Book Reviews

Introducing Speech and Language Processing

John Coleman
(University of Oxford)

Cambridge University Press (Cambridge introductions to language and linguistics), 2005, xi+301 pp; hardbound, ISBN 0-521-82365-X, $90.00; paperbound, ISBN 0-521-53069-5, $39.99

Reviewed by
Mary Harper
Purdue University

In October 2003, a group of multidisciplinary researchers convened at the Symposium on Next Generation Automatic Speech Recognition (ASR) to consider new directions in building ASR systems (Lee 2003). Although the workshop's goal of "integrating multi-disciplinary sources of knowledge, from acoustics, speech, linguistics, cognitive science, signal processing, human computer interaction, and computer science, into every stage of ASR component and system design" is an important goal, there remains a divide among these communities that can only be addressed through the educational process. The book *Introducing Speech and Language Processing* by John Coleman represents a bold effort to educate students in speech science about some of the important methods used in speech and natural language processing (NLP). This book represents an important first step for forging effective collaborations with the speech and language processing communities.

Coleman states in chapter 1 of his book that "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" (page 2). Coleman targets the book at students in a variety of disciplines, including arts, humanities, linguistics, psychology, and speech science, as well as early science and engineering students who want a glimpse into natural language and speech processing. However, since it assumes prior knowledge of basic linguistics, the text is likely to be less accessible to traditional science and engineering students. Coleman's motivation for writing this book is that the currently available textbooks in NLP and speech require knowledge that students from more of a humanities background would not have (e.g., programming, signal processing). The author also astutely points out that there tends to be a divide between the areas of signal processing and computational linguistics, although in recent years with ubiquity of statistical modeling and machine learning techniques in both areas, this divide is becoming much smaller. The author's motivation for this book is excellent: "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" (page 4).

The textbook covers a variety of techniques in speech and natural language processing, along with computer programs implementing many of them in either C or Prolog, and it capitalizes on Coleman's insights from courses offered to graduate linguistics students. It comes with a companion CD containing software needed to compile and/or execute the programs in the book, as well as source code for all of the described implementations. The *readme* file on the CD contains helpful installation notes, while the text describes how to compile and use each of the programs. Chapter 1 contains

a comprehensive list of topics that are covered from "first principles," provides details about the computational environment that is needed to compile and execute the programs provided on the CD, and a listing of computer skills one would need to get started. Coleman encourages the reader/student (I will use *student* henceforth) not just to run the programs but to also to "tinker" with them in order to gain a deeper understanding of the way they work. Chapter 1 also lays out the structure of the text graphically in order to depict the dependencies among the chapters. In addition to the book chapters, there is an appendix on ASCII characters, a helpful glossary, a list of references, and a comprehensive index. Importantly, there is also a companion website with errata, solutions to selected exercises, bug reports, software updates, additional programs, links to third-party software, and some nice bibliography links. Presumably, this page will be updated over time.

The overall chapter organization of the book is quite nice. Each chapter begins with a preview and a list of key terms (allowing the student an opportunity to look up the definitions prior to beginning to read the chapter content) and ends with a chapter summary, a set of exercises that are helpful for developing a deeper understanding of the materials discussed in the chapter, suggestions for further reading, and suggestions for readings to prepare for the next chapter. I will discuss chapters 2 through 9 in turn.

Chapter 2 discusses issues related to the digital representation of a signal with a focus on the composition of a sound file and how such a file can be loaded into a sound-editing program for audio display. The chapter starts off by guiding the student through the process of listening to a cosine waveform and then viewing the same file using a sound editing program such as Cool Edit 2000. The student is asked to fill in a worksheet with values for a cosine function and then plot the values. Coleman then presents important information on the digital representation of sound and on sampling theory. Given this knowledge, the student is walked through the process of generating and playing a cosine wave. The chapter contains a just-in-time introduction to C sufficient for a student to read and comprehend the cosine wave generation program *coswave.c*. Various computing terms (e.g., bit, compilation, machine code) are defined, followed by a discussion of C numeric data types and differences in representation across architecture. The C code presented in this chapter makes concrete Coleman's discussion of loops, arrays, calculation of mathematical expressions, overall program layout, and file output. The chapter ends with several helpful exercises. The first provides a very detailed set of instructions for compiling and executing the *coswave* program and then playing the generated output signal in Cool Edit 2000. It should be noted that Cool Edit 2000 is not a public-domain package and is no longer available through the original developers. Alternatives mentioned on the text's Web site (e.g., wavesurfer, Praat) can be used instead, although no details are offered about using them for the exercises. Students may face some challenges in opening and playing raw data files with these alternatives.

Chapter 3 introduces methods for modifying the digital representations of sound; in particular, the concept of filtering is introduced, followed by a very brief discussion of how filters are employed in a Klatt formant synthesizer. The chapter first discusses how operations can be applied to number sequences in C to set the stage for discussion of several speech-processing applications. RMS energy is then defined and a corresponding C program is discussed in detail. Next, a moving-average program is presented as an example of a low-pass filter. The concept of recursion is next introduced in order to pave the way for a discussion of IIR (Infinite Impulse Response) filters. High-, low-, and band-pass filters are defined and tables of coefficients for various filters are provided. An implementation of an IIR filter is discussed quite briefly; here the author relies on the

fact that there is similarity to the earlier moving-average program. Finally, after the basic introduction to filters, the Klatt synthesizer is discussed and a schematic diagram for the system is presented together with a brief discussion of the control parameters that are used to synthesize sound. IIR filters are tied in because they are used for modeling the glottal wave and filter-specific frequency components in order to obtain the resonant frequencies of the vocal tract required for the sound to be synthesized. A consonant–vowel example is used to demonstrate the synthesizer in action. There is a series of three exercises at the end of the chapter that should help the student get a better sense of filters and the type of sound generated by the Klatt synthesizer. The synthesizer exercises have a cookbook feel to them and give only a glimpse of what is needed to actually synthesize speech. At the end of the chapter, no further readings on filters are provided, although readings are recommended for the Klatt synthesizer and methods for estimating its parameters.

Chapter 4 discusses several programs to extract acoustic parameters from a speech signal. First up is the fast Fourier transform (FFT), for which a C implementation is presented and described in detail. The student is asked to apply the compiled code to an example speech file in order to generate its spectrum, which is then plotted in Excel or Matlab for comparison to the spectral analysis obtained using Cool Edit. Given this example, there is a discussion of the types of peaks found in the spectrum, the resonances of the vocal tract, and the harmonics, as a prelude to the discussion of cepstral analysis. Coleman first provides a high-level discussion of cepstral analysis, which employs an inverse FFT, followed by the discussion of its C implementation and an example using the executable. Cepstral analysis is then used to build a rudimentary pitch tracker, which is applied to a speech example. This leads to the discussion of a voicing detection algorithm. Next, the autocorrelation method for pitch tracking is presented together with its C implementation. Finally, the chapter discusses linear predictive coding (LPC) and various applications. The chapter ends with exercises to compare the cepstral and autocorrelation pitch trackers, to modify the output of the LPC program, and to analyze the vowels in a speech sample and use the LPC spectrum to estimate their formants. Additional readings are provided on the algorithms presented in this chapter.

Chapter 5 offers a change of pace as the book introduces finite-state machines (FSMs) with a focus initially on symbolic language applications. There is a shift from C to Prolog, although it would have been perhaps more coherent to stick with C. The discussion of the peculiarities of Prolog could be distracting to a novice programmer. Furthermore, the representation of an FSM in Prolog is tedious to read, and it may be difficult for the uninitiated to observe correspondences between the Prolog code and depictions of corresponding models. Simple examples are used to introduce the concept of, rather than a formal definition of, FSMs. Issues of coverage, over-generation, determinism, and nondeterminism of an FSM are discussed briefly. Although Coleman makes clear that backtracking is an issue for a nondeterministic FSM and notes that there are methods for converting such an FSM to a representationally equivalent deterministic form, existing tools that could be used for carrying out this conversion (e.g., the AT&T FSM library) are not discussed. Coleman next presents a Prolog implementation of an interesting English-like monosyllable FSM. A box is used to introduce a collection of facts about Prolog, and then there is a walk-through of the code. A nice set of exercises follows in which the student loads the FSM program and executes it in various ways, followed by a discussion of some examples of using the FSM to generate strings with particular constraints. A more formal presentation of FSMs is then provided together with a discussion of the state-transition-table representation. The chapter ends by

introducing the concept of finite-state transducers and providing several examples from various levels of processing, including phonetics, phonology, orthography, and syntax. Exercises at the end of the chapter build nicely upon the Prolog code already discussed. The suggested additional readings are appropriate, but perhaps too broad, as many are textbooks.

Chapter 6 turns to the topic of automatic speech recognition. Coleman provides a general discussion of knowledge-based and pattern-recognition approaches to ASR without a historical perspective. The knowledge-based method with its focus on feature extraction and knowledge integration is described at a very high level without the benefit of any hands-on exercises. Coleman uses dynamic time warping (DTW) to exemplify the pattern-matching approach, as it is a fairly straightforward dynamic programming algorithm of which the student can gain some understanding by filling in tables of distances. The chapter also contains a nice discussion on the sources of variability in speech, although no insights are offered on how they would be addressed by the two approaches to ASR. Only two exercises are found in this chapter, one to fill in matrices used by the dynamic time-warping algorithm and one asking the student to think about pitfalls of the pattern-matching approach. The chapter does not discuss the implementation of any of the methods discussed, although I believe a C implementation of DTW could have been added to good effect. There are some helpful recommended readings on ASR techniques, many of which are textbooks or edited books of papers.

Chapter 7 introduces probabilistic finite-state approaches, bringing together acoustic analysis with finite-state models. The chapter begins with a discussion of Markov models and the use of probabilistic methods for coping with uncertainty. Part-of-speech *n*-gram models are introduced together with a very brief discussion of probabilities and Markov models (along with a few simple exercises). Coleman then provides an informal discussion of the hidden Markov model (HMM), followed by a discussion of trigram models, data incompleteness, and backoff. Finally, the three basic problems for HMMs (Rabiner 1989; Rabiner and Juang 1993) are discussed, providing the student with a clearer understanding of the kinds of questions that can be addressed with them. There are two very short sections on using HMMs for part-of-speech tagging and speech recognition, but there are no code or exercises associated with them. The chapter ends with a discussion of Chomsky's objections to Markov models and a response to each. The only exercises appearing in this chapter concern probability and Markov models. The chapter does not discuss implementations of any of the approaches discussed, and yet it would seem that the student would gain a deeper understanding of many of the topics presented in this chapter by playing, for example, with a simple part-of-speech tagger. There are many publicly available resources that could be used to fill in this hole.

Chapter 8 moves on to parsing, building upon the knowledge of Prolog gained in chapter 5. A simple definite-clause grammar is introduced, followed by an intuitive discussion of parsing and recursive descent parsers. A second grammar, *dog_grammar.pl*, is then discussed together with difference lists in Prolog so that the grammar can be updated to produce a tree structure. Coleman then provides an example grammar that breaks phoneme sequences into syllables. The chapter ends with a very brief introduction to various parsing algorithms, chart parsing, issues of search, deterministic parsing, and parallel parsing. The chapter would have been improved by the addition of exercises; however, the student could load the grammars discussed in the chapter into Prolog and play with them. Several textbooks are recommended for additional reading, although the novice might gain a richer perspective by consulting the chapters on parsing of Allen (1994).

Chapter 9, the final chapter in the book, discusses the incorporation of probability into a context-free parsing algorithm. Coleman begins with a discussion about why a probabilistic approach is useful in computational linguistics, ranging from the fact that human judgments of grammaticality are gradient and at times uncertain of providing a good mechanism to account for collocations and the learnability of grammars. A simple probabilistic context-free grammar (CFG) is then presented, along with a discussion of how to obtain the grammar rules and estimate their probabilities. The chapter ends by discussing limitations of probabilistic CFGs and briefly introducing two alternative approaches, tree-adjoining grammars and data-oriented parsing. This chapter contains no exercises for the student. However, it does provide a list of materials to assist the student in learning more about C programming, digital signal processing, the Klatt synthesizer, speech recognition, Prolog, computational linguistics, and probabilistic grammars.

Overall, Coleman has written a textbook that more than adequately fulfills his goal of introducing the uninitiated to a variety of techniques in speech and language processing. Due to its broad coverage, the text is unable to delve deeply into many of the details, although this is mitigated in part by the fact that he provides additional readings for students with an interest in a particular topic. The reading list on the companion website would be improved by including more modern sources, pointers to current conferences and journals in speech and natural language processing (e.g., Bird 2005), and links to helpful resources available on the Internet (e.g., DISC 1999; Hunt 1997; Jamieson 2002; Kita 2000; Krauwer 2005; Manning 2005; Picone 2005). Additionally, although the book is not aimed at students with a strong background in mathematics or computer science, they would benefit from additional readings in these areas. The book would benefit from additional editing, as it contains errors that could easily confuse a novice, as well as from the addition of more hands-on exercises, particularly in Chapters 6 through 9. Quibbles aside, if the book builds bridges between the communities Coleman desires, it will have a broad impact that could be felt for years to come. I believe education is an important first step to building multidisciplinary solutions to some of the most pressing problems in speech and natural language processing. It would be wonderful to see more books with Coleman's vision.

References

Allen, James. 1994. *Natural Language Understanding*. The Benjamin/Cummings Publishing Company Inc., Redwood City, CA.

Bird, Steven, editor. 2005. ACL Anthology: A digital archive of research papers in computational linguistics. *acl.ldc.upenn.edu/*.

DISC. 1999. A survey of existing methods and tools for developing and evaluation of speech synthesis and of commercial speech synthesis systems. *www.disc2.dk/tools/SGsurvey.html*.

Hunt, Andrew. 1997. CompṠpeech frequently asked questions. *fife.speech.cs.cmu.edu/comp.speech/*.

Jamieson, Leah H. 2002. Notes for EE649 lectures: Speech processing by computer. *shay.ecn.purdue.edu/ee649/notes/*.

Kita, Kenji. 2000. Speech and language Web resources. *www-a2k.is.tokushima-u.ac.jp/member/kita/NLP/*.

Krauwer, Steven. 2005. Tools for NLP and speech. *www.elsnet.org/toolslist.html*.

Lee, Chin-Hui. 2003. NSF Symposium on Next Generation ASR. *users.ece.gatech.edu/chl/ngasr03/*.

Manning, Christopher. 2005. Linguistics, natural language, and computational linguistics meta-index. *www-nlp.stanford.edu/links/linguistics.html*.

Picone, Joseph. 2005. Automatic speech recognition. *www.cavs.msstate.edu/hse/ies/projects/speech/*.

Rabiner, Lawrence R. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Rabiner, Lawrence and Biing-Hwang Juang. 1993. *Fundamentals of Speech Recognition*. Prentice-Hall, Inc., Upper Saddle River, NJ.

Mary Harper joined the faculty of Purdue University in 1989, where she holds the rank of Professor in the School of Electrical and Computer Engineering. She recently finished a term of slightly more than three years as the Program Director for the Human Language and Communication Program at the National Science Foundation with the goal of advancing research in speech, natural language, and multimodal processing. Her research focuses on computer modeling of human communication with a focus on methods for incorporating multiple types of knowledge sources, including lexical, syntactic, prosodic, and, most recently, visual sources. Harper is currently at the Center for Advanced Study of Language, University of Maryland, College Park, MD 20742-0025; e-mail: harper@purdue.edu, mharper@casl.umd.edu; URL: yara.ecn.purdue.edu/∼harper.