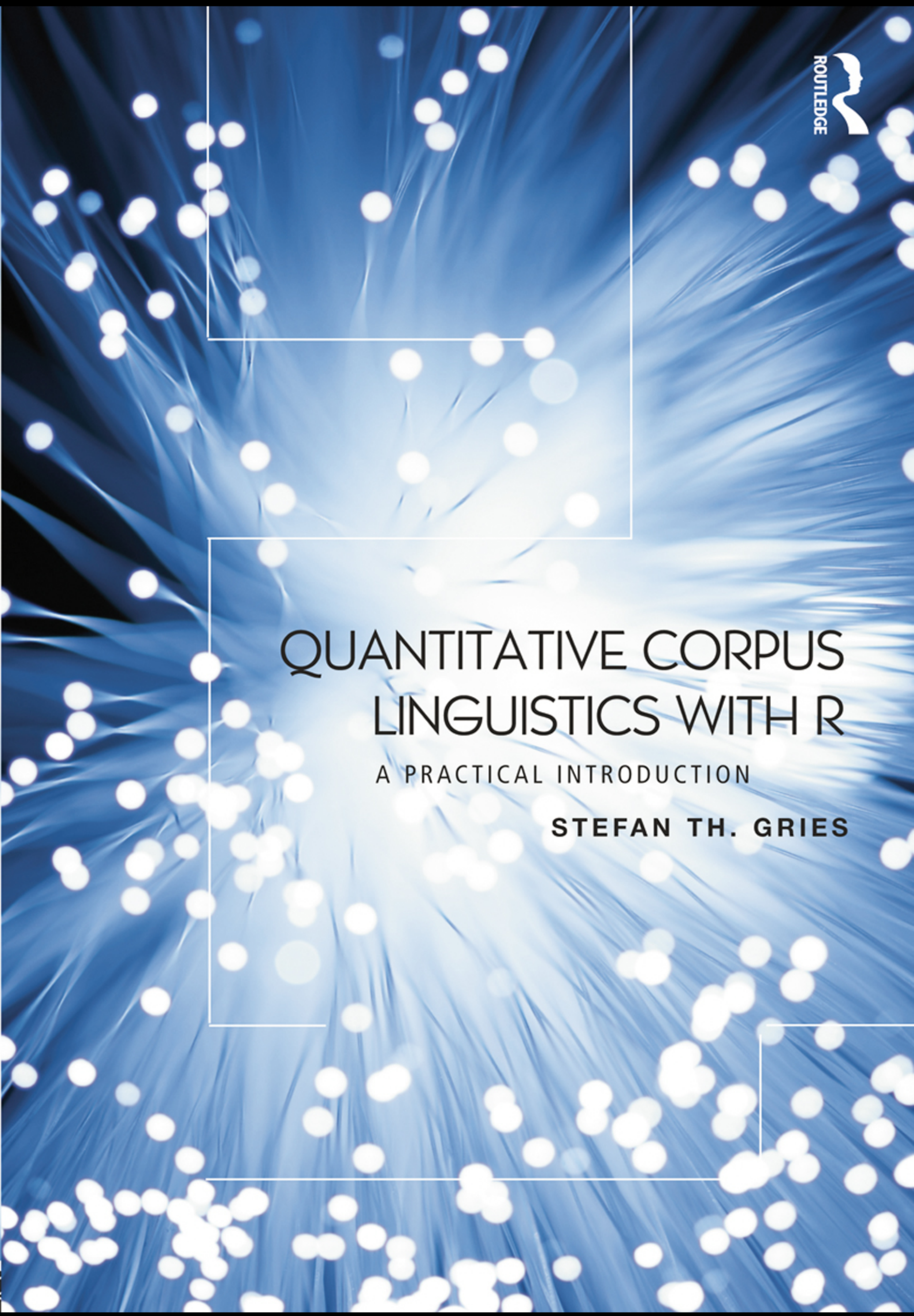

ROUTLEDGE

# QUANTITATIVE CORPUS LINGUISTICS WITH R

A PRACTICAL INTRODUCTION

STEFAN TH. GRIES

# Quantitative Corpus Linguistics with R

"Today's linguists are increasingly called upon to master technical and statistical domains for which they do not usually receive primary training in their graduate programs. This clear and highly approachable introduction is an invaluable guide both for seasoned linguists moving in a more quantitative, corpus-based direction as well as for a new generation of graduate students just beginning work in this field. Gries' book strikes a useful balance between the theoretical and the technical aspects of corpus linguistics, while making the case for why linguists should devote more attention to corpora and statistical methods in their own research and why R is the ideal tool for the job."

– Steven Clancy, University of Chicago, USA

"This is the book I've always wished existed. It is the ultimate guide to the wonderful world of R."

– Dagmar Divjak, University of Sheffield, UK

"*Quantitative Corpus Linguistics with R* is a most welcome contribution to the rapidly expanding field of empirically-oriented research in linguistics. Written in a clear and highly accessible style by a respected authority in the field, the book introduces innovative research practices in a step-wise, interactive fashion. It will be of great interest and enormous benefit to both researchers and advanced students who are looking for ways to perfect the art of data retrieval and evaluation."

– Doris Schönefeld, Ruhr University Bochum, Germany

"*Quantitative Corpus Linguistics with R* is radically different from other corpus linguistics textbooks. Gries' book offers a hands-on introduction to the methodological skills that are required to conduct sophisticated corpus linguistic analyses. An extremely useful book, this is the best practical introduction to the analysis of quantitative corpus data."

– Holger Diessel, Friedrich Schiller University of Jena, Germany

The first textbook of its kind, *Quantitative Corpus Linguistics with R* demonstrates how to use the open source programming language R for corpus linguistic analyses. Computational and corpus linguists doing corpus work will find that R provides an enormous range of functions that currently require several programs to achieve—searching and processing corpora, arranging and outputting the results of corpus searches, statistical evaluation, and graphing.

**Stefan Th. Gries** is Associate Professor of Linguistics at the University of California, Santa Barbara, USA.

# Quantitative Corpus Linguistics with R

A Practical Introduction

**Stefan Th. Gries**

Routledge
Taylor & Francis Group
NEW YORK AND LONDON

First published 2009
by Routledge
270 Madison Ave, New York, NY 10016

Simultaneously published in the UK
by Routledge
2 Park Square, Milton Park, Abingdon, Oxon OX14 4RN

*Routledge is an imprint of the Taylor & Francis Group, an informa business*

This edition published in the Taylor & Francis e-Library, 2009.

*Library of Congress Cataloging in Publication Data*
Gries, Stefan Thomas, 1970–
Quantitative corpus linguistics with R: a practical introduction / Stefan Th. Gries.
p. cm.
Includes bibliographical references and index.
[etc.]
1. Corpora (Linguistics) 2. Computational linguistics. 3. Linguistics—Statistical methods. I. Title.
P128.C68G75 2008
410.1—dc22
2008032294

ISBN 0-203-88092-7 Master e-book ISBN

ISBN10: 0–415–96271–4 (hbk)
ISBN10: 0–415–96270–6 (pbk)
ISBN10: 0–203–88092–7 (ebk)

ISBN13: 978–0–415–96271–1 (hbk)
ISBN13: 978–0–415–96270–4 (pbk)
ISBN13: 978–0–203–88092–0 (ebk)

# Contents

# Acknowledgments

I wish to thank the following people for input and support: Steven J. Clancy, Dagmar S. Divjak, Laura Kassner, and Gwillim Law for going over large parts of the book and useful and much appreciated suggestions for improvement, and Caroline V. David for discussion of an early draft of this book. Also, I am particularly grateful to two of the reviewers, one incredibly conscientious graduate student reviewer and one who turned out to be Randi Reppen, as well as to many students and participants of classes, summer schools, and workshops where parts of this book were used who provided much feedback over the last few years. Also, I thank Grabor Grothendieck for his incredibly useful R package `gsubfn` and much valuable input and discussion as well as Dani Bolliger and Heike Wagner for input regarding regular expressions and HTML. Last but certainly not least, I thank Stefanie Wulff for much feedback and invaluable support.

# 1
# Introduction

## 1.1 Why Another Introduction to Corpus Linguistics?

This book is an introduction to corpus linguistics. If you are a little familiar with the field, this probably immediately triggers the question "Why yet another introduction to corpus linguistics?" This is a valid question because given the upsurge of corpus-linguistic studies there are also already quite a few introductions available. Do we really need another one? Predictably, I think the answer is "Yes", and the reason is that this introduction is radically different from every other introduction to corpus linguistics that's out there. For example, there are a lot of things that are regularly dealt with at length in introductions to corpus linguistics that I will only be concerned with very little:

- the history of corpus linguistics: Kaeding, Fries, early 1m word corpora, up to the contemporary mega corpora and the lively web-as-corpus discussion;
- how to compile corpora: size, sampling, balancedness, representativity, . . .;
- how to create corpus markup and annotation: lemmatization, tagging, parsing, . . .;
- kinds and examples of corpora: synchronic vs. diachronic, annotated vs. unannotated;
- what kinds of corpus-linguistic research other people have done.

One important characteristic of this book is that I would like to teach you *how* to do corpus linguistics. This is important since, to me, corpus linguistics is a method of analysis and thus talking about *how* to do things should enjoy a high priority.[1] Therefore, as opposed to reporting many previous studies, I will be more concerned with:

- aspects of data retrieval: how to generate frequency lists, concordances, collocation displays, etc. (don't worry if you do not know these terms, they will be explained later);
- aspects of data evaluation: how to save your results; how to import them into a spreadsheet program for further annotation; how to analyze results statistically; how to represent the results graphically; how to report your results.

A second important characteristic of this book is that it only uses open source software:

- R, the corpus linguist's all-purpose tool (cf. R Development Core Team 2008): a software that is a calculator, a statistics program, a (statistical) graphics program, and a programming language at the same time; the version used in this book is R 2.8.0 for Microsoft Windows and Ubuntu Intrepid Ibex, but all versions after 2.2 should work fine nearly all of the time;
- Tinn-R for Windows 2.1.1.3, a text editor that is particularly suited for writing scripts for, and interacting with, R;
- OpenOffice.org Calc 3.0.

The choice of these software tools, especially the decision to use R, has a variety of important implications, which should be mentioned early. As I just mentioned, R is a full-fledged programming language and, thus, a very powerful tool. However, this degree of power does come at a cost: in the beginning, it is undoubtedly more difficult to do things with R than with ready-made free or commercial concordancing software that has been written specifically for corpus-linguistic applications. For example, if you want to generate a frequency list or concordance of a word in a corpus with R, you must write a small *script* or a little bit of *code* in R's programming language, which is the technical way of saying, you write lines of text that are instructions to R. If you do not need pretty output, this script may actually consist of just two lines, but it could be more. On the other hand, if you have a ready-made concordancer, you click a few buttons (and enter a search term) to get the job done. You may therefore think, "Why go through the trouble of learning a programming language such as R?" It turns out, there is a variety of very good reasons for this, some of them are related to corpus linguistics, some are not.

First, the effort that goes into writing a script usually needs to be undertaken only once. As you will see below, once you have written your first few scripts while going through this book, you can usually reuse them for many different tasks and corpora, and the amount of time that is required to perform a particular task becomes identical to that of using a ready-made program. I freely admit, for example, that nearly all corpus-linguistic retrieval tasks in my own research are done with (somewhat adjusted) scripts or small snippets of code from this book. As a matter of fact, once you explore how to write your own functions, you can easily write your own versatile or specialized concordance functions yourself (I will make one such function available below). This way, the actual effort of doing a frequency list, concordance, or a collocate display often reduces to about the time you need with a concordance program. In fact, R may even be faster than competing applications: for example, some concordance programs read in the corpus files once before they are processed and then again for performing the actual task—R requires only one pass and may, therefore, outperform its competitors in terms of processing time.

Second, by learning to do your analyses with a programming language, you usually have more control over what you are actually doing: different concordance programs have different settings or different ways of handling searches that are not always obvious to the (inexperienced) user. As you will see in one of the assignments below, when you do a concordance of the string "perl" in the file <C:/_qclwr/_inputfiles/corp_perl.txt> with the default settings in the programs AntConc 3.2.1w, WordSmith Tools 4.0, and MonoConc Pro 2.2, then AntConc finds 253 matches whereas WordSmith Tools and MonoConc Pro 2.2 find 248 matches. Users then not only face the problem of what to do with these conflicting results, but are then basically required to figure out why the counts differ or, put differently, how these programs have defined what a word is and how you can change their settings, etc. With a programming language, *you* are in the driver's seat: you define what a word is, and you define how a particular search is implemented for a particular search word and a particular corpus, and your results will be maximally replicable.

Third, if you use a particular concordancing software, you are at the mercy of its developer. If the developers change its behavior, its results output, or its default settings, you can only hope that this does not affect your results. Worse even, developers might even discontinue the development altogether . . . But there are also additional important advantages of the fact that R is an open source tool/programming language. For instance, there is a large number of functions that are contributed by users all over the world and often allow effective shortcuts that are not or hardly possible with ready-made applications, which you cannot tweak as you wish, and contrary to commercial concordance software, bug-fixes are available very fast.

Fourth and related to that, a programming language is a much more versatile tool than any ready-made software application. You will see many examples below where R provides more precise and/or customizable output than most available concordancers, or where R allows you to do things that regular concordance software could only do with a lot of manual effort or, more commonly, not at all. For example, there are many corpora or databases that come in formats that ready-made concordancers cannot handle at all but which, with just a small script, are easy to deal with in R. One case in point is the CELEX database, which provides a wealth of phonological information on English, German, and Dutch words, but comes in a format that is difficult or even impossible to use with spreadsheet software or concordancing programs but which is easy to handle in R (cf. Gries 2004b, 2006b for examples); another one is that many concordancers, even commercial ones, can't handle Unicode yet and are thus severely limited in terms of the languages they can be used with—R has no such problems (at least on MacOS X and Linux; Windows is unfortunately a bit of a different story). In a way, once you have mastered the basic mechanisms, there is basically no limit to what you can do with it both in terms of linguistic processing and statistical evaluation.

Fifth and also related to that, the knowledge you will acquire here is less specialized: you will not be restricted to just one particular software application (or even one version of one particular software application) and its restricted set of features. Rather, you will acquire knowledge of a programming language and regular expressions that will allow you to use many different utilities and to understand scripts in other programming languages, such as Perl or Python. (At the same time, R is simpler than Perl or Python, but can also interface with them via RSPerl and RSPython respectively; cf. <http://www.omegahat.org/>.) For example, if you ever come across scripts by other people or decide to turn to these languages yourself, you will benefit from knowing R in a way that no ready-made concordancing software would allow for. If you are already a bit familiar with corpus-linguistic work, you may now think, "But why turn to R and not use Perl or Python (especially since you say Perl and Python are similar anyway and many people already use one of these languages)?" This is a good question, and I can even tell you that sometimes Perl is even faster than R and I myself used Perl for a few applications (cf. Danielsson 2004 for a brief introduction to corpus linguistics using Perl). However, I think I also have a good answer why to use R instead. First, the issue of speed is much less of a problem than one may think. R is fast enough for most applications anyway and usually so stable that you can simply execute a script while you are in class or even over the night and collect the results afterwards. Second, R has other advantages. One is that, in addition to the text processing capabilities, R also offers a large number of ready-made functions for the statistical evaluation and graphical representation of data, which allow you to perform just about *all* corpus-linguistic tasks within only one programming environment: you can do your data processing, data retrieval, annotation, statistical evaluation, graphical representation . . . *everything* within just one environment whereas if you wanted to do all these things in Perl or Python, you would require a huge amount of separate programming. Consider a very simple example: R has a function called `table` that generates a frequency table. To perform the same in Perl you would either have to have a small loop counting elements in an array and in a stepwise fashion increment their frequencies in a hash or, more cleverly, to program a subroutine that you would then always call upon. While this is no problem with a one-dimensional frequency list, this is much harder with multidimensional frequency tables: Perl's arrays of arrays or hashes of arrays, etc. are not for the faint-hearted while R's `table` is easy to handle and additional functions (e.g., `xtabs`, `ftable`, . . .) allow you to handle such tables very easily. I believe learning one environment is sufficiently hard for beginners and therefore recommend using the more comprehensive environment with the greater number of simpler functions. And, once you have mastered the fundamentals of R and face situations where you do need maximal computational power, switching to Perl in a limited number of cases will be easier for you anyway, especially since much of the programming language's syntax is similar and the regular expressions used in this book are all Perl compatible. (Let me tell you, though, that in all my years using R, there were a mere two instances where I had to switch to Perl.)

A final, very down-to-earth advantage of using open source software is of course that it comes free of charge. Any student or any department's computer lab can afford it without expensive licenses, temporally limited or functionally restricted licenses, or irritating nag screens. All this, hopefully, makes a strong case for the choice of software.

## 1.2 Outline of the Book

This book is structured as follows. The next chapter, Chapter 2, defines the notion of a corpus and provides a brief overview of what I consider to be the three most central corpus-linguistic methods, namely frequency lists, collocations, and concordances. Chapter 3 then introduces you to the fundamentals of R, covering a variety of functions from different domains, but the area which receives most consideration is that of text processing. Chapter 4 illustrates how the methods introduced in Chapter 3 are applied to corpus data in the ways outlined in Chapter 2. Using a variety of different kinds of corpora, you will learn in detail how to write your own small programs in R for corpus-linguistic analysis. Chapter 5 introduces you to some fundamental aspects of statistical thinking and testing. The questions to be covered in this chapter are, What are hypotheses? How do I check whether my results are noteworthy?, etc.

Chapter 6 introduces you to a variety of applications. When you download all necessary files from the book's companion website, you will also download a set of case studies, each with between one and six assignments. These case studies introduce you to more realistic and, sometimes, more demanding corpus-linguistic applications. Some of these are based on published research; others exhibit features that are useful for corpus-linguistic studies and that are sometimes offered by other (commercial) software programs. The focus, however, is not on these other studies or programs—these studies serve as templates on the basis of which you get assignments and problems to solve by writing your own scripts to replicate the findings reported in a study or prepare the data for an independent project. As such, these case studies are from a wide array of applications: there are examples from morphology, syntax, semantics, and text linguistics; apart from concordancing, etc., you learn how to identify statistically significant key words in texts, use R to automatically access websites as corpus data, compute mean lengths of utterances for language acquisition research, and much more. In addition to the actual case studies, I also point you to a variety of domains of research, to which you can also apply the skills you learn from this book. Last, the appendices contain links to relevant websites, answer keys to all exercise boxes, and solutions to all assignments.

Before we begin, a few short comments on the nature of this book are necessary. As I already mentioned, this introduction to corpus linguistics is different from every other introduction I know—actually, it is the kind of introduction I always wanted to buy and could never find. This has two consequences. On the one hand, this book is not a book that requires much previous knowledge: it neither presupposes linguistic knowledge other than what you learn in your first introduction to linguistics nor mathematical or any programming knowledge.

On the other hand, this book is an attempt to teach you a lot about how to be a good corpus linguist. As a good corpus linguist, you have to combine many different concrete methodological skills (*and* many equally important analytical skills that I will not be concerned with here). Many of these methodological skills are addressed here, such as some very basic knowledge of computers (operating systems, file types, etc.), data management, regular expressions, some elementary programming skills, some elementary knowledge of statistics, etc. What you must know therefore is that (i) nobody has ever learned all this just by reading—you must *do* things—and (ii) this is *not* an easy book that you can read 10 minutes at a time in bed before you fall asleep. What these two things mean is that you should read this book while sitting at your computer and directly entering the code and working on the examples. You will need practice to master all the concepts introduced here, but will be rewarded by acquiring skills that give you access to a variety of data and approaches you may not have considered accessible to you—at least that's what happened to me. Undergraduates in my corpus classes without programming experience have quickly learned to write small programs that do things better than many concordancing programs, and you can do the same.

In order to facilitate your learning process, there are four different ways in which I try to help you get more out of this book. First, there are small think breaks. These are small assignments that you should try to complete before you read on, and answers to them follow immediately in the text. Second, there are exercise boxes with small assignments. Ideally, you should complete them and check your answers in the answer key before you read any further, but it is not always necessary to complete them to understand what follows so you can also return to them later at your own leisure. Third, there are many boxes with recommendations for further study/exploration. With perhaps one or two exceptions, they are just that, recommendations, which means you do not need to follow up on them to understand later material in the book—I just encourage you to follow them up anyway. Fourth and in addition to the above, I would like to mention that the companion website for this book is hosted at a Google group called "CorpLing with R", which I created and maintain. I would like to encourage you to go to the website of the newsgroup (the URL can be found in the Appendix) and become a member. Not only do you have to go to the companion website anyway to get all the files that belong to this book, but if you become a member of the list:

- you can send questions about corpus linguistics with R to the list and, hopefully, get useful responses from some kind soul(s);
- post suggestions for revisions of this book there;
- inform me and the other readers of errors you found and, of course, be informed when other people or I myself find errata.

Thus, while this is not an easy book, I hope these aids help you to become a good corpus linguist. If you work through the whole book, you will be able to do many things you could not even do with commercial concordancing software; many of the scripts you find here are taken from actual research, are in fact simplified versions of scripts I have used myself for published papers. In addition, if you also take up the many recommendations for further exploration that are scattered throughout the book you will probably find ever new and more efficient ways of application.

## 1.3 Recommendation for Instructors

If this book is not used for self-study but in a course, I would recommend using it in a two-quarter or two-semester sequence such that

- in the first quarter or semester, the course deals with the contents of Chapters 1 to 5, supplemented with additional readings if required;
- in the second quarter or semester, the students deal with assignments of Chapter 6, either all of them or, more likely, assignments that fit each student's personal interest and discuss them with the instructor.

Another possibility is to deal with Chapters 1 to 4 in the course of a quarter or semester and assign (parts of the) assignments in Chapter 6 as small homework assignments. Obviously, however, many more arrangements are conceivable.

# 2
# The Three Central Corpus-linguistic Methods

This last point leads me, with some slight trepidation, to make a comment on our field in general, an informal observation based largely on a number of papers I have read as submissions in recent months. In particular, we seem to be witnessing as well a shift in the way some linguists find and utilize data—many papers now use corpora as their primary data, and many use internet data.

(Joseph 2004: 382)

In this chapter, you will learn what a corpus is (plural: corpora) and what the three methods are to which nearly all corpus-linguistic work can be reduced.

## 2.1 Corpora

Before we start to actually look at corpus linguistics, we have to clarify our terminology a little. In this book, I will distinguish between three different things: a corpus, a text archive, and an example collection.

### 2.1.1 What is a Corpus?

In this book, the notion of "corpus" refers to a machine-readable collection of (spoken or written) texts that were produced in a natural communicative setting, and the collection of texts is compiled with the intention (1) to be representative and balanced with respect to a particular linguistic variety or register or genre and (2) to be analyzed linguistically. The parts of this definition need some further clarification themselves:

- "machine-readable" refers to the fact that nowadays virtually all corpora are stored in the form of plain ASCII or Unicode text files that can be loaded, manipulated, and processed platform-independently. This does not mean, however, that corpus linguists only deal with raw text files—quite the contrary: some corpora are shipped with sophisticated retrieval software that makes it possible to look for precisely defined syntactic and/or lexical patterns. It does mean, however, that you would have a hard time finding corpora on paper, in the form of punch cards or digitally in HTML or Microsoft Word document formats; the current standard is text files with XML annotation.

- "produced in a natural communicative setting" means that the texts were spoken or written for some authentic communicative purpose, but not for the purpose of putting them into a corpus. For example, many corpora consist to a large degree of newspaper articles. These meet the criterion of having been produced in a natural setting because journalists write the article to be published in newspapers and to communicate something to their readers, but not because they want to fill a linguist's corpus. Similarly, if I obtained permission to record all of a particular person's conversations in one week, then hopefully, while the person and his interlocutors usually are aware of their conversation being recorded, I will obtain authentic conversations rather than conversations produced only for the sake of my corpus.
- I use "representative [. . .] with respect to a particular variety" here to refer to the fact that the different parts of which the linguistic variety I am interested in are all manifested in the corpus. For example, if I was interested in phonological reduction patterns of speech of adolescent Californians and recorded only parts of their conversations with several people from their peer group, my corpus would not be representative in the above sense because it would not reflect the fact that some sizable proportion of the speech of adolescent Californians may also consist of dialogs with a parent, a teacher, etc., which would therefore also have to be included.
- I use "balanced with respect to a particular linguistic variety" to mean that not only should all parts of which a variety consists be sampled into the corpus, but also that the proportion with which a particular part is represented in a corpus should reflect the proportion the part makes up in this variety and/or the importance of the part in this variety. For example, if I know that dialogs make up 65 percent of the speech of adolescent Californians, approximately 65 percent of my corpus should consist of dialog recordings. This example already shows that this criterion is more of a theoretical ideal: How would one measure the proportion that dialogs make up of the speech of adolescent Californians? We can only record a tiny sample of all adolescent Californians, and how would we measure the proportion of dialogs—in terms of time? in terms of sentences? in terms of words? And how would we measure the importance of a particular linguistic variety? The implicit assumption that conversational speech is somehow the primary object of interest in linguistics also prevails in corpus linguistics, which is why many corpora aim at including as much spoken language as possible, but on the other hand a single newspaper headline read by millions of people may have a much larger influence on every reader's linguistic system than twenty hours of dialog. In sum, balanced corpora are a theoretical ideal that corpus compilers constantly bear in mind, but the ultimate and exact way of compiling a balanced corpus has remained mysterious so far.

Many people consider a text archive to be something different, namely a database of texts

- that may not have been produced in a natural setting;
- that has often not been compiled for the purposes of linguistic analysis;
- that has often not been intended to be representative with respect to a particular linguistic variety or speech community;

- that has often not been intended to be balanced with respect to a particular linguistic variety or speech community.

However, the distinction between the corpora and text archives is often blurred. It is theoretically easy to make, but in practice often not adhered to very strictly. For example, if a publisher of a popular computing periodical makes all the issues of the previous year available on his website, then the first criterion is met, but not the last three. However, because of their availability and their size, many corpus linguists use them as resources, and as long as one bears their limitations in terms of representativity, etc. in mind, there is little reason not to do that.

Finally, an *example collection* is just what the name says it is, namely a collection of examples that, typically, the person who compiled the examples came across and noted down. For example, much psycholinguistic research in the 1970s was based on collections of speech errors compiled by the researchers themselves and/or their helpers. Occasionally, people refer to such collections as error corpora but we will not use the term *corpus* for these. It is easy to see how such collections compare to corpora. On the one hand, for example, an error collection may fail to be representative and balanced because some errors—while occurring frequently in authentic speech—are more difficult to perceive than others and thus hardly ever make it into a collection. This would be an analog to the balancedness problem outlined above. On the other hand, the perception of errors is contingent on the acuity of the researcher while, with corpus research, the corpus compilation would not be contingent on a particular person's perceptual skills. Finally, because of the scarcity of speech errors, usually all speech errors perceived (in a particular amount of time) are included into the corpus whereas, at least usually and ideally, corpus compilers are more picky and select the material to be included with an eye to the criteria of representativity and balancedness outlined above.[1] Be that as it may, for the sake of terminological clarity, we will carefully distinguish the notions of corpora, text archives, and example collections and focus only on the first category.

### 2.1.2 What Kinds of Corpora are There?

Corpora differ in a variety of ways. There are a few distinctions you should be familiar with if only to be able to find the right corpus for what you want to investigate. The most basic distinction is that between *general corpora* and *specific corpora*. The former intend to be representative and balanced for a language as a whole—within the above-mentioned limits, that is—while the latter are by design restricted to a particular variety, register, genre, . . .

Another important distinction is that between *raw corpora* and *annotated corpora*. Raw corpora consist of files only containing the corpus material (cf. (1)) while annotated corpora in addition also contain additional information. Annotated corpora are very often annotated according to the standards of the Text Encoding Initiative (TEI) or the Corpus Encoding Standard (CES), and then have two parts. The first part is called the header, which provides information that is typically characterized as markup. This is information about (i) the text itself, e.g. where the corpus data come from, which language is represented in the file? which (part of a) newspaper or book has been included? who recorded whom where and when? who has the copyright? which annotation comes with the file? and information about (ii) its formatting, printing, processing, etc. Markup refers to objectively codable information—the fact that there is a paragraph in a text or that a particular speaker is female can be made without doubt. This information helps users to quickly determine whether a particular file is part of the register one wishes to investigate or not.

The second part is called the body and contains the corpus data proper—i.e. what people actually said or wrote—as well as linguistic information that is usually based on some linguistic theory: parts of speech or syntactic patterns, for example, can be matters of debate. In what follows

I will briefly (and non-exhaustively!) discuss and exemplify a few common annotation schemes (cf. Leech's and McEnery and Xiao's articles in Wynne 2005 as well as McEnery, Xiao, and Tono 2006: A.3 and A.4 for more discussion).

First, a corpus may be *lemmatized* such that each word in the corpus is followed by its lemma, i.e. the form under which you would look it up in a dictionary (cf. (2)). A corpus may have so-called *part-of speech tags* so that each word in the corpus is followed by an abbreviation giving the word's part of speech (POS) and sometimes also some morphological information (cf. (3)). A corpus may also be phonologically annotated (cf. (4)). Then, a corpus may be *syntactically parsed*, i.e. each word is followed by an abbreviation giving the word's syntactic status (cf. (5)). Finally and as a last example, a corpus may contain several different annotations on different lines (or tiers) at the same time, a format especially common in language acquisition corpora (cf. (6)).

```
(1)  I did get a postcard from him.
(2)  I<I> did<do> get<get> a<a> postcard<postcard> from<from> him<he>.<punct>
(3)  I<PersPron> did<VerbPast> get<VerbInf> a<Det> postcard<NounSing> from<Prep>
     him<PersPron>.<punct>
(4)  [@:] • I • ^did • get • a • !p\ostcard • fr/om • him# • - • -
(5)  <Subject, • NP>
          I<PersPron>
     <Predicate, • VP>
          did<Verb>
          get<Verb>
          <DirObject, • NP>
                a<Det>
                postcard<NounSing>
          <Adverbial, • PP>
                from<Prep>
                him<PersPron>.
(6)  *CHI:     I did get a postcard from him
     %mor:
     pro|I • v|do&PAST • v|get • det|a • n|postcard • prep|from • pro|him • .
     %lex:     get
     %syn:     trans
```

Other annotation includes annotation with regard to semantic characteristics, stylistic aspects, anaphoric relations (coreference annotation), etc. Nowadays, most corpora come in the form of SGML or XML files, and we will explore many examples involving these formats in the chapters to come. As is probably obvious from the above, annotation can sometimes be done completely automatically (possibly with human error checking), semi-automatically, or must be done completely manually. Part-of-speech tagging, probably the most frequent kind of annotation, is usually done automatically, and taggers are claimed to achieve accuracy rates of 97 percent, a number that I sometimes find hard to believe when I look at corpora, but that is a different story.

Then, there is a difference between *diachronic corpora* and *synchronic corpora*. The former aims at representing how a language/variety changes over time while the latter provides, so to speak, a snapshot of a language/variety at one particular point of time. Yet another distinction is that between *monolingual corpora* and *parallel corpora*. As you might already guess from the names, the former have been compiled to provide information about one particular language/variety, etc. whereas the latter ideally provide the same text in several different languages. Examples include translations from EU Parliament debates into the 23 languages of the European Union or the Canadian Hansard corpus, containing Canadian Parliament debates in English and French. Again ideally, a parallel corpus does not just have the translations in different languages, but has the translations sentence-aligned, such that for every sentence in language $L_1$, you can automatically retrieve its translation in the languages $L_2$ to $L_n$.

The next distinction to be mentioned here is that of *static corpora* vs. *dynamic/monitor corpora.* Static corpora have a fixed size (e.g. the Brown corpus, the LOB corpus, the British National Corpus) whereas dynamic corpora do not, since they may be constantly extended with new material (e.g. the Bank of English).

The final distinction I would like to mention at least briefly involves the encoding of the corpus files. Given especially the predominance of work on English in corpus linguistics, until rather recently, many corpora came in the so-called ASCII (American Standard Code for Information Interchange) character encoding, an encoding scheme that encodes $2^7 = 128$ characters as numbers and that is largely based on the English characters. With these characters, special characters that were not part of the ASCII character inventory (e.g. the "é" was paraphrased as "&eacute;"). However, the number of corpora for many more languages has been increasing steadily and given the large number of characters that writing systems such as Chinese have, this is not a practical approach, language-specific character encodings were developed (e.g. ISO 8859-1 for Western European Languages vs. ISO 2022 for Chinese/Japanese/Korean languages). However, in the interest of overcoming compatibility problems that arose because now different languages used different character encodings, the field of corpus linguistics is currently moving towards using only one unified (i.e. not language-specific) multilingual character encoding, Unicode (most notably UTF-8). This development is in tandem with the move towards XML corpus annotation and, more generally, UTF-8's becoming the most widely used character encoding on the WWW (cf. <http://googleblog.blogspot.com/2008/05/moving-to-unicode-51.html>).

Now that you know a bit about the kinds of corpora that are out there, there is one other really important point to be made. While we will see below that corpus linguistics has a lot to offer to the analyst, it is worth pointing out that, strictly speaking at least, the only thing corpora can provide is information on frequencies. Put differently, there is no meaning in corpora, and there are no functions in corpora—corpora only contain

- frequencies of occurrence, i.e. how often morphemes, words, grammatical patterns, etc. occur in (parts of) a corpus; or
- frequencies of co-occurrence of the same kinds of items, i.e. how often do morphemes occur with particular words? how often do particular words occur in a certain grammatical construction? etc.

. . . and it is up to the researcher to interpret frequencies of occurrence and co-occurrence in meaningful or functional terms. The assumption underlying basically all corpus-based analyses, however, is that, since formal differences correspond to functional differences, different frequencies of (co-)occurrences of formal elements are supposed to reflect functional regularities. "Functional" is understood here in a very broad sense as anything—be it semantic, discourse-pragmatic, . . .—that is intended to perform a particular communicative function. On a very general level, the frequency information a corpus offers is exploited in three different ways, which will be the subject of this chapter: frequency lists (cf. Section 2.2), lexical co-occurrence lists or collocations (cf. Section 2.3), and concordances (cf. Section 2.4).

## 2.2 Frequency Lists

The most basic corpus-linguistic tool is the frequency list. You generate a frequency list when you want to know how often words occur in a corpus. Thus, a frequency list of a corpus is usually a two-column table with all words occurring in the corpus in one column and the frequency with which they occur in the corpus in the other column. Since the notion of *word* is a little ambiguous here, it is useful to introduce a common distinction between (word) type and (word) token. The string "the word and the phrase" contains five (word) tokens ("the", "word", "and", "the", and "phrase"), but only four (word) types ("the", "word", "and", and "phrase"), of which one occurs twice. In this parlance, a frequency list lists the types in one column and their token frequencies in the other. Typically, one out of three different sorting styles is used: frequency order (ascending or, more typically, descending; cf. the left panel of Table 2.1), alphabetical (ascending or descending), and occurrence (each word occurs in a position reflecting its first occurrence in the corpus).

Apart from this simple form, there are two other varieties of frequency lists that are sometimes found. On the one hand, a frequency list may provide the frequencies of all words together with the words with their letters reversed. This may not seem particularly useful at first, but even a brief look at the central panel of Table 2.1 clarifies that this kind of display can sometimes be very helpful because it groups together words that share a particular suffix, here the adverb marker -*ly*. On the other hand, a frequency list may not list each individual word token and their frequencies but so-called *n*-grams, i.e. sequences of *n* words such as bigrams (where $n = 2$, i.e. word pairs) or trigrams (where $n = 3$, i.e. word triples) and their frequencies; cf. the right panel of Table 2.1 for an example involving bigrams.

Table 2.1 Examples of differently ordered frequency lists

| Words | Freq | Words | Freq | Bigrams | Freq |
|---|---|---|---|---|---|
| the | 62,580 | yllufdaerd | 80 | of the | 4,892 |
| of | 35,958 | yllufecaep | 1 | in the | 3,006 |
| and | 27,789 | yllufecarg | 5 | to the | 1,751 |
| to | 25,600 | yllufecruoser | 8 | on the | 1,228 |
| a | 21,843 | yllufeelg | 1 | and the | 1,114 |
| in | 19,446 | yllufeow | 1 | for the | 906 |
| that | 10,296 | ylluf | 2 | at the | 832 |
| is | 9,938 | yllufepoh | 8 | to be | 799 |
| was | 9,740 | ylluferac | 87 | with the | 783 |
| for | 8,799 | yllufesoprup | 1 | from the | 720 |

Frequency lists are sometimes more problematic than they seem, because they presuppose that the linguist (and/or his computer program) has a definition of what a word is and that this definition is shared by other linguists (and their computer programs). This need not be the case, however, as the following exercise will demonstrate.

**Exercise box 2.1: What is a word, or what does your computer think a word is?**

(1) Write up a plain English definition of how you would "tell a computer program" what a word is. When you are finished, do the next exercise.

(2) How does your definition handle the expressions *better-suited* and *ill-defined*? How does it handle *armchair-linguist* and *armchair linguist* and *armchair-linguist's*? How does it handle *http://www.linguistics.ucsb.edu*? How does it handle *This* and *this*? How does it handle *1960* and *25-year-old*? And *favor* and *favour*? And *Peter's car*, *Peter's come home*, and *Peter's sick*? And what about позволено (Cyrillic) or 宜蘭童玩節停辦 (Chinese) in an English text?

Bear in mind, therefore, that you need to exercise caution in comparing frequency lists from different sources.[2] Another noteworthy aspect is that frequency lists are often compiled so as not to include words from a so-called stop list. For example, one can often exclude a variety of frequent function words such as *the*, *of*, *and*, etc., because these are often semantically not particularly revealing since they occur nearly indiscriminately throughout the corpus.[3]

For what follows below, it is useful to introduce the distinction between a lemma and a word-form: *go*, *goes*, *going*, *went*, and *gone* are all different word forms even though they belong to the same lemma, namely *go*; in the remainder of this chapter, I will only use *word* where the difference between *lemma* and *word form* is not relevant. Computer programs normally define word forms as a sequence of alphabetic (or alphanumeric) characters uninterrupted by whitespace (i.e. spaces, tabs, and newlines) and allow the user to specify what to do with hyphens, apostrophes, and other special characters.

Frequency lists are useful for a variety of purposes. For example, much contemporary work in usage-based linguistics assumes that the type and token frequencies of linguistic expressions are correlated with the degree of cognitive entrenchment of these expressions, and studies such as Bybee and Scheibman (1999) and Jurafsky et al. (2000) related frequencies of expressions to their readiness to undergo processes of phonological reduction and grammaticalization. For example, Bybee and Scheibman (1999) show that the vowel in *don't* is more likely to be reduced to schwa in frequent expressions such as *I don't know* (as opposed to, say, *They don't integrate easily*). In psycholinguistics, models of language production have long incorporated frequency effects, but there is now a growing recognition that information of percentages, conditional probabilities, etc. of all kinds are represented in the linguistic systems of speakers and, thus play a primary role at all levels of linguistic analysis. On the more practical side of things, word frequency lists are useful to choose experimental stimuli correctly: for example, the logs of frequency of words are related to psycholinguistic measures such as reaction times to these words (in priming or naming studies), which often makes it necessary to control for such effects in order not to bias your reaction times. Also, you may want to do a study on tip-of-the-tongue states and, thus, make sure the words you use are sufficiently infrequent.

A much more applied example which we will also be concerned with below involves the identification of words that are (significantly) overrepresented in one corpus as compared to a balanced reference corpus. This can be used to identify the words that are most characteristic of a particular domain: a word that is about equally frequent in a particular corpus and a larger reference corpus is probably not very revealing in terms of what the smaller corpus is about. However, if a corpus contains the words *economic*, *financial*, *shares*, *stocks* much more often than a balanced reference corpus, guessing the topics covered in the smaller corpus is straightforward.

This kind of approach can be used to generate lists of important vocabulary for language learners.

In the domain of natural language processing or computational linguistics, the frequency of items is relevant to, among other things, speech recognition. For example, imagine a computer gets ambiguous acoustic input in a noisy environment and tries to recognize which word is manifested in the input. If the computer cannot identify the input straightforwardly, one (simplistic) strategy would be for it to assume that the word it hears is the most frequent one of all those that are compatible with the acoustic input. Another area of interest is, for example, spelling error correction, where frequency lists can be useful in two ways: first, for the computer to recognize that a string is probably a typo because it does not occur in a gold standard frequency list of words of the input language; second for the computer to rank suggestions for correction such that the computer first determines a set of words that are sufficiently similar to the user's input and then ranks them according to their similarity to the input and their frequency. From a methodological perspective, frequency lists are useful for computing many co-occurrence statistics. Finally, as will be obvious from the set of reading recommendations below, frequency lists may reflect sociocultural differences.

## 2.3 Lexical Co-occurrence: Collocations

One of the most central methodological concepts is that of co-occurrence. Corpus linguists have basically been concerned with three different kinds of co-occurrence phenomena:

- collocation: the probabilistic co-occurrence of word forms such as *different from* vs. *different to* vs. *different than*; or the absolute frozenness of expressions such as *kith and kin* or *by and large*;
- colligation: the co-occurrence of word forms with grammatical phenomena such as part-of-speech categories, grammatical relations, or definiteness such as the preference of *consequence* to occur as a complement (but not an adverbial) and with an indefinite article;
- (grammar) patterns or collostructions: the co-occurrence of words/lemmas with morpho-syntactic patterns or constructions such as the ditransitive construction or the cleft construction such as the preference of *to hem* to occur in the passive; or the association of the ditransitive to forms of the verb *to give*.

In this section, we will restrict ourselves to collocations because they are a natural extension of frequency lists.

Collocations are co-occurrences of words, which are then referred to as *collocates*. Often, one uses the letters L and R (for left and right) together with a number indicating the position of one collocate with respect to the other to talk about collocations. For example, if you call up a concordance of the word *difference*, then you will most certainly find that the most frequent L1 collocate is *the* while the most frequent R1 collocate is *between*. Thus, a collocate display for a word tells you which other words occur at different positions around the target word and how frequently. A *collocate* display for a word, accordingly, is usually a table with several columns such that each column lists the collocates of the target word at a particular position together with their frequencies. In other words, a collocate display is a list of frequency lists for particular positions around a word. To look at a simple example with two adjectives, consider the two adjectives *alphabetic* and *alphabetical*.

Table 2.2 is a collocate display of *alphabetic*; the strings in angular brackets are word class tags (cf. the Appendix for pointers to lists of all tags used in this book), and Table 2.3 is a collocate

Table 2.2 A collocate display of *alphabetic* based on the BNC

| Word at L1 | Freq L1 | Node word | Freq Node | Word at R1 | Freq R1 |
|---|---|---|---|---|---|
| <w prf>of | 8 | <w aj0>alphabetic | 42 | <w nn1>literacy | 7 |
| <w at0>the | 6 | | | <w nn1>writing | 5 |
| <w at0>an | 5 | | | <w nn1>order | 3 |
| <w prp>in | 2 | | | <w nn1>character | 3 |
| <w prp>such as | 2 | | | <w cjc>and | 2 |
| <w dps>our | 2 | | | <w nn1>system | 2 |
| <w cjs>when | 2 | | | <w nn2>characters | 2 |
| <w aj0 widespread> | 1 | | | <w nn1>culture | 2 |
| <w nn2>systems | 1 | | | <w prp>in | 1 |
| <w aj0>varying | 1 | | | <c pun>. | 1 |

Table 2.3 A collocate display of *alphabetical* based on the BNC

| Word at L1 | Freq L1 | Node word | Freq Node | Word at R1 | Freq R1 |
|---|---|---|---|---|---|
| <w prp>in | 77 | <w aj0>alphabetical | 234 | <w nn1>order | 89 |
| <w at0>an | 36 | | | <w nn1>index | 15 |
| <w at0>the | 23 | | | <w nn1>list | 13 |
| <w prf>of | 6 | | | <w nn1>indexing | 12 |
| <w cjc>and | 6 | | | <w nn1>subject | 12 |
| <c pun>. | 6 | | | <w nn1>sequence | 11 |
| <c pun>, | 6 | | | <w nn1>listing | 9 |
| <w aj0>ascending | 5 | | | <w nn1>guest | 6 |
| <w cjc>or | 5 | | | <w cjc>and | 5 |
| <w aj0>strict | 4 | | | <w nn1>description | 2 |

display of *alphabetical*; both are based on data from the British National Corpus World Edition (BNC).

Note that a collocate display is read vertically: Row 1 of Table 2.2 does not tell you anything about the string *of alphabetic literacy*—it tells you that *of* is the most frequent word at L1 (occurring there eight times) and that *writing* is the second most frequent word at R1 (with five occurrences). Now, if we compare the R1 collocates of *alphabetic* and *alphabetical*, can you already discern a tendency of the specific semantic foci of the two adjectives?

**THINK**

**BREAK**

The difference between the two adjectives can probably be paraphrased easiest by stating what the opposites of the two adjectives are. My suggestion would be that the opposite of *alphabetic* is *numeric* whereas the opposite of *alphabetical* is *unordered*, but a more refined look at the data may reveal a more precise picture.

Collocate displays are an important tool within semantics and lexicography (cf. Sinclair 1987, Stubbs 1995, Kilgarriff and Tugwell 2001), as well as language teaching (cf. Lewis 2000); the phenomenon of near synonymy is a particularly fruitful area of application. Consider for example the English adjectives *fast*, *quick*, *rapid*, and *swift*. While we would be hard-pressed to intuitively identify the difference between these semantically extremely similar expressions, inspecting the position R1 in collocate displays for these adjectives would give us a very good clue as to the nouns these adjectives usually modify attributively and exhibit semantic or other distributional regularities.

Another area of application is what has been referred to as semantic prosody, i.e. the fact that collocates of some word *w* may imbue *w* with a particular semantic aura even though this aura is not part of the semantics of *w* proper. One of the standard textbook examples is the English verb *to cause*. As you probably sense intuitively, *to cause* primarily, though not exclusively, collocates with negative things (*problem*, *damage*, *harm*, *havoc*, *distress*, *inconvenience*, etc.) although causing as such need not be negative. This is not only a theoretically interesting datum, it also has implications for, say, research on irony and foreign language teaching since, for example, if a foreign language learner uses a word *w* without being aware of *w*'s semantic prosody, this may result in comical situations or, more seriously, communicating unwanted implications.

## 2.4 (Lexico-)Grammatical Co-occurrence: Concordances

However useful collocate displays are, for many kinds of analysis they are still not optimal. On the one hand, it is obvious that collocate displays usually provide information on lexical co-occurrence, but the number of grammatical features that is amenable to an investigation by means of collocates is limited. On the other hand, even the investigation of lexical co-occurrence by means of collocate displays can be problematic. If you investigate near synonymous adjectives such as *big*, *great*, and *large* (or *deadly*, *fatal*, and *lethal*) by looking at R1, you reduce both the precision and the recall of your results:

- precision is defined as the quotient of the number of accurate matches returned by your search divided by the number of all matches returned by your search. The collocate approach reduces precision because the R1 collocate of *big* in *the big and mean guy* is *and* rather than *guy*, and the inclusion of *and*, while of course accurate, doesn't tell you much about the semantics of *big*;[4]
- recall is defined as the number of accurate matches returned by your search divided by the number of all possible accurate matches in the data. The R1 collocate approach may reduce recall because, as we have seen in the example above, you miss *guy* in *big and mean guy* or in *the guy is big*.

The final method I am going to introduce here gets rid of this problem, though at a cost. This method, probably the most widespread corpus-linguistic tool until now, is the *concordance*. You generate a concordance if you want to know in which (larger) contexts a particular word is used. Thus, a concordance of the word *w* is a display of every occurrence of *w* together with a user-specified context; it is often referred to as KWIC (= Key Word In Context) display.[5] This user-specified context is normally either the whole sentence in which the word in question occurs (usually with some highlighting or bracketing; cf. Table 2.4.), or the word in question in a central column together with a user-specified number of words or characters to its left and its right (cf. Table 2.5 for an example). The display in Table 2.5 is especially useful because it can be sorted in various ways, such as according to the words at R1. Obviously, unlike collocate displays, concordances are read horizontally just like normal (parts of) sentences, which is in fact what they show.

While the concordance displays in Tables 2.4 and 2.5 are not always easily exploitable, they are maximally comprehensive: you can look at your search word and every possible linguistic element and how (frequently) it co-occurs with it (not just words as in the collocate display), but the price you have to pay is that you cannot usually extract this information semi-automatically as in the

Table 2.4 An example display of a concordance of *before* and *after* (sentence display)

| # | Match |
|---|---|
| 1 | at that time and erm, we had a lot of German shutters and cameras in museum [[before]] September the third on September the fourth when I got to work they were all out. |
| 2 | It will be easy enough to bleach them with some Milk of Magnesia the night [[before]] he comes home. |
| 3 | And as, as we said [[before]], erm, many of the, erm people who lived in the poorer parts of erm the country, whether in urban or rural En erm England didn't really know about basic nutrition and and health I mean you just |
| 4 | They should be kept in an airtight jar, and rinsed in methylated spirits [[before]] wearing. |
| 5 | Yes if you looked [[after]] a child. |
| 6 | And erm some immediately post-war recipe books and I'm sure you know if you'd like to look at them [[after]] I've finished talking you might even remember some of the er the er My wife still uses the. |
| 7 | This was covered by a biscuit tin lid on top of which was a kettle filled with water for an [[after]] dinner cup of tea and the washing up. |
| 8 | [[After]] the blitz on London in September nineteen forty, the government introduced a scheme whereby payments for damage to the furniture of persons earning less than four hundred pounds a year would be made, up to one hundred percent of th |
| 9 | There was a campaign [[after]] the war to bring back Tottenham puddings they had somebody in Harlow who was Well what was, what, what was Tottenham pudding then. |
| 10 | [[After]] Japan's entry into the war all imports of rubber from the far east were suspended. |

Table 2.5 An example display of a concordance of *before* and *after* (tabular)

| L1 | Node | R1 | R2 |
|---|---|---|---|
| museum | before | September | the |
| night | before | he | comes |
| said | before | erm | many |
| spirits | before | wearing | |
| looked | after | a | child |
| them | after | I | 've |
| an | after | dinner | cup |
| | After | the | blitz |
| campaign | after | the | war |
| | After | Japan | 's |

collocate display. In other words, concordances usually need manual analysis and annotation: In the example of *big and mean guy* from above, for example, if your corpus is not syntactically annotated, you must read the concordance line(s) yourself to determine that *big* also modifies *guy*. Given maximal explicitness, the utility of concordances is only limited by this latter fact: Inspecting a three-million-line concordance of some word is simply not feasible, and in those cases one normally retorts to heuristic techniques to filter out the relevant patterns and/or only investigate a sample of the concordance, hoping it will reflect the overall tendency well enough.

For further study/exploration:

- on corpora and corpus linguistics in general: Leech (1993); Kennedy (1998: Chapter 2); Bowker and Pearson (2002); McEnery and Wilson (2003: Chapter 2); Baker, Hardie, and McEnery (2006); and my two favorites: Biber, Conrad, and Reppen (1998) as well as McEnery, Xiao, and Tono (2006)
- on the compilation as well as the markup and annotation of corpora: Biber (1990, 1993); Sinclair (1991: Ch. 1); de Haan (1992); Leech (1993); Kennedy (1998: Chapter 2); McEnery, Xiao, and Tono (2006: Sections A.1–A.4); Beal, Corrigan, and Moisl (2007a, b), and the references cited there