CHAPTER

1 Introduction

CHAPTER PREVIEW

KEY TERMS

computational linguistics
speech technology
signal processing
practical exercises
hardware
free software

This chapter briefly describes what the book is about, who it is aimed at, and why the subject is a useful one to study. It also describes the computational resources you will need in order to follow the practical work: basically just an ordinary PC with a CD-ROM reader and a sound card.

1.1 About this book

This is a **first, basic, elementary and short textbook in speech and natural language processing for beginners with little or no previous experience of computer programming.** The subject is, of course, somewhat technical, and there are many other textbooks that are much more technical, aimed, for example, at computer science or electrical engineering students. This book, in contrast, is aimed at beginners, including: arts, humanities and social sciences students of language, such as linguistics and psychology students; speech science students, that is, those studying communication disorders, or training for careers in speech and language therapy; as well as beginning science and engineering students seeking a quick entry-point to the main methods of the field. I have taught the course to arts students (specifically, graduate students of linguistics) several times. Since they had little technological expertise, I expect that the course may be useful for other beginners. I have expended a great deal of care and attention on trying to make it suitable for beginners, while still covering enough ground to be useful and interesting. (One caveat to that is that it is *primarily* written for language, speech and linguistics students. Consequently, I shall assume some prior knowledge of basic linguistics. See section 1.4 for more details.) I have also designed this book so that someone working alone could use it, if no formal course were available.

1.2 Purpose of this book

Languages are large and slippery entities to get to grips with, and staying on top of the data requires us to use the best tools for the job. Today, the computer is one of the most important of those tools. Many language technologies are becoming widely available on the internet, or as standard components of computer operating systems, including spelling and grammar correction, speech recognition and synthesis, and machine translation. The engineers that develop these systems value the input of linguistics graduates, especially those with some understanding of the methods commonly in use. Similarly, computational techniques have been adopted with advantage in an increasingly wide range of areas of language. A practical knowledge of these techniques can suggest powerful new analysis methods with which to study various aspects of language.

Speech processing and natural language processing are two exciting new subjects of growing relevance. They are subjects that provide new insights and tools for the analysis and generation of physical properties of sounds and the implementation of grammars in testable, working models. Unfortunately, however, existing textbooks in these subjects are mostly unsuitable for the typical student of language and speech, who normally has little experience of computer programming and no prior knowledge of digital signal processing or computational linguistics. Those textbooks assume more mathematical or programming background than is reasonable to expect of some students, especially those who are new to

the subject. Furthermore, although researchers in these two areas have begun to take note of each other's work, there continues to be a division between **signal processing** courses, which are normally taught in electronic engineering degrees, and **computational linguistics**, which is mainly directed to students of computer science or linguistics. Also, computational linguistics textbooks tend to concentrate on syntactic and semantic processing: while I shall not neglect those areas, this book also gives as much attention to phonological processing, in order to make a bridge between speech and natural language processing. This textbook provides students with a grounding in the subject that will enable them to progress to more technical books and papers if they wish. Suggestions for further reading and carefully selected background readings, and numerous exercises, are given throughout.

1.3 Some reasons to use this book

There are several reasons why students of language, in particular, may benefit from this text book:

- First, some understanding of acoustic theory is now expected of all **phonetics** students: the time when subjective phonetic transcriptions were deemed an adequate record of speech for many purposes has long since passed. For instance, the discovery of 'covert contrasts' (i.e. measurable but hardly noticeable differences between sounds) cannot be overlooked in the study of adult and child language (Dinnsen 1985; Nolan 1992; Peng 2000; Scobbie et al. 2000), studies of sound change and slips of the tongue (Mowrey and Mackay 1990).
- Second, in **experimental phonetics**, the standard tools provided in speech analysis packages are often either insufficient – how can we measure the degree of nasality or aspiration? – or impossible to use correctly without understanding the computations involved in standard techniques, such as spectral analysis or pitch tracking.
- Third, in **phonology**, an understanding of the phonetic grounding is essential. Without it, a student has no resources with which to critically consider the distinctive features, or the concepts of gradient contrast, phonetic distance and near merger. Much of the literature on acoustic features (e.g. Jakobson, Fant and Halle 1961) would be unintelligible, which partly explains the widespread avoidance of acoustics in phonological theory.
- Recent computational linguistic work on **probabilistic grammars** is having a profound impact on linguistic theory (Klavans and Resnik 1996; Bod et al. 2003), including theoretical phonology (Coleman 2000b), **morphology** (Sproat et al. 1996; Goldsmith 2001) and **syntax** (Schabes 1992; Bod 1998) – but only for those people who study it! Linguistic theory in the 1960s–1990s tended to focus on key examples, such as the ambiguity of 'the man saw the boy in the park with a telescope', the intuitive ungrammaticality of certain sentences, or

the mismatch between the phonological and morphological bracketing of 'ungrammaticality'. From the perspective of spoken language processing, though, we might ask whether there is any point expending a great deal of effort writing rules to describe such unproductive derivations as *hymn* ⇒ *hymnal*, or *electric* ⇒ *electricity* (Halle and Mohanan 1985), when the output forms are easily (and arguably more plausibly) simply stored/memorized. Faced with a word or a sentence that can be parsed in many ways (under any theory), how do we choose the right analysis, without hand waving or appealing to unexplained principles of 'pragmatics'? When writing rules in syntax or phonology, how do we deal with the fact that parsing is not deterministic?

- A great deal of the practice of linguistics concerns the grammar of a particular language, where a reasonable degree of *coverage* is required, as well as *consistency* and *correctness*. Languages are large, and the grammar-writer's goals of coverage, consistency and correctness are more easily reached by using computational tools. **Corpus-based linguistics** has moved on from being a minority interest of the 1960s and 1970s to become the common practice of computational linguistics today.
- Although the analytical techniques set out in this book were mostly developed in the first place by communications engineers and computer scientists, from whom we have much to learn, there is much that students of language can offer in return to the emerging field of spoken-language processing. Questions raised by linguists that are profoundly relevant to the subject include: What is the extent of a phonological contrast (Kelly and Local 1986; Hawkins and Slater 1994, West 1999)? Given the existence of phonological neutralization (Trubetzkoy 1969) and the non-uniqueness of phonemic analyses of a language (Chao 1934), is it reasonable – or reliable – to use phonemic transcriptions in speech technology? Does syllable structure help in speech recognition (Church 1983, 1988) or synthesis (Coleman 1992)? What about morphology? To what extent does our knowledge of word-structure and its effects depend on word grammar or analogy/similarity to other words (Bailey and Hahn 2001)? How do we assign stress to words we have not encountered before (Coleman 2000a)?

My personal motivation for writing this book rests on a refusal to let the old sociological divide between arts and sciences stand in the way of a new wave of spoken language researchers with a foot in both camps. I do not expect a reader to metamorphose into a signal processing or computational linguistics wizard – not straight away, anyway. But if, by following this course, you can understand what our speech and natural language processing colleagues in the Computer Science or Electronic Engineering departments are talking about, and if you can tinker with the programs in order to try them out, and/or adapt them to your own purposes, I will feel that the book has succeeded.

1.4 What's in the book (and what's not)

This book, the accompanying CD-ROM and the companion website contain text, lecture notes, and all the software for a short introductory course of eight two-hour classes, covering both speech and natural language processing in an integrated fashion. The CD-ROM contains all the program and data files used in the book, so you won't need to type them out by hand. Almost no prior experience of computation is assumed, beyond general computer literacy and some familiarity with a PC for word processing, use of the Windows user interface, and the practical operation of hardware, such as powering on and off, and use of disks. On the other hand, since my students also follow courses in phonetics (including acoustic phonetics), phonology, syntax, semantics and other areas of linguistics, I have assumed the reader has some training in some of these areas, and I shall not attempt to repeat material that may be found in other introductory textbooks on language.

A note for Mac and Unix users

The course software was originally developed for use in a Unix environment and has been transferred to PCs in order to reach a wider readership. In order to make this book suitable for beginners, and at the same time marketable, I have written it for PC/Windows users only. The software can be compiled and run on other machines, or under other operating systems, provided that the user has appropriate technical support on which to call.

Every effort has been taken to ensure that the course could be followed by someone who is a newcomer to spoken language processing, whether a student on a first course, a lecturer preparing such a course, or even a reader working alone without additional formal instruction. The emphasis from the outset is on practical exercises in the use and modification of computer programs written in C and Prolog, in order that students will rapidly gain confidence and acquire new skills.

Programming languages for speech and language processing

This course uses two programming languages, C and Prolog, because they have different advantages. C is particularly good for working with numbers, such as digital sound recordings, whereas Prolog is particularly good for work with the more abstract structures of language and logic. When I say 'use', I mean 'tinker about with': this is *not* a textbook in computer programming, though I do present and pick apart a number of useful, working programs, and there are a number of programming exercises. The main aim is to give students the confidence to *use, dissect* and *adapt* the programs on offer to their own purpose. By seeing how they work, the theory comes alive and

you learn to alter them to suit your own interests. As a side-effect of tinkering, you will most likely also become increasingly proficient at programming. More importantly, you may gain the confidence to delve into programs in any *other* programming languages you may come across. A good deal of work on speech is done in Pascal, Fortran, C++ and (increasingly) Java; and work on language in Lisp and Perl, to name a few other languages you may encounter in the literature. Though many people find particular languages more suited to particular areas of work, in general, almost any program can be written in *all* of these languages. This means that it is important to be fairly flexible about what programming languages you may use. As with human languages, it is best to have an open and accepting attitude, and not think that you must attain perfect fluency in one language before you will even consider using another.

The text explains from first principles the operation of many useful programs, including:

Speech processing

1. How to record sound on your computer using digital audio.
2. How to generate sounds from numbers.
3. How to calculate various properties of recorded sounds, such as overall average volume.
4. How to manipulate digital recordings, for example, how to reduce a mains hum or a tape hiss in order to clean up a low-grade tape recording.
5. How to determine the frequency spectrum of part of a signal, to find some of the acoustic characteristics of speech sounds, using Fourier analysis.
6. How to calculate the pitch of a speech signal, using the cepstral method and the autocorrelation method.
7. How to determine which parts of a speech signal are voiced and which are voiceless.
8. How to encode speech signals using linear predictive coding (LPC), and some of the uses of such an encoding.
9. How to simulate the generation of speech using a formant synthesizer and an LPC synthesizer.
10. How to compare and evaluate the similarities between two stretches of speech, using dynamic time warping, one method of automatic speech recognition.
11. How speech recognizers work.

Natural language processing

1. How to use finite-state transducers to compute relationships between descriptions on two different levels, including: (a) speech signals and

phonetic labelling (a second method of automatic speech recognition); (b) phonetic and phonemic labelling; (c) phonemic and orthographic labelling.
2. How a kind of probabilistic finite-state transducer, Hidden Markov Models, can be used to *learn* the relationships between descriptions on two different levels.
3. How to work out syllable structure, metrical structure, word structure (morphology) and sentence structure (syntax) using phrase structure grammars and parsers.
4. When a parser provides several possible structural analyses, how to determine their relative likelihood, using probabilistic grammars.

What is *not* covered

Sufficient material is provided to enable the reader to branch off into the further study of either speech processing or natural language processing, using one of the more technical textbooks in those areas, such as Harrington and Cassidy 1999 or Jurafsky and Martin 2000. In order to keep the book to nine chapters, certain topics had to be omitted. I have therefore not included anything on connectionist methods (neural networks), machine learning, semantic processing, machine translation or discourse representation. (If you want to find out about these subjects, I recommend McLeod et al. 1998, Morgan and Scofield 1991 and Jurafsky and Martin 2000.) The chapters on syntactic parsing using phrase structure grammars do not follow any specific theory of grammar, but focus instead on the computation of hierarchical constituent structures of the kind encountered in almost all mainstream varieties of grammar, as well as morphology and phonology. I have tried to be fairly non-partisan throughout, but to provide the reader with a set of techniques and an armoury of useful computational tools with which to experiment.

Tinkering

An essential feature of this course is that I encourage you to try out the programs in the form in which I have given them, to see them in operation, and then tinker with them to adapt and alter them in various ways. In my view, tinkering is an invaluable part of the process of developing a deep understanding of how computational methods work. Theory is fine, of course, but there is no substitute for practice. And this is not a computer science textbook. (For that reason, I have no qualms about simply leaving out important topics that are, quite frankly, just too hard for an introductory course.)

On a related point, don't be afraid to tinker with programs you don't really understand. If, after reading the text and looking at a program listing, it is still a mystery, don't worry! That is no reason not to try the program out. After all, most of the time we use a computer we are running programs that we know nothing about. So, a partial understanding of a program is better than none at all. And by tinkering and using the program, you may come to understand its every line very well.

1.5 Computational set-up needed for this book

All the software can be run on a standard (preferably, reasonably new) PC with a CD-ROM reader and a sound card. No expensive special-purpose hardware or software is required or assumed. To be specific, you need to have the following:

1. A PC with a 486 or, preferably, Pentium processor running the Windows 95 or more recent Windows operating system. The software has not been checked for use on an older processor or other operating systems, although with a little technical expertise or support it could be got to work on other machines or under other operating systems.
2. For the speech processing chapters, a sound card, with the ability to record and play audio files with the suffix '.wav' is needed. An amplifier and loudspeaker or headphones will be needed for audio playback, and a microphone (or player for audio recordings) will be needed for input. The miniature microphones built in to some PCs are only just good enough for speech processing, and built-in miniature loudspeakers do not do justice to the high-fidelity audio signals we shall be considering. The small loudspeakers sold for use with multimedia PCs are sometimes adequate, as are the usual kind of stereo headphones provided with portable cassette or CD players (though hi-fi headphones would be preferable). Inexpensive microphones for home audio recording, karaoke, music performance and so on are widely available from home audio and music stores. Some microphones will also require amplification to the level required for 'line input': though audio cards often accept a microphone input, they are usually specifically intended for use with 'PC mikes', not ordinary dynamic microphones. You may also need an adapter, as most microphone cables end in a ¼-inch jack plug, but most PC sound cards have a socket for a *miniature* jack plug.
3. A CD-ROM drive, in order to copy the textbook software onto your computer.
4. A reasonable amount of memory (256 Mb should be OK).
5. For users of shared-access computers, such as those provided by university computing services or departments, the appropriate privileges for installation of the course software, and a quota of file space for temporary storage. Note that the software on the CD-ROM takes up 30 Mb. In chapter 2 I explain why the digital storage of medium quality audio signals requires very large files, generally much larger than the size of text files. For instance, a file of 1 megabyte will store only 32 seconds of speech at a sampling rate of 16000 samples per second and 16-bit resolution, the lowest specification that we usually consider adequate for speech processing work. For study purposes, therefore, at least a few megabytes of disk space will be required. For serious experimental work with speech, hundreds or thousands of megabytes of disk space is typically needed. Fortunately, new computers nowadays

have disk drives with a capacity measured in gigabytes (thousands of megabytes), plenty for working with speech.
6. Although they are not essential to the course, a connection to the Internet and software such as a web browser or ftp will be useful for accessing archives of publicly available software, including the additional course software on the companion website, www.islp.org.uk. I shall assume you have Internet access.

A number of free or inexpensive software products not provided on the CD-ROM are also mentioned in various chapters. In time, these will inevitably be revised or withdrawn, so additional information and links on the book website will provide updated information.

1.6 Computational skills that are necessary in order to use the book

You need to know a little about how to use a PC:

1. How to power it up (and, if necessary, how to start Windows).
2. How to use the mouse and keyboard.
3. How to attach a microphone or tape player (for input) and loudspeaker or headphones (for output).
4. How to use Windows to manage your files.
5. If the computer is your own (or if you otherwise have the privileges to do so), how to transfer files from the CD-ROM or floppy to your own disk.
6. How to open a Command Prompt window (i.e. exit to DOS or run a DOS program).
7. It is helpful also to know how to use a web browser to inspect a remote site and use it to download publicly available software.
8. How to type plain text to a file.
9. If something doesn't work as you think it should, you'll need somewhere to turn for help. (Refer to the Acknowledgements section for a notice concerning the terms under which the software is supplied.)

Computer programs are written as plain text, that is, without the kind of formatting that word processors can provide, such as bold face, italics, underlining, different fonts or sizes of fonts. Plain text uses just the letters of the alphabet, numbers, punctuation marks and a small number of special characters, such as the space, tab, carriage return and linefeed characters. Each of these can be represented by a computer as a number: the convention for numbering characters is given in the Appendix, the ASCII table ('ASCII' stands for American Standard Code for Information Interchange). To write a file of plain text, it is best to use a simple text editor, such as Notepad or WordPad. Some commercially available C and Prolog packages also include a text editor. There are also many publicly available shareware or free software products, such as PFE (see next section). If a text editor is not available, it is also possible to

make plain text files using most word processors, by adhering to the following guidelines:

1. Type text in any font, using only ordinary keystrokes, not special combinations of keys. Use of the shift key is fine, but special characters generated using combinations of the Control (Ctl), Alt, Alt Gr or Windows key with other keys should not normally be used.
2. Different fonts, sizes or colours of fonts, bold, underline, italics, subscripts, superscripts etc. should not be used.
3. When the file is to be saved (either when typing is complete or when, after a few minutes' work it is desired to save a backup), a plain text version should be saved rather than a version in the word processor's proprietary format. For example, in Microsoft Word, do *not* save the file as a .doc file, or in Word Perfect, do *not* save the file as a .wpf file. To save a file as plain text in Microsoft Word, pull down the 'File' menu, select 'Save As', and then select 'Text Only' under the 'Save as type' menu. (Note that a file saved as 'file.c' in this way is actually named 'file.c.txt' by Windows! If you really want it to be called 'file.c', it is best to use a proper text editor, such as PFE, not a word processor.)

1.7 Free software suggestions

Although the book software is complete, it would certainly be useful to start to collect some other software tools, such as speech processing packages, text corpora and so on. At the time of writing, a number of useful software products are available either at no cost or as shareware, at a reasonable registration cost, and can be downloaded from many ftp archives. While I am happy to recommend these, I expect that the details on the companion website about their availability will inevitably change quite soon. Consequently, for more up-to-date information about how to get this software, consult the website, www.islp.org.uk.

1.8 Book structure

Roughly speaking, there are two 'strands' to the book, one on speech processing and the other on natural language processing. The speech processing program examples are written in C, and the natural language processing program examples are written in Prolog. Consequently, some chapters in each strand can be studied independently of the other strand. But in an effort to bring the two strands together into a more unified treatment of the subject, there are three places in particular at which the two strands are brought together. First, chapter 5, on finite-state machines, provides example applications in language processing (phonology, orthography and syntactic categorization) and at the speech–language (acoustics–phonology) interface. This chapter is a foundation for chapter 6 (on speech recognition techniques) and chapter 7 (on Hidden Markov Models), as well as the more language-oriented chapter 9. Chapter 9 itself brings together the speech and