to have to type this out repeatedly, so we can take advantage of a feature of CSS that allows us to list the different elements a

definition applies to by separating them from each other by a comma, just like in an ordinary written list.

Exercise 84f

Add the style definitions for all the syntactic element classes.
Test the result.

Please note that, so far, we've only set two general properties for the syntax elements, but still need to set their individual display options, so as to be able to recognise and distinguish them visually from one another. Creating the definitions that are individual to each syntactic category essentially entails finding suitable background and foreground colours to make the different types either maximally distinctive or, in contrast, to possibly emphasise common features, such as the similarity between the different types of questions, which can be expressed by using the same background colour for both. For some of the other features, we can use specific types of colour coding semantics to signal 'open' and 'closed' options. For instance, to signal the open-ended nature of wh-questions, we can use green, while we can use red to indicate the limited options of yes/no-questions. For yes and no answers, we can employ a similar type of traffic signal analogy and encode them like one-way street signs. Table 11.3 summarises the colour semantics we want to use.

Table 11.3 Colour semantics and CSS styles

<i>element(s)</i>	<i>background colour</i>	<i>semantics</i>	<i>font colour</i>	<i>semantics</i>
<q-wh> & <q-yn>	#f5f984;	shows similarity between questions		
<q-wh>			green	analogy to a green traffic light, i.e. 'go' to signal open choices
<q-yn>			red	analogy to a red traffic light, i.e. 'no go' to signal closed set of choices
<yes>	blue	analogy to traffic sign, entering a one-way street the right way, i.e. 'go ahead'		
<no>	red	analogy to traffic sign, entering a one-way street the wrong way, i.e. 'no entry'		

Exercise 84g

Add the above definitions to the style sheet and test the results.

Another thing we want to do here, apart from applying colour-coding semantics to our units, is to actually add the punctuation that was missing from the original transcripts. We can do this by using the counterpart of the *before* pseudo-class, *after*, in order to generate it automatically, based on the syntactic class. This works in exactly the same way as when we pre-pended content to all turns.

Exercise 84h

Define *yes*, *no*, *decl*, *frag*, and *address*, as being followed by a dot, the questions a question mark, discourse markers a dash, and imperatives by an exclamation mark in the style sheet.

There are still a few more syntactic elements that we haven't defined any colour-coding for. Simply create definitions for these using the following values:

- discourse markers: white font, #ff8000 background
- declaratives: blue font, white background
- imperatives and fragments: background #d9ffff
- imperatives: red font
- fragments: font colour #4b4b4b.

Exercise 84i

Create the remaining colour definitions in the appropriate places inside your style sheet and check the results again.

One very important feature of the style sheet rules is that the definitions inside the style sheet are applied by the browser in exactly the order that they were written in. This effectively means that any later re- or additional definition for already defined specific elements or attributes will override earlier existing definitions. We can exploit this feature in order to display requests for information contained inside fragments in the same font colour as wh-questions.

Exercise 84j

Create a definition for the font colour of both *wh*-questions and any items that have a speech act *reqInfo* in the appropriate place, setting them to #009b00.

In order to refer to the speech act attribute–value combination, you need to use a similar syntax to the one we used for our definitions of the different turns, only without an element name in front of the square brackets.

Finally, we also want to override the punctuation mark that occurs after all syntactic units that may contain requests for information, which may also include declarative questions, apart from the fragments we just handled.

Exercise 84k

Based on your knowledge from the earlier tasks in this exercise, you should already know how to achieve this, so just write an appropriate definition and test the results again.

When examining the results so far, you may have noticed that all empty elements were also removed from the display. Unfortunately, as CSS definitions are generally designed to help us render the content occurring inside non-empty elements, we cannot simply write a definition that will display the empty elements because they technically speaking don't contain any content. This is why we have to use a little trick to have them displayed inside our browser, which is that we simply 'recreate' them using the *after* pseudo-class. All we have to do to achieve this is first to declare them as inline elements and then write the appropriate content definitions, where we use the same text as in the actual elements, but get their relevant attribute values using the `attr()` syntax we used previously for retrieving the turn numbers. As we also want them to be unobtrusive, we'll use a grey font colour (#808080) for all empty elements.

Exercise 84l

First, define all empty elements, *pause*, *unclear*, *overlap*, and *backchannel*, as inline elements with the font colour specified above.

Next, write the individual definitions for them, using the *after* pseudo-class with an appropriate CSS content attribute that uses the correct attribute of the XML element to extract and display its value.

Test the result in your browser again.

The two lengthier exercises above were essentially designed to provide you with two different perspectives on the use of annotations. The first one basically gave you a means of designing your own ways of classifying your data in a sensible manner, which then enables you to use the corpus-based analysis methods we've discussed throughout the course in order to extract and count relevant information. The second one was designed to allow you to develop alternative views of your data, something that may help you, or any students or colleagues you may create teaching materials or presentations for, to quickly visualise different features that might be present in individual texts. This may, for instance, make it possible for students to notice the number of discourse markers or syntactic fragments that occur in spoken interaction, or identify other important elements of such types of interaction. For analysis purposes, both techniques, if used sensibly, will often lead to a cyclical refinement of the categories identified and their respective annotation forms, enabling the researcher to 'fine-tune' the analysis results and thereby also the conclusions that can ultimately be drawn from the corpus data.

11.5 More Complex Forms of Annotation

The form of annotation I introduced you to above already contains many different bits of information, so that it may appear fairly complex to you. However, I still refer to this as 'Simple XML' because it remains easily readable due to the relatively low number of container elements and the separation of text from non-empty tags. This separation, incidentally, is possible because most XML processors simply ignore any whitespace they 'perceive' as redundant. Because these processors generally parse the XML and often produce tree structures based on the hierarchy of the elements, they'd thus have rendered our sample in the same way, even without the additional line breaks I suggested you insert for readability.

Some proponents of 'pure' XML technology would even frown upon what we've done here and argue that XML was meant to be processed by the computer, anyway, and could then be rendered in whichever way it would be best viewed. However, this argument ignores the fact that, before any annotation is finished, it repeatedly, and often for very long periods of time, needs to be read and edited by humans, so that readability does indeed represent an issue in annotation. Even the best fully automated annotation still needs to be checked for errors by human editors, although, as we've seen in the many examples of mis-tagging in the BNC and COCA, this is often not done for reasons of time and cost involved, especially the larger the amount of data processed becomes. Furthermore, any process of rendering, combined with editing and saving, involves making modifications to the data in forms invisible to the user, and may be error-prone because

it repeatedly needs to save the changed data and then update the view again. In addition to this, using such software also ties the average user unnecessarily into using often complex annotation tools that themselves represent a relatively steep learning curve, apart from further potential issues regarding platform availability and setup.

Adding further levels of annotation to corpus data almost always leads to added complexity in the data, although, for instance, striking the right balance between using an appropriate number of container elements, empty elements, and suitable attributes can already help us to go a long way, as hopefully Exercise 84I has shown you. Further decisions of course need to be made regarding how much meta-information each file in the corpus absolutely needs to contain, or whether there isn't a choice to relegate some of this information to external *header files* (cf. Leech et al. 2000: 13) that can be accessed as and when this information may be necessary, and can easily be distributed along with the corpus. These have the distinct advantage that they keep the original data 'clean', and supplementary information can even be added later without having to change the original data.

The BNC data we've been working with represent a different way of handling annotation, in that they, for instance, contain a relatively complex header that contains a high degree of meta-information, which can be seen in Figure 11.3.

```
<bncDoc xml:id="KST"><teiHeader><fileDesc><title><title> 12 conversations recorded by
'Margaret2' (PS6RG) between 20 and 27 February 1992 with 7 interlocutors, totalling 5346
s-units, 31800 words, and 2 hours 47 minutes 0 seconds of recordings.
</title><respStmt><resp> Data capture and transcription </resp><name> Longman ELT </name>
</respStmt></title><edition><edition>BNC XML Edition, December 2006
</edition></edition><extent> 32166 tokens; 34252 w-units; 5346 s-units
</extent><publication><distributed>Distributed under licence by Oxford University
Computing Services on behalf of the BNC Consortium.</distributed><availability> This
material is protected by international copyright laws and may not be copied or redistributed
in any way. Consult the BNC Web Site at http://www.natcorp.ox.ac.uk for full licensing and
distribution conditions.</availability><idno type="bnc">KST</idno><idno type="old"> XMa6RG
</idno></publication><sourceDesc><recording><recording xml:id="KSTRE000" n="042201"
dur="133" date="1992-02-20" time="16:15" type="Walkman"/><recording xml:id="KSTRE001" n=
"042202" dur="546" date="1992-02-20" time="16:15" type="Walkman"/><recording xml:id=
"KSTRE002" n="042203" dur="1328" date="1992-02-20" time="16:15" type="Walkman"/><recording
xml:id="KSTRE003" n="042204" dur="795" date="1992-02-20" time="16:15" type="Walkman"/>
<recording xml:id="KSTRE004" n="042205" dur="1520" date="1992-02-21" time="16:15" type=
"Walkman"/><recording xml:id="KSTRE005" n="042206" dur="606" date="1992-02-21" time="13:15"
type="Walkman"/><recording xml:id="KSTRE006" n="042207" dur="465" date="1992-02-21" time=
"13:15" type="Walkman"/><recording xml:id="KSTRE007" n="042301" dur="1506" date="1992-02-21"
time="18:00+" type="Walkman"/><recording xml:id="KSTRE008" n="042401" dur="2332" date=
"1992-02-27" time="16:00" type="Walkman"/><recording xml:id="KSTRE009" n="042601" dur="70"
date="1992-02-25" time="09:15" type="Walkman"/><recording xml:id="KSTRE00A" n="042602" dur=
"558" date="1992-02-25" time="09:15" type="Walkman"/><recording xml:id="KSTRE00B" n="042701"
dur="161" date="1992-02-27" time="13:10" type="Walkman"/>
</recording></sourceDesc></fileDesc><encodingDesc><tagsDecl><namespace name=""><tagUsage
gi="align" occurs="3714"/><tagUsage gi="c" occurs="6017"/><tagUsage gi="div" occurs="12"/>
<tagUsage gi="event" occurs="2"/><tagUsage gi="gap" occurs="20"/><tagUsage gi="mw" occurs=
```

Figure 11.3 TEI header for BNC file KST

Figure 11.3 shows the TEI header for a single file, KST, from the BNC. As should be obvious from this, apart from linguistically relevant information about the speakers involved, there's also a lot of non-linguistic information stored here

regarding the recording, the distribution, etc. The *Text Encoding Initiative* (TEI; <http://www.tei-c.org/index.xml>), upon whose recommendations the above header and general annotation of the BNC is based, is an organisation of researchers involved in annotating language data, and has made numerous suggestions about how and what to annotate in samples of both spoken and written language. However, often their ideas for standards take into consideration far more detail than may be necessary, so, while it's well worth looking at these recommendations to get some initial ideas if you want to prepare a corpus for general distribution, often working according to their guidelines may be overkill for smaller, personal projects.

Another popular format, especially in language engineering circles, is the so-called *standoff* format (Thompson & McKelvie 1997). This format makes it possible to link different levels of annotation in separate XML documents to one base file – usually based on word tokens – in a similar way to the kinds of databases behind BNCweb and COCA. However, apart from the fact that word tokens are not always the appropriate base unit in texts, as you'll hopefully remember from our discussion in Section 9.1.1, this type of format a) makes it impossible to read any file without the use of dedicated software, and b) is in fact considerably more complex than a database approach. As such, it probably only lends itself to research and corpus building for relatively large research projects, where there is also support for dedicated software. An example of the use of standoff annotation is the OASIS corpus (see Section 2.4.1.3).

From the above discussion, you'll hopefully have seen that, especially for smaller corpus projects, the motto should always be 'the simpler, the better'. And one always ought to remember that, even though one might never know how a corpus may potentially be used by others, not all types of annotation need to be included in all corpora, simply because this may be possible or relatively easily achievable. Therefore, for instance, a very good, and still valid, example of the use of different versions of one and the same set of corpus data is the SEC (see Section 2.3.1.2), which actually exists in five different forms.

Solutions to/Comments on the Exercises

Exercise 82

The elements you should be able to identify in the first part of this exercise are <paragraph>, <unit>, <word>, and <punc />. The first three of these represent container elements for the different syntactic/textual levels described, while the last one is an empty one, representing punctuation as a non-word.

In terms of attributes, you should be able to find 27, where 'n' represents numerical identifiers for all the 14 textual elements, 'pos' the 12 PoS categories for the words, and 'type' which type of punctuation is present at the end of the syntactic unit.

Exercise 83

(a) Inserting the XML declaration and saving the file should generally present no problem, but ensuring the actual encoding for the file in UTF-8 (without BOM) may present a little bit of an issue, depending on which editor you're using. If your editor is encoding-sensitive, like Notepad++, it'll normally have retained the original encoding of the text file, which was already the correct one. If you need to change the encoding, though, where exactly this can be done, and also how you can see this, may vary. And, unfortunately, there's no way for me to tell you how exactly this will work in your editor if you're using a different one, so I can only give you some tips as to where such option can possibly be set. In some editors, the encoding can be set in the 'Save' dialogue, together with the file name. For instance, in Windows Notepad, you can select the option for UTF-8 under 'Encoding' in this dialogue, although, unfortunately, there's no way to specify the additional option to exclude the BOM we don't want, and which is in fact unnecessary and, if present, may also cause display issues in some browsers. In other editors, such as Notepad++, there are additional options available via a dedicated 'Encoding' menu item, where you can specify what to encode a file in or even to convert between a limited set of encodings. In Notepad++, you can also specify the default encoding for any files you create under 'Settings→Preferences→New Document', where I'd recommend you set the option for 'UTF-8 without BOM', as well as use the additional setting 'Apply to opened ANSI files'.

(b) Hopefully, you remembered that *only* the start tag can have any attributes in it, so that our start tag should now read `<dialogue corpus="test" id="01">`. Our end tag, which you need to insert after the last line of the dialogue, will then be `</dialogue>`.

(c) Wrapping a `<turn>` element around each turn should be quite straightforward. For instance, the tag for the first turn ought to look similar to this: `<turn speaker="A" n="01">`. Of course, you can also opt for single quotes, if you want to, as long as you remember to be consistent with your opening and closing quotes. If your browser indicates an error, go back to the XML file and first verify whether you've set all start and end tags correctly, as this is generally the most common error. One of the most common error messages you'll probably encounter will be something like "XML Parsing Error: mismatched tag. Expected: `</turn>`". The browser will usually also try to provide a line and column (i.e. character position) number for where the error was found, but this will often not be very helpful because XML parsers are unfortunately not very good at locating such errors, so that, frequently, the error location pointed out is only towards the end of the file where the browser (finally) 'notices' a mismatch between start and end tags. While trying to fix potential errors, however, some editors may provide a certain degree of help through syntax highlighting. For instance, when you click on the start tag in Notepad++, it'll normally highlight

both the start tag and its corresponding end tag, so if you go through the file from the top, you can perhaps identify the error this way, even if it may be a rather tedious process, especially if the file is long. Anyway, once you think you've fixed an error, switch back to the browser and refresh the display.

Using regex replacements here, rather than tediously adding all the tags manually, is possible because the turns are basically all represented in our file as single lines that start with a speaker id, followed by a dot, the turn number, a colon, a space, and then the rest of the turn. Thus you can construct a regex expression that captures the speaker id to be reused as `\1`, the number of the turn as `\2`, and everything following the colon and space until the end of the line in `\3`. And, to be able to automatically keep the turn tags separated from the text, we can add new lines in the appropriate places. To do all this, you need to define one character class enclosed in round brackets to match the speakers, either A or B, at the beginning of the line, ideally escape the dot that follows, then capture one or more digits for the turn number, again enclosed in round brackets, match a colon, one or more spaces (just to be on the safe side), then as many characters as possible, again in round brackets, until you reach the end of the line. To avoid any potential unwanted spaces at the beginning or end of the line, you can also add some optional whitespace. Therefore, the search term, with regex search option switched on, would be: `^\s*([AB])\.\s*(\d+)\s*:\s*(.+)\s*$`. To achieve the replacement, including line breaks, the code construct would then be `<turn n="\2" speaker="\1">\n\3\n</turn>`.

(d) This part of the exercise may initially be more difficult because you'll have to think 'out of the box' in terms of new and unusual syntactic categories, but you'll soon get used to this. The beginning of the file should now look like this:

```
<?xml version="1.0" encoding="UTF-8" ?>
<dialogue id="01" corpus="test">
<turn speaker="A" n="01">
<frag>
good afternoon
</frag>
<frag>
Virgin train line Sandra speaking
</frag>
<q-wh>
for which journey do you wish to purchase a ticket
</q-wh>
</turn>
<turn speaker="B" n="02">
<frag>
er Euston to Manchester please
</frag>
</turn>
```


(e) Again, this part may be a little difficult if you've never worked with/on speech acts before, but most of the ones listed should be relatively intuitive. One thing the list of speech acts I gave you doesn't include is a complete set for speech acts signalling responses, although some of them, such as, for example, 'accept' or 'refuse', already contain such information. Other speech acts, though, may be responding in a similar way, but this response may be paired with one of the other options. Thus, a 'reqInfo' speech act by one speaker will often trigger an 'answer' by the other, but this answer is generally a statement ('state') in our above taxonomy. Thus, to be more explicit, we could in fact also allow more than one speech act to occur in our annotation, which would result in 'answer-state' for this particular example.

(f) Replacing the pauses so that our empty tag will, for example, either look like `<pause />` or `<pause length="9s" />`, should be quite easy. The empty element for the 'unclear' 5 syllables should, logically, become `<unclear length="5 syllables" />`. Regarding overlap tags, you'd actually be quite right in assuming that, theoretically, these should be container elements because they mark up specific spans of text. However, as XML forbids the use of overlapping tags, and overlap passages span across different speaker turns, using `<overlap>...</overlap>` elements would break the document's well-formedness, so it's best to use the empty tags `<overlap type="start" />` and `<overlap type="end" />`, possibly in combination with a number ('n') attribute. As backchannels constitute text where another speaker simply provides feedback to the current speaker who holds the turn, but doesn't really interrupt to take over, it also makes sense to incorporate this via an empty element. As before, when we created the turn elements, because there are easily recognisable patterns, you can save a lot of time if you're able to construct appropriate regex replacement patterns, so you should definitely try this.

For a complete solution of the annotated dialogue, see Appendix B. As an alternative to some of the steps we modelled as regexes above, and for slightly more convenient manual annotation of the remaining XML structure, you could also use an annotation tool, such as my Simple Corpus Tool, which actually includes an editor that'll allow you to add these tags and attributes through the click of a button. For a fully automatic large-scale annotation of the syntax, speech acts, etc., you can also try my Dialogue Annotation and Research Tool (DART), which not only allows you to annotate hundreds of dialogues in this way within minutes, but also to post-edit/correct the annotations, as well as to carry out similar analysis operations to those we learnt how to perform in AntConc, including concordancing, n-gram analysis, etc.

Exercise 84

(a) The display options for the whole dialogue you set should hopefully look as follows, where I've simply written everything on one single line for compactness:

`dialogue {background-color: #ffffdd; font-size: 1.25em; margin-left: 2.5%;}`. As with XML, in CSS slight mistakes in the syntax may cause errors, so if the display you get doesn't seem to reflect our properties, you need to check your style sheet for errors.

(b) Adding the style sheet reference should need no further comment.

(c) The definition for the turns should look like this: `turn {display: block; color: black; margin-left: 5.5%; line-height: 1.5em; text-indent: -3.5em;}`.

(d) This part of the exercise is probably a bit more difficult, especially because there are some fairly complex new CSS terms and concepts involved. However, with a little bit of effort, you'll hopefully be able to set the appropriate display entries for both dialogue participants to:

- `turn[speaker=A]:before {content: "Agent:";}`
- `turn[speaker=B]:before {content: "Caller:";}`.

(e) Again, this part may sound a little complex, but with a little experimenting you'll hopefully manage to produce the following definitions:

- `turn[speaker=A]:before {content: attr(n) " - Agent:";}`
- `turn[speaker=B]:before {content: attr(n) " - Caller:";}`.

(f) Adding the general definition for all syntactic element classes is quite straightforward. The line required in the style sheet is: `address, decl, dm, frag, imp, no, yes, q-yn, q-wh {display: inline; margin-left: .5%;}`.

(g) This should again be straightforward and not require further comment.

(h) For adding the punctuation marks, you should end up with the following definitions:

- `yes:after, no:after, frag:after, decl:after, address:after {content: ".";}`
- `q-wh:after, q-yn:after {content: "?";}`
- `dm:after {content: "-";}`
- `imp:after {content: "!";}`

(i) As the basic colour options for the remaining syntax elements are quite straightforward, I won't provide a separate solution for these here.

(j & k) Changing all wh-questions and any other elements that contain requests for information as a speech act to a separate font colour can be achieved via: `q-wh, [sp-act=reqInfo] {color: #009b00;}`, and adding a question mark after any request for information by using `[sp-act=reqInfo]:after {content: "?";}`.

(1) We can set the unobtrusive font colour for all empty elements in one go, using the following definition: `pause`, `unclear`, `overlap`, `backchannel` `{display: inline; color: #808080;}`.

The individual definitions for displaying content should look as follows:

- `pause:after {content: "<pause length='\"attr(length)\"' />";}`
- `unclear:after {content: "<unclear length='\"attr(length)\"' />";}`
- `overlap:after {content: "<overlap type='\"attr(type)\"' />";}`
- `backchannel:after {content: "<backchannel content='\"attr(content)\"' />";}`

For a complete version of the style sheet, see Appendix C.

Sources and Further Reading

- Bradley, Neil. (1998). *The XML Companion*. Harlow: Addison-Wesley.
- Garside, Roger, Leech, Geoffrey, & McEnery, Tony. (Eds.) (1997). *Corpus Annotation: Linguistic Information from Computer Text Corpora*. London: Longman.
- Leech, Geoffrey. (1997). Introducing Corpus Annotation. In Garside, Roger, Leech, Geoffrey, & McEnery, Tony. (Eds.) (1997). *Corpus Annotation: Linguistic Information from Computer Text Corpora*. London: Longman.
- Leech, Geoffrey, Weisser, Martin, Wilson, Andrew, & Grice, Martine. (2000). Survey and Guidelines for the Representation and Annotation of Dialogue. In Gibbon, Mertins, & Moore. (Eds.). (2000). *Handbook of Multimodal and Spoken Language Systems*. Dordrecht: Kluwer Academic Publishers.
- World Wide Web Consortium. Extensible Markup Language. <http://www.w3.org/XML/>.

Conclusion and Further Perspectives

Throughout this book, I've tried to give you many opportunities to develop an awareness, coupled with some basic experience, of how linguistic data can be used to solve linguistic puzzles or questions. At the same time, whenever appropriate, I've also tried to point out other potential applications for such data, for example, in the development of teaching materials/textbooks, grammars, or direct application in the classroom, but of course such a list will always be incomplete as there are too many applications of corpus linguistics to be listed exhaustively. To get a better overview of other applications, you can – now that you should have more than at least a basic understanding – turn to the additional literature I've listed for the different sections, especially the many other, generally more theoretically oriented, textbooks, or the two handbooks I listed in the Introduction, O'Keeffe & McCarthy (2010) and Lüdeling & Kytö (2008), for more in-depth information or further inspiration, and also be able to evaluate their contents carefully. For now, all that remains for me to do is to summarise what we've tried to achieve here, and to give you some further pointers on where to get information other than from publications.

In the first two main sections of the book (Chapters 2–4), we started out by investigating the different forms language data may come in, especially in the shape of existing corpora, and then moved on to developing an understanding of how you can complement such data by collecting your own, including which problems and pitfalls you may encounter in this endeavour, focussing on the nature and sometimes 'messiness' of electronic data. Here, I already tried to give you at least some sense of how data that hasn't been prepared well may cause specific issues

and errors in later analysis stages. Apart from teaching you how to prepare your data, this was essentially also a plea for showing diligence in preparing corpora, especially if your intention may be to distribute your data to a wider audience.

In the next major section (Chapters 5–10), we then investigated various techniques for analysing language data using established methods of corpus linguistics. Here, as much as possible, I've tried to show you complementary techniques for working both with your own data and larger, general, corpora through web-based interfaces, thereby also allowing you to some extent to test findings derived from your own data against reference corpora, or to compare linguistic phenomena across different corpora. Throughout this section, we encountered various issues with tools and methods, again partly illustrating the effects of data where flaws in the basic compilation of the corpus may cause potential errors in the result, but partly also pointing out potential shortcomings in the particular tools at our disposal. What you'll hopefully have come to realise from this in particular is that no tool, even if it may have been designed with the best of intentions by the author(s), and to incorporate as many facilities as possible, and even if it may be highly customisable, such as AntConc, is ever perfect for all our potential purposes. Thus, we constantly need to be aware of such shortcomings, as well as try and find new and creative, but nevertheless solid, ways of overcoming these issues.

One of the major issues we've repeatedly encountered, especially concerning the mega corpora we've worked with, is that the creation of large-scale resources may frequently lead to the compilers taking shortcuts when it comes to ensuring the quality of the data in terms of tokenisation and annotation. Furthermore, the design of the annotation and interfaces available may sometimes exhibit flaws from a linguistic perspective, as we've, for instance, seen for the CQP architecture behind BNCweb, which treats punctuation tokens in exactly the same way as genuine words, thereby potentially skewing all the statistics produced by the tool. Another issue we've encountered in this context is that often tools designed for analysis almost force us into accepting the 'orthographic word' as the correct unit of analysis, which is e.g. exemplified in the fact that the BYU interfaces don't allow us to compare 'words' with 'phrases' directly. These are just some of the issues we need to constantly be aware of when we use such tools, so the idea that 'bigger is better', even if it is indeed often important to work with very large amounts of data for such research as collocation analysis in order to be able to find rarer combinations, may not always be fully justified if the quantity of data isn't equally matched by quality. Having criticised some of the larger corpora and their interfaces rather heavily, I now need to relativise this again, as, of course, creating such large and complex resources still remains a highly laudable effort despite these apparent flaws, and having these mega corpora and associated interfaces already takes us a very long way in advancing our knowledge of language and its uses, apart from providing us with huge amounts of extremely valuable real-life materials for teaching and learning, and other applications of or in linguistics.

In the final brief section (Chapter 11), I've tried to provide you with a short, but nevertheless highly practical, glimpse at what current technology in the form of XML has to offer to linguists who want to enrich their data and/or visualise important facts inherent in it. Here, I've deliberately advocated a form of annotation I call 'Simple XML', which still makes it possible to read and edit the corpus data, or annotate it further, but without the need for complex interfaces many less computer-literate corpus linguists may have a hard time to even install, let alone use without accepting a steep learning curve.

This book being (only) an introductory textbook, the topics we've covered can obviously not be exhaustive, in particular because of the more practical nature of our approach. I've therefore had to restrict my discussion of relevant topics in corpus linguistics to what I consider the most essential ones for beginners. There are, however, also many other applications of corpus linguistics I've been unable to cover, but which may be of interest to you now that you've mastered the basics, and which I'd like to mention here briefly.

Diachronic/historical corpus linguistics is an important area that aims at answering many of the as yet unanswered questions in language history, including differences in style, vocabulary, grammar, and even pragmatics. In translation studies and teaching, parallel corpora, that is, corpora that cover similar textual matter or even represent aligned translations of texts, continue to have a very strong influence. In L1 Acquisition, there's a substantial body of data that has been collected following the standard of the Child Language Data Exchange System (CHILDES; <http://childes.psy.cmu.edu/>), and on which active research is being carried out. And, of course, corpora and the original techniques developed in corpus linguistics, such as concordancing and the creation of frequency lists, have also had a very strong impact on the areas of Computational Linguistics, Natural Language Processing (NLP), and Language Engineering. However, research in those disciplines has taken a very different direction that is often far removed from corpus linguistics, although sometimes self-proclaimed corpus linguists tend to forget this. The reason for this is quite simple: in these disciplines, the objective of research is generally not so much to understand how language works or can be taught, but to achieve very practical aims, such as processing large quantities of data (e.g. taken off the web) efficiently and fully automatically in order to provide interfaces to search engines, Q & A or dialogue systems, where humans can 'interact' in 'natural language' with the computer and get quick results. This is why creating efficient algorithms or 'quick-and-dirty' solutions is often more important in these areas because it's simply economically more viable, while understanding the finer nuances of language that linguists and other language professionals may be interested in is only of minor, if any, importance. This also appears to be at least part of the reason why textbooks in NLP, such as Manning & Schütze (1999) or Jurafsky & Martin (2009), tend to focus more on quantitative analyses and very quickly turn to discussions and implementations of probabilistic techniques for language analysis.

Of course, this use of statistics, as we've seen to a very small extent when we looked at collocations, etc., is also partly shared with corpus linguistics, and methods and textbooks in this domain, such as Oakes (1998) or Gries (2009), to name but two, enjoy increasing popularity. However, a word of warning is in order here. Many of the assumptions and tests in statistical analysis are still based on the premise of data being normally distributed, which is a feature often observed in nature, but simply not true for most aspects of language, as, for example, Zipf (1949) has clearly shown. This is probably the case because language, unlike natural phenomena that cannot be controlled, involves making conscious choices on many levels. Thus, while undeniably statistics do have a role to play in language analysis, many commonly used statistical measures are still not applicable to language research, and thus greater efforts need to be made to devise better, more language-appropriate measures.

As this brief overview has shown, there are still many more advanced topics in corpus linguistics for you to explore. However, for the moment, I'd first like to suggest that you consolidate what you've learnt from this book further, and devise your own ways of dealing with any issues you may encounter in your own research. If, however, somewhere along the line you should realise that what the tools available have to offer you is not enough, you can always investigate learning how to write your own analysis programs, as this will liberate you from many of the constraints described above, and will also give you a degree of flexibility in your research that no single tool can offer. The idea of doing this may at first appear daunting, but I can assure you that, if you've gone through this book carefully and now feel relatively secure in conducting the types of analysis described here, taking this next logical step will already have become much easier...

Last, but not least, I should also point out that there aren't only book or journal publications relevant to corpus linguistics, but also online discussion groups on various social or professional media, such as Facebook or LinkedIn, and especially the corpora mailing list at <http://www.hit.uib.no/corpora/> that you can subscribe to for up-to-date discussions, even if they, sadly, may often these days contain more content related to NLP-oriented conferences or research, rather than 'genuine' corpus linguistics. In addition, links to many corpora and resources appear on David Lee's 'devoted to corpora' website, which I already referred to in Chapter 2. And finally, once you've actually reached a stage where you're ready to share your own corpus research with others, you may well want to consider attending either the ICAME, CL, or TaLC (Teaching and Language Corpora) conferences that take place at regular (yearly) intervals.

Glossary

ACE: the Australian Corpus of English, also known as the Macquarie Corpus (see Section 2.3.1.1)

alignment: establishing a link between the source text and the translation, usually at the sentence, phrase or word level

ANC: the American National Corpus (see Section 2.3.2.2)

annotation: the process of enriching corpus data with (interpretative) linguistic information (see Chapter 11)

ARCHER: a diachronic corpus (see Section 2.3.3)

ASCII: American Standard Code for Information Interchange (see Section 3.3.1)

BNC: the British National Corpus (see Section 2.3.2.2)

BNCweb: the web interface of the BNC (see Section 8.1.1)

BoE: the Bank of English (see Section 2.5)

Brown: the Brown University Standard Corpus of Present-day American English (see Section 2.3.1.1)

balance: the notion in corpus building, especially for reference corpora, that a corpus should contain as many representative pieces of texts, from all representative genres, as well as possibly all relevant media (i.e. generally written vs. spoken vs written-to-be-spoken); often also linked to representativeness (see Section 3.1.3)

CES: the Corpus Encoding Standard

character encoding: a computer-based system of representing characters as numbers (see Section 3.3)

- colligation:** the association/collocation (see below) of a PoS category or word with a particular word class (see Section 10.5.2)
- collocation:** the characteristic co-occurrence of lexical patterns involving two or more words within a certain (limited) distance of one another, usually up to 4–5 words to the left or right of the word under investigation (the node); these can be investigated/identified ‘manually/visually’ or by means of statistical test for co-occurrence (see Sections 10.5.1 & 10.8.1)
- concordance:** a listing of a word/expression in a corpus, generally within a specific context (cf. KWIC) (see Section 5.1)
- concordancer:** a software package that generates concordances, as well as possibly displays of other linguistic features, for corpora (see Section 5.1)
- corpus (pl. corpora):** a collection of texts in machine-readable form (see Section 2.1)
- corpus-based (research):** research that primarily uses corpora to verify the researcher’s intuitions
- corpus-driven (research):** research based on few, if any, prior intuitions, where the results of corpus analysis directly lead to new insights and theories
- dispersion:** a term in descriptive statistics which refers to a quantifiable variation of measurements of differing members of a population within the scale on which they are measured (see Section 10.7)
- ditto tag:** in corpus annotation assigning the same part-of-speech code to each word in an idiomatic expression (see Section 9.1.1)
- Document Type Definition (DTD):** a separate document in markup languages such as HTML, SGML and XML that is linked to a content-bearing document and defines which elements and attributes are allowed to occur within the document, thereby constraining available options and making it possible to test the validity (as opposed to well-formedness) of the document (see Section 11.2.3)
- EAGLES:** Expert Advisory Group on Language Engineering Standards; an initiative of European researchers that made recommendations regarding standards and guidelines for best practice in Language Engineering
- EAP:** English for Academic Purposes
- error-tagging:** the process of adding error-related information to a corpus
- FLOB:** the Freiburg-LOB Corpus of British English, an update of the LOB corpus in the early 1990s (see Section 2.3.1)
- frequency:** the number of occurrences of a linguistic feature in a corpus (see Section 9.2)
- Frown:** the Freiburg-Brown Corpus of American English, an update of the Brown Corpus in the early 1990s (see Section 2.3.1)
- HTML:** Hypertext Markup Language (see Sections 4.2 & 11.2.3)
- header:** the (top) part of a corpus that provides bibliographical information and meta-data related to the corpus (see Sections 3.2.1 & 11.5)
- ICE:** the International Corpus of English; an initiative to represent as many varieties of English as possible in the form of comparable corpora (see Section 2.3.2.1)

- ICLE:** the International Corpus of Learner English; an initiative to represent as many varieties of learner English as possible in the form of comparable corpora (see Section 2.4.1.2)
- keyword:** words in a corpus whose occurrence is unusually frequent (positive keywords) or infrequent (negative keywords) in comparison to a reference corpus (see Section 9.3.3)
- KWIC:** key-word-in-context (concordance); a listing of search terms found in a corpus, where the search term(s) generally appears centred on the line, with a fixed number of words or characters to the left and right (see Section 5.1)
- learner corpus:** a collection of essays/academic writing by, or interviews with, learners in the form of a corpus, possibly also containing error coding/annotation (see Section 2.4.1.2)
- lemma:** cf. headword/entry in a lexicon; generally, a lemma is assumed to subsume all forms of a word paradigm, with further distinctions according to PoS (see Sections 8.1.8 & 8.2.1)
- lemmatisation:** grouping together all of the different inflected forms of the same word, possibly also according to PoS
- LLC:** the London-Lund Corpus; one of the earliest spoken corpora, containing exhaustive prosodic markup (see Section 2.3.1.2)
- LOB:** the Lancaster-Oslo-Bergen Corpus of British English (see Section 2.3.1)
- LOCNESS:** the Louvain Corpus of Native English Essays; a corpus of native-speaker essays specifically created for comparison purposes with learner data (see Section 2.4.1.2)
- log-likelihood:** statistical measure used for collocation or keyword analysis; also referred to as G^2
- markup:** a term often used synonymously with annotation to refer to adding enriching information to a corpus; sometimes specifically used to refer to the physical act of marking specific parts of a text using specific symbols (see Chapter 11)
- meta-data:** a term used to describe data about data, typically the contextual information of corpus samples (see Section 3.2.1)
- MI:** mutual information, a statistical measure of co-occurrence that measures the strength of collocations (see Section 10.8.1)
- MICASE:** the Michigan Corpus of Academic Spoken English (see Section 2.4.1.1)
- monitor corpus:** a corpus that is constantly updated and enlarged with new corpus materials (see Section 2.5)
- MWU:** multi-word unit; a unit of sense that consists of multiple words (see Section 9.1.1)
- node:** the central word or search term in a collocation or concordance (see Sections 10.5.3, 10.6 & 10.8.1)
- normalisation/norming:** the first term may refer to two things, either a) the process of standardising transcriptions/data by regularising spelling where multiple variants may exist (see Section 4.3), or b) the process of norming frequency

counts by a relative factor, such as per million words (see Section 9.5); the second term only applies to b)

OCR: optical character recognition; a process where scanned images of text are converted to machine-readable and searchable text

OLAC: the Open Language Archives Community; “an international partnership of institutions and individuals who are creating a worldwide virtual library of language resources” (<http://www.language-archives.org/>)

parallel corpus: a corpus which is composed of source texts and their translations in one or more different languages; sometimes referred to as translation corpus

parsing: a process that analyses the sentences in a corpus into their constituents (see Chapter 11), also called treebanking or bracketing

polysemy: the ability of word forms to represent multiple meanings or word classes, e.g. *bank* as ‘financial institution’ (noun), ‘side of a river’ (noun), ‘act of keeping an account with a financial institution’ (verb), ‘turn sharply’ (verb)

PoS (part-of-speech): word class or morpho-syntactic category (see Chapter 7)

post-editing: human correction of automatically processed data (see Section 11.5)

reference corpus: a balanced representative corpus for general usage (see Section 2.4); in keyword analysis, a corpus that is used to provide a reference wordlist (see Section 9.3.3)

representativeness: the notion in corpus building that a corpus built for a specific (or general) purpose should include a suitable number of texts that describe the characteristics of the domain(s) covered suitably (see Section 3.1.3)

SEC: the Lancaster/IBM Spoken English Corpus (see Section 2.3.1.2)

SED: the Survey of English Dialects corpus

semantic preference: a term used to indicate the particular words a node word collocates with in its immediate environment

semantic prosody: a term used to indicate the particular words or topics, including positive or negative connotations, a node word collocates with in its wider, non-immediate, environment; see also semantic preference

SEU: Survey of English Usage; an early, non-computerised corpus, later half computerised as the LLC (see Section 2.3.1.2)

SGML: Standard Generalized Markup Language; the first standardised markup language used for linguistic purposes, now superseded by XML (see Section 11.2.3)

sorting: arranging concordance or a frequency list data in a certain order, alphabetical, descending alphabetical, ascending frequency, descending frequency, or reverse sorted (see Section 5.2.1)

SPAAC/SPAADIA: the Speech Act Annotated Dialogue Corpus (see Section 2.4.1.3)

span: in collocation analysis, the number of words to be taken into consideration on either side of the node word (see Section 10.8.1)

- specialised corpus:** a domain-specific corpus designed to represent a particular sublanguage or genre (see Section 2.4)
- subcorpus:** a component of a corpus, usually defined using certain criteria such as text types and domains (see Section 9.3.2)
- tagging:** an alternative term for annotation, especially word-level annotation such as PoS tagging and semantic tagging (see Chapter 7)
- tagset:** a scheme of codes for corpus annotation, especially PoS tagging (see Section 7.1)
- TEI:** the Text Encoding Initiative; an organisation of researchers involved in annotating language data that publishes guidelines and recommendations for such annotations (see Section 11.5)
- text archive:** a repository of texts of different types, possibly also including complete or partial corpora (see Section 4.1.3)
- token:** an actual occurrence of any given word form, as opposed to its type (see Section 9.1.2)
- tokenisation:** also called segmentation, a process that divides running text into individual tokens considered to constitute words (see Section 10.2)
- transcription:** converting spoken data into a written form, either rendering it orthographically or phonetically/phonemically
- treebank:** an alternative term for a parsed corpus (see Chapter 11)
- t-score:** a statistical measure to establish the certainty of collocations (see Section 10.8.1)
- type:** a word form (label), as opposed to its individual occurrences (tokens) in a text (see Section 9.1.2)
- type-token ratio:** the ratio of types to tokens
- Unicode:** a character encoding system designed to support the processing and display texts from languages across the world, including phonetic characters, etc. (see Section 3.3.2)
- URL:** Uniform Resource Locator, i.e. a web address (see Section 4.2.4)
- wildcard:** a special character such as an asterisk (*) or a question mark (?) that can be used to represent one or more characters in pattern matching (see Section 8.1.5)
- word (frequency) list:** a list of words occurring in a corpus, generally with frequency information (see Section 9.2)
- WSC:** the Wellington Corpus of Spoken New Zealand English (see Section 2.3.1)
- WWC:** the Wellington Corpus of Written New Zealand English (see Section 2.3.1)
- XCES:** XML Corpus Encoding Standard
- XML:** the Extensible Markup Language (see Section 11.2)

Appendix A

The CLAWS C5 Tagset

<i>Tag</i>	<i>Description</i>
AJ0	adjective (unmarked) (e.g. GOOD, OLD)
AJC	comparative adjective (e.g. BETTER, OLDER)
AJS	superlative adjective (e.g. BEST, OLDEST)
AT0	article (e.g. THE, A, AN)
AV0	adverb (unmarked) (e.g. OFTEN, WELL, LONGER, FURTHEST)
AVP	adverb particle (e.g. UP, OFF, OUT)
AVQ	wh-adverb (e.g. WHEN, HOW, WHY)
CJC	coordinating conjunction (e.g. AND, OR)
CJS	subordinating conjunction (e.g. ALTHOUGH, WHEN)
CJT	the conjunction THAT
CRD	cardinal numeral (e.g. 3, FIFTY-FIVE, 6609) (excl. ONE)
DPS	possessive determiner form (e.g. YOUR, THEIR)
DT0	general determiner (e.g. THESE, SOME)
DTQ	wh-determiner (e.g. WHOSE, WHICH)
EX0	existential THERE
ITJ	interjection or other isolate (e.g. OH, YES, MHM)
NN0	noun (neutral for number) (e.g. AIRCRAFT, DATA)
NN1	singular noun (e.g. PENCIL, GOOSE)
NN2	plural noun (e.g. PENCILS, GEESE)
NP0	proper noun (e.g. LONDON, MICHAEL, MARS)
NULL	the null tag (for items not to be tagged)

<i>Tag</i>	<i>Description</i>
ORD	ordinal (e.g. SIXTH, 77TH, LAST)
PNI	indefinite pronoun (e.g. NONE, EVERYTHING)
PNP	personal pronoun (e.g. YOU, THEM, OURS)
PNQ	wh-pronoun (e.g. WHO, WHOEVER)
PNX	reflexive pronoun (e.g. ITSELF, OURSELVES)
POS	the possessive (or genitive morpheme)'S or '
PRF	the preposition OF
PRP	preposition (except for OF) (e.g. FOR, ABOVE, TO)
PUL	punctuation - left bracket (i.e. (or [)
PUN	punctuation - general mark (i.e. . ! , ; - ? ...)
PUQ	punctuation - quotation mark (i.e. ' ' ")
PUR	punctuation - right bracket (i.e.) or])
TOO	infinitive marker TO
UNC	"unclassified" items which are not words of the English lexicon
VBB	the "base forms" of the verb "BE" (except the infinitive), i.e. AM, ARE
VBD	past form of the verb "BE", i.e. WAS, WERE
VBG	-ing form of the verb "BE", i.e. BEING
VBI	infinitive of the verb "BE"
VBN	past participle of the verb "BE", i.e. BEEN
VBZ	-s form of the verb "BE", i.e. IS, 'S
VDB	base form of the verb "DO" (except the infinitive), i.e. DO
VDD	past form of the verb "DO", i.e. DID
VDG	-ing form of the verb "DO", i.e. DOING
VDI	infinitive of the verb "DO"
VDN	past participle of the verb "DO", i.e. DONE
VDZ	-s form of the verb "DO", i.e. DOES
VHB	base form of the verb "HAVE" (except the infinitive), i.e. HAVE
VHD	past tense form of the verb "HAVE", i.e. HAD, 'D
VHG	-ing form of the verb "HAVE", i.e. HAVING
VHI	infinitive of the verb "HAVE"
VHN	past participle of the verb "HAVE", i.e. HAD
VHZ	-s form of the verb "HAVE", i.e. HAS, 'S
VM0	modal auxiliary verb (e.g. CAN, COULD, WILL, 'LL)
VVB	base form of lexical verb (except the infinitive)(e.g. TAKE, LIVE)
VVD	past tense form of lexical verb (e.g. TOOK, LIVED)
VVG	-ing form of lexical verb (e.g. TAKING, LIVING)
VVI	infinitive of lexical verb
VVN	past participle form of lex. verb (e.g. TAKEN, LIVED)
VVZ	-s form of lexical verb (e.g. TAKES, LIVES)
XX0	the negative NOT or N'T
ZZ0	alphabetical symbol (e.g. A, B, c, d)

Appendix B

The Annotated Dialogue File

```
<?xml version="1.0" encoding="UTF-8" ?>
<?xml-stylesheet type="text/css" href="./dialogue.css"?>
<dialogue id="01" corpus="test">
<turn n="1" speaker="A">
<frag n="01" sp-act="greet">
good afternoon
</frag>
<frag n="02" sp-act="identifySelf">
Virgin train line Sandra speaking
</frag>
<q-wh n="03" sp-act="reqDirect">
for which journey do you wish to purchase a ticket
</q-wh>
</turn>
<turn n="2" speaker="B">
<frag n="04" sp-act="direct">
er Euston to Manchester please
</frag>
</turn>
<turn n="3" speaker="A">
<dm n="05" sp-act="init">
<pause /> now
</dm>
```

<q-yn n="06" sp-act="reqInfo">
do you hold a current debit or credit card
</q-yn>
</turn>
<turn n="4" speaker="B">
<dm n="07" sp-act="acknowledge">
aha
</dm>
</turn>
<turn n="5" speaker="A">
<q-yn n="08" sp-act="reqInfo">
do you have a railcard at all
</q-yn>
</turn>
<turn n="6" speaker="B">
<no n="09" sp-act="answer">
no
</no>
</turn>
<turn n="7" speaker="A">
<q-wh n="10" sp-act="reqInfo">
and how many people's travelling
</q-wh>
</turn>
<turn n="8" speaker="B">
<frag n="11" sp-act="answer">
just one please
</frag>
</turn>
<turn n="9" speaker="A">
<q-wh n="12" sp-act="reqInfo">
and what date is it you're travelling
</q-wh>
</turn>
<turn n="10" speaker="B">
<decl n="13" sp-act="answer">
the Saturday which is the third i think
</decl>
</turn>
<turn n="11" speaker="A">
<frag n="14" sp-act="confirm">
third of Oc...
</frag>
</turn>

<turn n="12" speaker="B">
<frag n="15" sp-act="complete">
<overlap type="start" /> October
</frag>
</turn>
<turn n="13" speaker="A">
<frag n="16" sp-act="echo">
</frag>
<q-wh n="17" sp-act="reqInfo">
departing at what time from London Euston
</q-wh>
</turn>
<turn n="14" speaker="B">
<decl n="18" sp-act="state">
i'm not sure what time the trains are
</decl>
<q-yn n="19" sp-act="reqInfo">
do you know
</q-yn>
</turn>
<turn n="15" speaker="A">
<dm n="20" sp-act="init">
well
</dm>
<decl n="21" sp-act="state">
the trains run at 10 to the hour every hour
</decl>
</turn>
<turn n="16" speaker="B">
<frag n="22" sp-act="echo">
every hour <pause />
</frag>
<decl n="23" sp-act="express-opinion">
and that'll be the 14 50 i think then
</decl>
</turn>
<turn n="17" speaker="A">
<frag n="24" sp-act="state">
arriving at 17 30
</frag>
</turn>
<turn n="18" speaker="B">
<decl n="25" sp-act="acknowledge">
that's right

```

</decl>
<yes n="26" sp-act="acknowledge">
yeah
</yes>
</turn>
<turn n="19" speaker="A">
<q-wh n="27" sp-act="reqInfo">
when are you returning
</q-wh>
</turn>
<turn n="20" speaker="B">
<frag n="28" sp-act="answer">
er Monday
</frag>
</turn>
<turn n="21" speaker="A">
<frag n="29" sp-act="reqInfo">
departing at what time
</frag>
</turn>
<turn n="22" speaker="B">
<frag n="30" sp-act="unclassifiable">
em {unclear_2_syllables} that time
</frag>
</turn>
<turn n="23" speaker="A">
<frag n="31" sp-act="state">
on the half hour
</frag>
<decl n="32" sp-act="confirm">
and that's Monday the fifth
</decl>
<yes n="33" sp-act="acknowledge">
yeah
</yes>
</turn>
<turn n="24" speaker="B">
<decl n="34" sp-act="acknowledge">
that's right yeah
</decl>
<imp n="35" sp-act="suggest">
em let's say <pause /> half 2 in the afternoon
</imp>
</turn>

```

<turn n="25" speaker="A">
<dm n="36" sp-act="init">
<pause length="8s" /> now
</dm>
<decl n="37" sp-act="state">
there's a train at 14 30 from Manchester Picadilly <pause />
</decl>
<decl n="38" sp-act="state">
you arrive in London Euston for 17 hundred
</decl>
</turn>
<turn n="26" speaker="B">
<dm n="39" sp-act="acknowledge">
ok
</dm>
<decl n="40" sp-act="appreciate">
that's great
</decl>
</turn>
<turn n="27" speaker="A">
<dm n="41" sp-act="init">
now
</dm>
<decl n="42" sp-act="informIntent-hold">
i'm just going to check with you what's your cheap-
est fare available to you
</decl>
</turn>
<turn n="28" speaker="B">
<dm n="43" sp-act="accept">
sure <pause length="9s" />
</dm>
<decl n="44" sp-act="state">
i was quoted 19 pounds
</decl>
</turn>
<turn n="29" speaker="A">
<decl n="45" sp-act="state">
the 19 pounds is not available at the dates and times specified
</decl>
</turn>
<turn n="30" speaker="B">
<dm n="46" sp-act="acknowledge">
right

```
</dm>
</turn>
<turn n="31" speaker="A">
<decl n="47" sp-act="state">
you'd need to travel out an hour later on Satur-
day and come back on the 19 59 train in the evening on Monday
</decl>
</turn>
<turn n="32" speaker="B">
<dm n="48" sp-act="acknowledge">
right em
</dm>
</turn>
<turn n="33" speaker="A">
<decl n="49" sp-act="state">
if it has to be the dates and times that you speci-
fied to me <backchannel content="ok" />
</decl>
<decl n="50" sp-act="state">
that's a super advance return at 30 pounds
</decl>
</turn>
<turn n="34" speaker="B">
<frag n="51" sp-act="confirm">
30 em <pause />
</frag>
<imp n="52" sp-act="hold">
let me think <pause />
</imp>
<q-yn n="53" sp-act="req-modal">
do you mind if i cancel
</q-yn>
</turn>
<turn n="35" speaker="A">
<decl n="54" sp-act="confirm">
you don't want to book at all
</decl>
</turn>
<turn n="36" speaker="B">
<no n="55" sp-act="acknowledge">
no thank you
</no>
</turn>
<turn n="37" speaker="A">
```

```
<dm n="56" sp-act="init">  
ok then sir  
</dm>  
<frag n="57" sp-act="bye">  
<overlap type="start" /> bye  
</frag>  
</turn>  
<turn n="38" speaker="B">  
<frag n="58" sp-act="bye" >  
bye <overlap type="end" />  
</turn>  
</dialogue>
```

Appendix C

The CSS Style Sheet

```
/*
style sheet for displaying call centre dialogues
author: Martin Weisser
last edit: 11:34 12-Sep-2014
*/
/* this controls the display for the dialogue container*/
dialogue {background-color: #ffffdd; font-size: 1.25em;
margin-left: 2.5%;}
/* every turn is displayed as a block level element, i.e. with
spacing before and after */
turn {display: block; color: black; margin-left: 5.5%;
line-height: 1.5em; text-indent: -3.5em;}
/* every turn with speaker attribute A has Agent: pre-pended
automatically, every turn by speaker B, Caller: */
turn[speaker=A]:before {content: attr(n) " - Agent:";}
turn[speaker=B]:before {content: attr(n) " - Caller:";}
/* indent every syntactic tag and make it an inline element */
address, decl, dm, frag, imp, no, yes, q-yn, q-wh {display:
inline; margin-left: .5%;}
/* every discourse marker (DM) gets an orange background and
white foreground*/
dm { color: white; background-color: #ff8000;}
/* a dash is automatically appended to every DM */
```

```

dm:after {content: " -";}
yes {color: white; background-color: blue;}
no {color: white; background-color: red;}
imp, frag {background-color: #d9ffff;}
frag {color: #4b4b4b;}
imp {color: red;}
imp:after {content: "!";}
decl {color: blue; background-color: white;}
yes:after, no:after, frag:after, decl:after, address:after
{content: ".";}
q-wh, q-yn {background-color: #f5f984;}
q-wh, [sp-act=reqInfo] {color: #009b00;}
q-yn {color: red;}
q-wh:after, q-yn:after {content: "?";}
/*
change the punctuation mark after any reqInfo tag to a
question mark, overriding any previous mark
*/
[sp-act=reqInfo]:after {content: "?";}
/*
in order to be able to display empty elements, we have to
cheat because they don't normally get displayed.
what we can do here is to use the pseudo-element syntax for
text appearing after elements in order to insert the content
*/
pause, unclear, overlap, backchannel {display: inline;
color: #808080;}
pause:after {content: "<pause length='"attr(length)'" />";}
unclear:after {content: "<unclear length='"attr(length)'" />";}
overlap:after {content: "<overlap type='"attr(type)'" />";}
backchannel:after {content: "<backchannel content='"attr
(content)'" />";}

```

References

- Abney, Steven. (1996). Statistical Methods and Linguistics. In Klavans, J. & Resnik, P. (Eds.). (1996). *The Balancing Act*. Cambridge, MA: MIT Press.
- Anderson, Wendy & Corbett, John. (2009). *Exploring English with Online Corpora: An Introduction*. Basingstoke: Palgrave Macmillan.
- Anthony, Laurence. (2014). AntConc (Version 3.4.3) [Computer Software]. Tokyo, Japan: Waseda University. Downloadable from <http://www.laurenceanthony.net/>
- Atkins, Sue, Clear, Jeremy, & Ostler, Nicholas. (1992). Corpus Design Criteria. *Literary and Linguistic Computing*, 7(1).
- Barnbrook, Geoffrey. (1996). *Language and Computers: A Practical Introduction to the Computer Analysis of Language*. Edinburgh: EUP.
- Biber, Douglas. (1993). Representativeness in Corpus Design. *Literary and Linguistic Computing*, 8(4).
- Biber, Douglas, Conrad, Susan, & Reppen, Randi. (1998). *Corpus Linguistics: Investigating Language Structure and Use*. Cambridge: CUP.
- Biber, Douglas, Johansson, Stig, Leech, Geoffrey, Conrad, Susan, & Finegan, Edward. (1999). *Longman Grammar of Spoken and Written English*. London: Longman.
- Burnard, Lou. (2009). The BNC in Numbers. <http://www.natcorp.ox.ac.uk/corpus/index.xml?ID=numbers>. [last accessed: 12-Aug-2015]
- Bondi, Marina & Scott, Mike. (Eds.). (2010). *Keyness in Texts*. Amsterdam: John Benjamins.
- Cheng, Winnie, Greaves, Chris, & Warren, Martin. (2008). *A Corpus-driven Study of Discourse Intonation: the Hong Kong Corpus of Spoken English (Prosodic)*. Amsterdam/Philadelphia: John Benjamins.
- Cloren, Jan. (1999). Tagsets. In van Halteren, H. (Ed.). (1999). *Syntactic Wordclass Tagging*. Dordrecht: Kluwer Academic Publishers.

- Cotterill, Janet. (2008). How to Use Corpus Linguistics in Forensic Linguistics. In O'Keeffe, A. & McCarthy, M. (Eds.) (2010). *The Routledge Handbook of Corpus Linguistics*. London: Routledge.
- Coxhead, Averil. (2002). The Academic Word List: A Corpus-based Word List for Academic Purposes. In Kettemann & Marko. (Eds.). (2002). *Teaching and Learning by Doing Corpus Analysis*. Proceedings of the Fourth International Conference on Teaching and Language Corpora, Graz 19–24 July, 2000. Amsterdam: Rodopi.
- Davies, Mark. (2009). The 385+ Million Word *Corpus of Contemporary American English* (1990–2008+): Design Architecture, and Linguistic Insights. *International Journal of Corpus Linguistics*, 14(2).
- Davies, Mark. (2010). The Corpus of Contemporary American English as the First Reliable Monitor Corpus of English. *Literary and Linguistic Computing*, 25(4).
- DeRose, Stephen. (1988). Grammatical Category Disambiguation by Statistical Optimization. *Computational Linguistics*, 14(1), pp. 31–39.
- Dybkjær, Laila, Hemsén, Holmér, & Minker, Wolfgang (Eds.). (2007). *Evaluation of Text and Speech Systems*. Berlin/New York: Springer.
- Edwards, Jane & Lampert, Martin. (Eds.). (1993). *Talking Data: Transcription and Coding in Discourse Research*. Hillsdale, New Jersey: Lawrence Erlbaum Associates.
- Fillmore, Charles. (1992). 'Corpus Linguistics' vs. 'Computer-Aided Armchair Linguistics'. In Svartvik, Jan. (Ed.). *Directions in Corpus Linguistics*. Berlin: DeGruyter, pp. 35–60.
- Friedl, Jeffrey. (2006). *Mastering Regular Expressions* (3rd edition). Sebastopol, CA: O'Reilly.
- Garside, Roger, Leech, Geoffrey, & McEnery, Anthony. (Eds.) (1997). *Corpus Annotation: Linguistic Information from Computer Text Corpora*. London: Longman.
- Garside, Roger & Smith, Nicholas. (1997). A Hybrid Grammatical Tagger: CLAWS4. In Garside, R., Leech, G., & McEnery, A. (Eds.) (1997). *Corpus Annotation: Linguistic Information from Computer Text Corpora*. London: Longman.
- Gries, Stefan. (2009). *Quantitative Corpus Linguistics with R: A Practical Introduction*. New York/London: Routledge.
- Hoffmann, Sebastian, Evert, Stefan, Smith, Nicholas, Lee, David, & Berglund Prytz, Ylva. (2008). *Corpus Linguistics with BNCweb – A Practical Guide*. Frankfurt: Peter Lang.
- Hunston, Susan. (2002). *Corpora in Applied Linguistics*. Cambridge: CUP.
- Hunston, Susan. (2008). How Can a Corpus Be Used to Explore Patterns? In O'Keeffe, A. & McCarthy, M. (Eds.) (2010). *The Routledge Handbook of Corpus Linguistics*. London: Routledge.
- Illouz, Gabriel. (2000). Sublanguage Dependent Evaluation: Toward Predicting NLP Performances. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC)*, Athens, Greece. pp. 1251–1254.
- Jenks, Christopher. (2011). *Transcribing Talk and Interaction*. Amsterdam: John Benjamins.
- Jurafsky, Daniel & Martin, James. (2009). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition* (2nd edition). Upper Saddle River, NJ: Prentice Hall.
- Kennedy, Graeme. (1998). *An Introduction to Corpus Linguistics*. London: Longman.
- Kettemann, Bernhard & Marko, Georg. (Eds.). (2002). *Teaching and Learning by Doing Corpus Analysis*. Proceedings of the Fourth International Conference on Teaching and Language Corpora, Graz 19–24 July, 2000. Amsterdam: Rodopi.

- Ladefoged, Peter. (2003). *Phonetic Data Analysis: An Introduction to Fieldwork and Instrumental Techniques*. Oxford: Blackwell.
- Lee, David. (2002). Genres, Registers, Text Types, Domains and Styles: Clarifying the Concepts and Navigating a Path through the BNC Jungle. In Kettemann & Marko. (Eds.). (2002). *Teaching and Learning by Doing Corpus Analysis*. Proceedings of the Fourth International Conference on Teaching and Language Corpora, Graz 19–24 July, 2000. Amsterdam: Rodopi.
- Leech, Geoffrey, Rayson, Paul, & Wilson, Andrew. (2001). *Word Frequencies in Written and Spoken English*. London: Longman.
- Leech, Geoffrey & Smith, Nicholas, (2000). *Manual to Accompany the British National Corpus (Version 2) with Improved Word-class Tagging*. UCREL: Lancaster University. Available from <http://www.natcorp.ox.ac.uk/docs/bnc2postag_manual.htm>. Last accessed: 12-Aug-2015.
- Leech, Geoffrey & Weisser, Martin. (2013). The SPAADIA Annotation Scheme. Available from <http://martinweisser.org/publications/SPAADIA_Annotation_Scheme.pdf>
- Leech, Geoffrey, Weisser, Martin, Wilson, Andrew, & Grice, Martine. (2000). Survey and Guidelines for the Representation and Annotation of Dialogue. In Gibbon, Mertins, & Moore. (Eds.). (2000). *Handbook of Multimodal and Spoken Language Systems*. Dordrecht: Kluwer Academic Publishers.
- Leech, Geoffrey, Myers, Greg, & Thomas, Jenny. (Eds.). (1995). *Spoken English on Computer*. London: Longman.
- Legal Information Institute. (n.d.). U.S.C. : Title 17 – COPYRIGHTS. <<http://www.law.cornell.edu/uscode/text/17>>. [last accessed: 12-Aug-2015]
- Lindquist, Hans. (2009). *Corpus Linguistics and the Description of English*. Edinburgh: EUP.
- Lüdeling, Anke & Kytö, Merja. (Eds.). (2008). *Corpus Linguistics: An International Handbook*. Berlin: DeGruyter.
- McCarthy, Michael & O’Keeffe, Anne. Historical Perspective. In O’Keeffe, A. & McCarthy, M. (Eds.) (2010). *The Routledge Handbook of Corpus Linguistics*. London: Routledge.
- Manning, Christopher & Schütze, Hinrich. (1999). *Foundations of Statistical Natural Language Processing*. Cambridge, MA: MIT Press.
- McEnery, Tony & Hardie, Andrew. (2012). *Corpus Linguistics: Method, Theory and Practice*. Cambridge: CUP.
- Meyer, Charles. (2002). *English Corpus Linguistics: An Introduction*. Cambridge: CUP.
- Mikheev, Andrei. Text Segmentation. In Mitkov, R. (Ed.). (2003). *The Oxford Handbook of Computational Linguistics*. Oxford: OUP.
- Nattinger, James & DeCarrico, Jeanette. (1992). *Lexical Phrases in Language Teaching*. Oxford: OUP.
- Oakes, Michael. (1998). *Statistics for Corpus Linguistics*. Edinburgh: EUP.
- O’Keeffe, Anne & McCarthy, Michael. (Eds.) (2010). *The Routledge Handbook of Corpus Linguistics*. London: Routledge.
- Ooi, V. (1998). *Computer Corpus Lexicography*. Edinburgh: EUP.
- Paroubek, Patrick. (2007). Evaluating Part-Of-Speech Tagging and Parsing. In Dybkjær, L., Hensen, H., & Minker, W. (Eds.). (2007). *Evaluation of Text and Speech Systems*. Berlin/New York: Springer.
- Santorini, Beatrice. (1990). Part of Speech Tagging Guidelines for the Penn Treebank Project (3rd Revision, 2nd Printing).

- Scott, Mike. (1997). PC Analysis of Key Words – and Key Key Words. *System*, 25(2), pp. 233–245.
- Scott, Mike. (2010). Problems in Investigating Keyness, or Clearing the Undergrowth and Marking Out Trails... In Bondi & Scott. (Eds.). (2010). *Keyness in Texts*. Amsterdam: John Benjamins.
- Schmid, Helmut. (1994). Probabilistic Part-of-Speech Tagging Using Decision Trees. *Proceedings of International Conference on New Methods in Language Processing*, Manchester, UK.
- Sinclair, John. (1991). *Corpus, Concordance, Collocation*. Oxford: OUP.
- Sinclair, John. (2003) *Reading Concordances*. Harlow: Pearson Education Ltd.
- Sinclair, John. (2005). Corpus and Text – Basic Principles. In Wynne, M. (Ed.). *Developing Linguistic Corpora: A Guide to Good Practice*. Oxford: Oxbow Books: pp. 1–16. Available from <<http://www.ahds.ac.uk/creating/guides/linguistic-corpora/chapter1.htm>> [last accessed 12-Aug-2015].
- Stubbs, Michael. (1995). Collocations and Semantic Profiles: On the Cause of the Trouble with Quantitative Studies. *Functions of Language*, 2(1), pp. 23–55.
- Stubbs, Michael. (1996). *Text and Corpus Analysis*. Oxford: Blackwell.
- Taylor, A., Marcus, M., & Santorini, B. (2003). The Penn Treebank: An Overview. In Abeillé, A. *Treebanks*. Springer Netherlands. pp. 5–22.
- Thompson, H. & McKelvie, D. (1997). Hyperlink Semantics for Standoff Markup of Read-only Documents. In *Proceedings of SGML Europe'97*. Available from <<http://www.ltg.ed.ac.uk/~ht/sgmleu97.html>> [last accessed 12-Aug-2015].
- Tribble, Christopher. (2008). What Are Concordances and How Are They Used? In O'Keeffe, A. & McCarthy, M. (Eds.) (2010). *The Routledge Handbook of Corpus Linguistics*. London: Routledge.
- van Halteren, Hans. (Ed.). (1999). *Syntactic Wordclass Tagging*. Dordrecht: Kluwer Academic Publishers.
- Weisser, Martin. (2001). *A Corpus-Based Methodology for Comparing and Evaluating Native and Non-Native Speaker Accents*. Unpublished PhD thesis: Lancaster University.
- Weisser, Martin. (2009). *Essential Programming for Linguistics*. Edinburgh: EUP.
- Weisser, Martin. (2013). ICEweb (Version 1.0) [Computer Software]. Downloadable from http://martinweisser.org/ling_soft.html#iceweb.
- Weisser, Martin. (2014). The Dialogue Annotation and Research Tool (DART) (Version 1.0) [Computer Software]. Downloadable from http://martinweisser.org/ling_soft.html#DART.
- Weisser, Martin. (2014). The Simple Corpus Tool (Version 1.21) [Computer Software]. Downloadable from http://martinweisser.org/ling_soft.html#viewer.
- Weisser, Martin. (2014). The Simple PoS Tagger (Version 1.0) [Computer Software]. Downloadable from http://martinweisser.org/ling_soft.html#tagger.
- Wikipedia. (n.d.). Fair use. Available from <http://en.wikipedia.org/wiki/Fair_use>. [last accessed: 12-Aug-2015]
- WordBanks. (2009). Available from <http://wordbanks.harpercollins.co.uk/Docs/WBO/WordBanksOnline_English.html> [last accessed 12-Aug-2015].
- World Wide Web Consortium. (n.d.). Extensible Markup Language. <http://www.w3.org/XML/>.

- Wynne, Martin. (Ed.). (2005). *Developing Linguistic Corpora: A Guide to Good Practice*. Oxford: Oxbow Books. Available from <<http://www.ahds.ac.uk/creating/guides/linguistic-corpora/>> [last accessed 12-Aug-2015].
- Xiao, Richard. (2008). Well-known and Influential Corpora. In Lüdeling, A. & Kytö, M. (Eds.). (2008). *Corpus Linguistics: An International Handbook*. Berlin: DeGruyter.
- Zipf, George. (1949). *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*. Cambridge, MA: Addison-Wesley Press.

Index

- American National Corpus
(ANC). *See* Chapter 2
- annotation, 227
- morpho-syntactic, 77, 101,
112, 143, 228
- ditto tags, 148, 221
- pragmatic, 24, 228,
238–239
- speech acts, 24, 228, 238,
240, 245, 251, 252
- scheme, 101
- semantico-pragmatic, 228
- syntactic, 274
- c-unit, 194–195, 196, 215,
219, 237, 238
- labelled bracketing, 229
- s-unit, 201, 225
- tags
- HTML, 58, 64, 233
- HTML/XML, 37
- PoS, 101–105, 111–113,
122, 130–131, 135, 137,
142–144, 186, 188, 199,
218, 228
- SGML, 232
- XML, 201, 230–233,
236–239, 246, 249–251
- XML
- standoff, 248
- XML declaration, 236, 237,
241, 249
- AntConc. *See* software
- archives (compressed)
- .zip, 46, 51, 60–61, 63
- balance, 18, 20, 25, 27, 30–32,
160, 223, 271
- BNCweb. *See* web interfaces
- British National Corpus (BNC).
See Chapter 2
- BROWN Corpus. *See* Chapter 2
- Cascading Style Sheets (CSS),
229
- case sensitivity, 72, 78, 79
- character sets. *See* encoding
- Child Language Data Exchange
System (CHILDES), 256
- Chinese. *See* languages (natural)
- COCA. *See* web interfaces
- Colligation, 200, 203, 205, 207,
212, 218, 224–225
- collocation, 19, 29, 31, 67, 160,
196–198, 200, 203, 205,
207–212, 216, 220,
223–227, 255, 257, 272,
273, 274, 275
- node, 203, 208, 209, 210,
211, 220, 223, 225, 272,
273, 274
- compiling, 30, 43, 53
- computational linguistics, 256
- concordancers
- line-based, 69, 88, 204
- stream-based, 69, 88, 203
- context menu, 35, 45, 53, 61, 62
- corpora. *See* Chapter 2
- types
- diachronic, 15, 20, 25, 38,
271
- general, 21, 29, 32, 114,
122, 169, 255, 274

- corpora (*Continued*)
 monitor, 25, 30, 133, 178, 273
 snapshot, 25
 specific, 22
 synchronic, 15
 corpus (definition), 13
 Corpus of Contemporary American English (COCA). *See* Chapter 2
 co-text. *See* context
 CSS. *See* Cascading Style Sheets (CSS)
 CSS concepts
 attr(), 242, 245, 252, 253
 background-color, 235, 240, 252
 display, 234, 235, 240, 241–245, 249–253
 font-style, 235
 font-weight, 235
 line-height, 241, 252
 margin-left, 240, 241, 252
 pseudo-class, 242, 244, 245
 text-indent, 241, 252

 DART. *See* software
 data
 definition, 3
 datum, 3
 derived form, 33
 database management system (DBMS), 15
 de-lexicalised verbs, 220
 dialogue types
 task-oriented, 24, 207
 transactional, 25
 domain, 32, 153, 158, 169, 182

 elements (SGML/HTML/XML), 231
 encoding, 38–39, 41, 42, 46–48, 56, 185, 233, 234, 236, 249, 271, 275
 character sets, 14, 39, 41–42, 69, 85, 94, 233
 8-bit, 39
 ASCII, 39, 40, 94, 233, 271
 double-byte, 39
 Latin1, 39, 40
 legacy, 39
 non-Latin, 69
 Unicode, 14, 39, 176, 275
 UTF-8, 14, 39, 40, 42, 45–48, 185, 233, 236, 249
 English. *See* languages (natural)
 ethical/legal issues
 copyright, 4, 33, 44, 45, 58, 179
 fair use, 33
 quoting, 6
 surreptitious recording, 33

 file managers, 51, 52, 60, 61, 63, 76, 110
 file types/formats
 audio
 .mp3, 54
 .wav, 54
 extension, 37, 46, 47, 50, 71, 164, 165, 176
 proprietary, 37–38, 40, 42, 58, 229
 MS Word, 37, 58
 PDF, 37–38, 42, 45, 52, 58, 62, 65
 Postscript, 37
 Word Perfect, 37
 text
 .htm(l), 37
 .txt, 37, 46–48, 164, 171, 204
 .xml, 37
 plain text, 13, 37–38, 40–42, 46–48, 52, 58, 63, 66, 75, 78, 137, 230, 234
 forensic linguistics, 22
 formatting, 15, 37, 38, 41–45, 49, 50, 56, 59, 63, 64, 118, 166, 189, 190, 229, 231, 233–236, 241–242

 dashes, 6, 105, 115, 244
 m-dash, 7, 185
 n-dash, 57
 font, 14, 40, 235, 240, 244, 245, 252, 253
 hyphen, 84, 98, 115, 119, 143, 147, 154, 180–182
 issues, 6
 layout, 34, 37–38, 42, 44–45, 48, 63, 64, 179, 231, 233–234
 line breaks, 37, 47–49, 56, 63–64, 66, 69, 90, 118, 203, 231, 233, 238, 246, 250
 normalising, 56
 formulaic language/
 constructions, 19, 67, 97, 103, 147, 191–193, 197, 199, 200, 208, 220
 frequency
 absolute, 156
 frequency lists, 6, 50, 52, 57, 66, 67, 130, 149–179, 182–192, 198, 204, 221, 256, 274
 lexical density, 160
 n-grams, 31, 196–198, 203–207, 210, 215, 221–223, 227, 251
 lexical bundles, 196, 197, 205, 208
 lexical phrases, 196, 197
 trigrams, 222
 norming, 136, 175, 273
 relative, 133, 134, 144, 156, 173, 175–178, 189–190, 208, 226
 token, 52, 149–151, 153, 155, 170–171, 176, 180–181, 185, 187, 189, 191, 206, 222, 225–226, 275
 type, 149–151, 155–157, 169, 173, 175, 177, 179, 180, 183–185, 187–192, 201, 206, 222, 225–226, 275

- genre, 4, 32, 36, 43, 51, 112,
146, 150, 158, 167, 169,
171, 176, 184, 191–192,
212, 275
differences, 6
German. *See* languages (natural)
- hapax legomena, 183, 210,
223
- header. *See* text (divisions), 35
- header files, 247
- headword, 131–132, 134,
143–144, 149, 161–162,
273
- HTML (*Hypertext Markup
Language*), 229
- ICEweb. *See* software
- idioms. *See* formulaic language/
constructions, 147
- information retrieval, 160
- keyword-in-context (KWIC),
68–69, 73, 74, 76, 123,
137, 179, 203, 205, 219,
272, 273
- Korean. *See* languages (natural)
- KWIC. *See* keyword-in-context
- language
type
agglutinated, 113
right-to-left, 14
- language engineering, 256, 272
- languages (natural)
Arabic, 14
Bulgarian, 113
Chinese, 14, 20, 39, 94, 113,
148
Czech, 20, 183
Dutch, 113
English, 26, 29, 39, 40, 45,
50, 82, 84–86, 94, 102,
103, 112, 113, 147, 148,
154, 156, 195, 200, 203,
223, 233, 271–274
American, 16, 20, 29, 87,
122, 127, 133, 135–136,
144, 203, 222, 271–272
Australian, 16, 17, 271
British, 87, 122, 127, 130,
135–136, 144, 272–273
Early Modern, 21
Elizabethan, 224
for Academic Purposes
(EAP), 22, 160, 184, 272
for Specific Purposes (ESP),
22, 160, 184
Hong Kong, 18, 150
Indian, 16
International, 19, 20, 137
Middle, 15, 30
New Zealand, 16, 275
Old, 14–15, 25, 30
Estonian, 113
Finnish, 113
French, 113, 148
Galician, 113
German, 113, 148, 195
Greek, 14
Hebrew, 14
Hindi, 14
Indic, 14
Italian, 113
Japanese, 14
Korean, 20, 39, 113
Latin, 113
Mongolian, 113
Polish, 20, 113
Portuguese, 113
Russian, 113
Slovak, 113
Spanish, 113
Swahili, 113
Vietnamese, 14
- languages (programming)
Java, 83
Perl, 83, 88
Python, 83
- lemma, 131–137, 144, 150–151,
161–162, 198, 273
- lemma query, 135, 144
- lemmatisation, 150, 273
- lexicography, 29, 67, 79, 151
- LOB Corpus. *See* Chapter 2
- macros, 46–47
- markup, 37, 40, 49, 71, 77, 179,
227–234, 239, 272, 273,
274
- Document Object Model
(DOM), 233
- Document Type Definition
(DTD), 232, 272
- elements, 231–232, 234–238,
241, 243, 245, 249,
251–252
root element, 236
schema, 234
span, 233, 274
- markup concepts
angle brackets, 35, 41, 63,
179, 231
attribute, 65, 231–232, 234,
239, 241–242, 245, 251
block-level, 235, 241–242,
252
inline, 233, 235, 242, 245,
252–253
valid, 234, 236, 237
well-formed, 233, 234, 236,
251, 272
- meaning potential, 34, 138
- meta-information, 34, 138,
273
- speaker-related
age, 56, 184
education, 56
provenance, 56
sex, 56, 184
social class, 184
- modality
deontic, 192
epistemic, 192
- morphology
case, 113
compounds, 91, 129, 135,
147–148, 154, 183

- morphology (*Continued*)
 conversion/zero-derivation, 102
 pseudo-compounds, 6
 stem, 91–92, 100, 102, 184
 multi-word unit (MWU), 109, 148, 150, 219, 273
- natural language processing (NLP), 256
- norms, 36, 149
- phrase alternation, 128
- plain text. *See* file types/formats
- polysemy, 72, 102, 113, 123, 129, 137, 149, 150, 274
- PoS tagging. *See* annotation (morpho-syntactic)
- production, 30
- punctuation, 6, 85, 89–92, 94, 95, 104, 114, 115, 119, 128–129, 144, 147, 150, 185, 188, 194–196, 211, 213–214, 226, 228, 244–245, 248, 252, 255
- query, 122–124, 126, 128–129, 131, 134–135, 138, 140–141, 143, 199–201, 203, 210–211, 215–218, 220–222
- reception, 30
- regex, 47, 82, 100, 123, 126–128, 180, 196, 198, 200, 203–204, 216, 218, 221, 237, 250–251
- regex concepts
 alternation, 89, 128
 anchoring, 88
 caret, 86, 90
 backreference, 89, 237
 bracketing, 89, 141
 escaping, 90
 backslash, 88, 90
- greediness
 greedy, 90
 non-greedy, 90
- lookaround, 91
 lookahead, 91–92, 99
 lookbehind, 91, 100
- quantifiers, 86, 90, 98–99, 126, 141, 215
- register, 4
 appropriateness, 67
 differences, 4, 6
 types, 5
- repositories
 Oxford Text Archive (OTA), 26, 44, 45
 Project Gutenberg, 44, 58, 92, 110, 206
 text archives, 8, 44, 275
- representation, 185, 228
 electronic form, 44
 form, 7, 14, 38, 149
 orthographically transcribed, 1, 3, 15, 30, 54, 150, 226
 phonetic, 77
 phonetic transcription, 16
 transliterated, 3, 15
- representativeness, 18, 21, 24, 27–29, 31–32, 43, 67, 113, 130, 161, 205, 271, 274
- sampling, 22, 25, 26, 29, 30, 32, 137, 142, 160, 172, 184, 206, 211, 216
- search term, 57, 68, 73–75, 77, 79, 82–83, 88–89, 91–93, 99, 125, 128, 137, 160, 201–202, 205, 207, 208, 210, 211, 250, 273
 slots, 128–129, 134, 201, 203
- search-and-replace, 46–48, 56–57, 59, 65, 75, 81, 83, 89
- SGML (Standard Generalized Markup Language), 229
- Simple Corpus Tool. *See* software
- Simple PoS Tagger. *See* software
- Simple Query Syntax, 128, 131
- singletons. *See* hapax legomena
- software
 Adobe Acrobat, 58
 AntConc, 69–79, 92, 94, 96–97, 151–156, 158, 163, 169–171, 175, 181, 186, 187, 203, 205–207, 208–211, 221, 224, 236, 239, 251, 255
 DART (Dialogue Annotation and Research Tool), 251
 freeware, 7, 109
 ICEweb, 50–52, 60
 Simple Corpus Tool, 69, 204, 251
 Simple PoS Tagger, 111–112
- spreadsheets
 MS Excel, 60, 63, 156–166, 177, 186–189, 203
 OpenOffice Calc, 60, 156, 163–166, 177, 186–189, 203
- text editors, 13, 37, 46–50, 52, 54, 57, 75, 138, 236, 237
 gedit, 47
 KomodoEdit, 47
 Kwrite, 47
 Notepad++, 47, 118, 171, 221, 249
 TextEdit, 47
 TextWrangler, 47, 57
- word processors
 MS Word, 38, 42, 46, 52, 58, 63, 65
 OpenOffice Writer, 46, 58, 65
- spelling variants, 15, 95, 128–129, 140–141, 235
- spoken language features
 backchannels, 55, 239, 245, 251, 253
 discourse markers (DMs), 180, 182, 214, 219, 238, 239, 244, 246

- fillers, 15, 27, 31, 180, 182, 186
 h-dropping, 185, 196
 hesitation markers, 6, 55, 214
 minimal responses, 55
 overlap, 27, 55, 228, 239, 245, 251, 253
 turn, 55, 179, 194, 201–202, 214, 219, 233, 237, 239–242, 244, 245, 249–252
 stance, 192
 statistics
 chi-square(d), 169
 dispersion, 178, 206, 272
 expected frequency, 208
 mutual information (MI), 208–212, 224, 273
 observed frequency, 208
 t-score, 208, 209, 211, 224, 275
 stop words, 156, 158–160, 182
 stop word lists, 156, 158–160
 stylistics, 160
 stylometry, 160
 subcorpora, 34, 161–163, 167–169, 172–173, 176, 178, 179, 184, 188–190, 211, 275
 symbolic, 112
 taggers
 CLAWS, 105, 109, 111–112, 118–119, 131, 142–143, 183, 191, 218–219
 Simple PoS Tagger, 111–112
 TreeTagger, 112, 113
 tagging. *See* annotation (morpho-syntactic)
 tagsets, 103, 109, 111–114, 129, 131, 132, 143, 275
 CLAWS, 109, 111, 143, 219
 text
 divisions
 body, 35, 41, 59
 front and back matter, 34, 36
 header, 35, 40, 41, 56, 58–59, 138, 162, 230, 247, 272
 headings, 35–37, 48–50, 63–64, 115, 119, 150, 153, 177, 179, 230, 232, 233
 Text Encoding Initiative (TEI), 233, 248, 275
 text type, 4, 43, 60, 112, 152, 156, 179, 199, 275
 tokenisation, 113, 150, 183, 187, 191, 196, 255
 web interfaces
 BNCweb, 15, 34, 122–130, 132, 138, 142, 144, 147, 150, 160–169, 170, 172–174, 179, 185, 186, 188–191, 198–202, 205, 208–209, 210–211, 216, 218, 220, 224, 225, 230, 248, 255, 271
 COCA, 15, 25, 30, 121, 122, 132–137, 144, 160, 178–179, 202–205, 211–212, 222, 246, 248
 whitespace, 85, 87, 94, 95, 147, 150, 170, 246, 250
 wildcards, 126, 128, 129, 131, 134, 141, 198, 199, 201
 WWW, 8, 46, 229
 XML (eXtensible Markup Language), 230–239
 Zipf, 157, 208, 257