“This book explains the fundamentals of language technology in a coherent and engaging way without bursting at the seams of daunting calculations and formulas that typically turn off even the most enthusiastic readers.”

Anna Feldman, Montclair State University

“A clear and timely introduction to the computational processing of language, an important and fast-growing area of study and application. This text provides clear, precise and concise explanations of many of the core problems, solutions and applications encountered in the area.”

Jason Baldridge, The University of Texas at Austin

The widening use of computers has powerfully influenced the way people communicate, search for, and store information. For the majority of individuals and situations, the primary vehicle for such information is natural language, where text and speech are crucial encoding formats for the information revolution.

This book introduces students to the fundamentals of how computers are used to represent, process, and organize textual and spoken information. It allows students to gain an effective understanding of how the computer works and where problems arise with the involvement of natural language. Self-contained chapters cover the central analytical concepts and provide students with tips on how to integrate this knowledge into their working practices.

The authors ground the concepts and analyses covered in the text in real-world examples familiar to students. Drawing on these examples, the authors teach students how to produce evidence-based analyses and arguments about language. The result is a book that teaches students to generate, justify, and argue for valid conclusions about the design, capabilities, and behavior of natural language systems.

Markus Dickinson is Assistant Professor at the Department of Linguistics, Indiana University and currently the director of the Computational Linguistics program. His research focuses on improving linguistic annotation for natural language processing technology and automatically analyzing the language of second language learners.

Chris Brew is a Senior Research Scientist with the Educational Testing Service in Princeton, where he is currently the scientific lead for the c-rater project on automated short answer grading. He has been active in Natural Language Processing for over 20 years, first in the UK, then as Associate Professor of Linguistics and Computer Science at The Ohio State University, where he co-directed the Speech and Language Technologies Laboratory, as well as the Computational Linguistics program.

Detmar Meurers is Professor of Computational Linguistics at the University of Tübingen. Before moving to Germany, he was an Associate Professor of Linguistics at The Ohio State University, where as co-director of the Computational Linguistics program he developed a teaching approach combining current technology and research issues with the fundamentals of the field. His research emphasizes the role of linguistic insight and modeling in Computational Linguistics, with a more recent focus on analysis and applications related to Second Language Acquisition.
Language and Computers
Language and Computers

Markus Dickinson, Chris Brew and Detmar Meurers
Brief Contents

What This Book Is About xi
Overview for Instructors xiii
Acknowledgments xvii

1 Prologue: Encoding Language on Computers 1
2 Writers’ Aids 33
3 Language Tutoring Systems 69
4 Searching 91
5 Classifying Documents: From Junk Mail Detection to Sentiment Classification 127
6 Dialog Systems 153
7 Machine Translation Systems 181
8 Epilogue: Impact of Language Technology 215

References 221
Concept Index 227
Contents

What This Book Is About xi
Overview for Instructors xiii
Acknowledgments xvii

1 Prologue: Encoding Language on Computers 1
   1.1 Where do we start? 1
       1.1.1 Encoding language 2
   1.2 Writing systems used for human languages 2
       1.2.1 Alphabetic systems 3
       1.2.2 Syllabic systems 6
       1.2.3 Logographic writing systems 8
       1.2.4 Systems with unusual realization 11
       1.2.5 Relation to language 11
   1.3 Encoding written language 12
       1.3.1 Storing information on a computer 12
       1.3.2 Using bytes to store characters 14
   1.4 Encoding spoken language 17
       1.4.1 The nature of speech 17
       1.4.2 Articulatory properties 18
       1.4.3 Acoustic properties 18
       1.4.4 Measuring speech 20
       Under the Hood 1: Reading a spectrogram 21
       1.4.5 Relating written and spoken language 24
           Under the Hood 2: Language modeling for automatic speech recognition 26
# Contents

## 2 Writers’ Aids

- **2.1 Introduction**
- **2.2 Kinds of spelling errors**
  - 2.2.1 Nonword errors
  - 2.2.2 Real-word errors
- **2.3 Spell checkers**
  - 2.3.1 Nonword error detection
  - 2.3.2 Isolated-word spelling correction
- **Under the Hood 3: Dynamic programming**

## 2.4 Word correction in context

- **2.4.1 What is grammar?**
- **Under the Hood 4: Complexity of languages**
- **2.4.2 Techniques for correcting words in context**
- **Under the Hood 5: Spell checking for web queries**

## 2.5 Style checkers

## 3 Language Tutoring Systems

- **3.1 Learning a language**
- **3.2 Computer-assisted language learning**
- **3.3 Why make CALL tools aware of language?**
- **3.4 What is involved in adding linguistic analysis?**
  - 3.4.1 Tokenization
  - 3.4.2 Part-of-speech tagging
  - 3.4.3 Beyond words
- **3.5 An example ICALL system: TAGARELA**
- **3.6 Modeling the learner**

## 4 Searching

- **4.1 Introduction**
- **4.2 Searching through structured data**
- **4.3 Searching through unstructured data**
  - 4.3.1 Information need
  - 4.3.2 Evaluating search results
  - 4.3.3 Example: Searching the web
  - 4.3.4 How search engines work
  - **Under the Hood 6: A brief tour of HTML**
- **4.4 Searching semi-structured data with regular expressions**
  - 4.4.1 Syntax of regular expressions
  - 4.4.2 Grep: An example of using regular expressions
  - **Under the Hood 7: Finite-state automata**
- **4.5 Searching text corpora**
  - 4.5.1 Why corpora?
  - 4.5.2 Annotated language corpora
  - **Under the Hood 8: Searching for linguistic patterns on the web**
Contents

5 Classifying Documents: From Junk Mail Detection to Sentiment Classification 127
  5.1 Automatic document classification 127
  5.2 How computers “learn” 129
    5.2.1 Supervised learning 130
    5.2.2 Unsupervised learning 131
  5.3 Features and evidence 131
  5.4 Application: Spam filtering 133
    5.4.1 Base rates 135
    5.4.2 Payoffs 139
    5.4.3 Back to documents 139
  5.5 Some types of document classifiers 140
    5.5.1 The Naive Bayes classifier 140
      Under the Hood 9: Naive Bayes 142
    5.5.2 The perceptron 145
    5.5.3 Which classifier to use 148
  5.6 From classification algorithms to context of use 149

6 Dialog Systems 153
  6.1 Computers that “converse”? 153
  6.2 Why dialogs happen 155
  6.3 Automating dialog 156
    6.3.1 Getting started 156
    6.3.2 Establishing a goal 157
    6.3.3 Accepting the user’s goal 157
    6.3.4 The caller plays her role 158
    6.3.5 Giving the answer 158
    6.3.6 Negotiating the end of the conversation 159
  6.4 Conventions and framing expectations 159
    6.4.1 Some framing expectations for games and sports 160
    6.4.2 The framing expectations for dialogs 160
  6.5 Properties of dialog 161
    6.5.1 Dialog moves 161
    6.5.2 Speech acts 162
    6.5.3 Conversational maxims 164
  6.6 Dialog systems and their tasks 166
  6.7 Eliza 167
    Under the Hood 10: How Eliza works 172
  6.8 Spoken dialogs 174
  6.9 How to evaluate a dialog system 175
  6.10 Why is dialog important? 176

7 Machine Translation Systems 181
  7.1 Computers that “translate”? 181
  7.2 Applications of translation 183
## Contents

7.2.1 Translation needs 183
7.2.2 What is machine translation really for? 184

7.3 Translating Shakespeare 185

7.4 The translation triangle 188

7.5 Translation and meaning 191

7.6 Words and meanings 193
    7.6.1 Words and other languages 193
    7.6.2 Synonyms and translation equivalents 194

7.7 Word alignment 194

7.8 IBM Model 1 198
    Under the Hood 11: The noisy channel model 200
    Under the Hood 12: Phrase-based statistical translation 204

7.9 Commercial automatic translation 205
    7.9.1 Translating weather reports 205
    7.9.2 Translation in the European Union 207
    7.9.3 Prospects for translators 208

8 Epilogue: Impact of Language Technology 215

References 221

Concept Index 227
What This Book Is About

The computer has become the medium of choice through which much of our language use is channeled. Modern computer systems therefore spend a good part of their time working on human language. This is a positive development: not only does it give everyone on the internet access to a world of information well beyond the scope of even the best research libraries of the 1960s and 1970s, it also creates new capabilities for creation, exploitation, and management of information. These include tools that support nonfiction, creative writing, blogs and diaries, citizen journalism and social interactions, web search and online booking systems, smart library catalogs, knowledge discovery, spoken language dialogs, and foreign language learning.

This book takes you on a tour of different real-world tasks and applications where computers deal with language. During this tour, you will encounter essential concepts relating to language, representation, and processing, so that by the end of the book you will have a good grasp of key concepts in the field of computational linguistics. The only background you need to read this book is some curiosity about language and some everyday experience with computers.

This is indeed why the book is organized around real-world tasks and applications. We assume that most of you will be familiar with many of the applications and may wonder how they work or why they don’t work. What you may not realize is how similar the underlying processing is. For example, there is a great deal in common between how grammar checkers and automatic speech-recognition systems work. We hope that demonstrating how these concepts recur – in this case, in something called $n$-grams – will reinforce the importance of applying general techniques to new applications.
What This Book Is About

The book is designed to make you aware of how technology works and how language works. We focus on a few applications of language technology (LT), computational linguistics (CL), and natural language processing (NLP). LT, CL, and NLP are essentially names for the same thing, seen from the perspectives of industry, linguistics, and computer science, respectively. The tasks and applications were chosen because: (i) they are representative of techniques used throughout the field; (ii) they represent a significant body of work in and of themselves; (iii) they connect directly to linguistic modeling; and (iv) they are the ones the authors know best. We hope that you will be able to use these examples as an introduction to general concepts that you can apply to learning about other applications and areas of inquiry.

How to use the book

There are a number of features in this textbook that allow you to structure what you learn, explore more about the topics, and reinforce what you are learning. As a start, the relevant concepts being covered are typeset in bold and shown in the margins of each page. You can also look those up in the Concept Index at the end of the book.

The Under the Hood sections included in many of the chapters are intended to give you more detail on selected advanced topics. For those interested in learning more about language and computers, we hope that you find these sections enjoyable and enlightening, though the gist of each chapter can be understood without reading them.

At the end of each chapter there is a Checklist indicating what you should have learned. The Exercises also found at the end of each chapter review the material and give you opportunities to go beyond it. Our hope is that the checklist and exercises help you to get a good grasp of each of the topics and concepts involved. We recognize, however, that students from different backgrounds have different skills, so we have marked each question with an indication of who the question is for. There are four designations: most questions are appropriate for all students and thus are marked with ALL; LING questions assume some background and interest in linguistics; CS questions are appropriate for those with a background in computer science; and MATH is appropriate for those wanting to tackle more mathematical challenges. Of course, you should not feel limited by these markers, as a strong enough desire will generally allow you to tackle most questions.

If you enjoy the topic of a particular chapter, we encourage you to make use of the Further reading recommendations. You can also follow the page numbers under each entry in the References at the end of the book to the place where it is discussed in the book.

Finally, on the book’s companion website http://purl.org/lang-and-comp we have collected resources and links to other materials that could be of interest to you when exploring topics around language and computers.
Everyday natural language processing tools, such as those mentioned in the previous section, provide new educational opportunities. The goal of our courses is to show students the capabilities of these tools, and especially to encourage them to take a reflective and analytic approach to their use.

The aim of this book is to provide insight into how computers support language-related tasks, and to provide a framework for thinking about these tasks. There are two major running themes. The first is an emphasis on the relation between the aspects of language that need to be made explicit to obtain working language technology and how they can be represented. We introduce and motivate the need for explicit representations, which arises from the fact that computers cannot directly work with language but require a commitment to linguistic modeling and data structures that can be expressed in bits and bytes. We emphasize the representational choices that are involved in modeling linguistic abstractions in a concrete form computers can use.

The second running theme is the means needed in order to obtain the knowledge about language that the computer requires. There are two main options here: either we arrange for the computer to learn from examples, or we arrange for experts to create rules that encode what we know about language. This is an important distinction, both for students whose primary training is in formal linguistics, to whom the idea that such knowledge can be learned may be unfamiliar, and for the increasing number of students who are exposed to the "new AI" tradition of machine learning, to whom the idea of creating and using hand-coded knowledge representations may be surprising. Our view is that the typical real-world system uses a synthesis of both approaches, so we want students to understand and respect the strengths and weaknesses of both data-driven and theory-driven traditions.
Chapter-by-chapter overview

Chapter 1. Prologue: Encoding Language  This chapter lays the groundwork for understanding how natural language is used on a computer, by outlining how language can be represented. There are two halves to this chapter, focusing on the two ways in which language is transmitted: text and speech. The text portion outlines the range of writing systems in use and then turns to how information is encoded on the computer, specifically how all writing systems can be encoded effectively. The speech portion offers an overview of both the articulatory and the acoustic properties of speech. This provides a platform for talking about automatic speech recognition and text-to-speech synthesis. The chapter closes with a discussion of language modeling in the context of speech recognition.

Chapter 2. Writers’ Aids  This chapter sets out to (i) explain what is currently known about the causes of and reasons for spelling errors; (ii) introduce the main techniques for the separate but related tasks of nonword error detection, isolated-word spelling correction, and real-word spelling correction; and (iii) introduce the linguistic representations that are currently used to perform grammar correction – including a lengthy discussion of syntax – and explain the techniques employed. The chapter describes classical computational techniques, such as dynamic programming for calculating edit distance between words. It concludes with a discussion of advances in technology for applying spelling correction to newer contexts, such as web queries.

Chapter 3. Language Tutoring Systems  In this chapter, we seek to (i) introduce some fundamentals of first and second language acquisition and the relevance of language awareness for the latter; (ii) explain how computer-assisted language learning (CALL) tools can offer feedback for exercises without encoding anything about language in general; (iii) motivate and exemplify that the space of well-formed and ill-formed variation arising in language use often is well beyond what can be captured in a CALL tool; (iv) introduce the idea that the need for linguistic abstraction and generalization is addressed in tokenization and part-of-speech tagging as two fundamental NLP processing steps, that even such basic steps can be surprisingly difficult, and how part-of-speech classes are informed by different types of empirical evidence; (v) motivate the need for analysis beyond the word level and syntactic generalizations; and (vi) showcase what a real-life intelligent language tutoring system looks like and how it makes use of linguistic analysis for both analysis and feedback. The chapter ends with a section discussing how in addition to the context of language use and linguistic analysis, learner modeling can play an important role in tutoring systems.

Chapter 4. Searching  To cover the task of searching, the goals of this chapter are (i) to outline the contexts and types of data in which people search for information (structured, unstructured, and semi-structured data), emphasizing the concept of
Overview for Instructors

one's information need; (ii) to provide ways to think about the evaluation and improvement of one's search results; (iii) to cover the important concept of regular expressions and the corresponding machinery of finite-state automata; and (iv) to delve into linguistic corpora, illustrating a search for linguistic forms instead of for content. The middle of the chapter provides more in-depth discussion of web searching, including how webpages are indexed and how the PageRank algorithm is used to determine relevance for a query.

Chapter 5. Classifying Documents  This chapter aims to (i) explain the idea of classifiers and machine learning; (ii) introduce the Naïve Bayes and Perceptron classifiers; (iii) give basic information about how to evaluate the success of a machine learning system; and (iv) explain the applications of machine learning to junk-mail filtering and to sentiment analysis. The chapter concludes with advice on how to select a machine learning algorithm and a discussion of how this plays out for a consulting company employing sentiment analysis as part of an opinion-tracking application designed to be used by corporate customers.

Chapter 6. Dialog Systems  The goals of this chapter are (i) to introduce the idea of dialog systems; (ii) to describe some of the ways in which researchers have conceptualized dialog, including dialog moves, speech acts, and conversational maxims; (iii) to show some of the ways of classifying dialog systems according to their purpose and design; and (iv) to illustrate how to measure the performance of dialog systems. We spend some time discussing the difficulties in automating dialogue and illustrate this with the example of the early dialog system Eliza.

Chapter 7. Machine Translation Systems  Starting from the general idea of what it means to translate, in this chapter we (i) introduce the idea of machine translation (MT) and explain its capabilities and limits; (ii) indicate the differences between direct MT systems, transfer systems, and modern statistical methods; and (iii) set machine translation in its business context, emphasizing the idea of a translation need. The chapter includes extended discussion of IBM's Model 1 translation model and of the Noisy Channel Model as applied to translation. It also discusses the translation needs of the European Union and those of the Canadian Meteorological Service, and contrasts them with the very difficult requirements for a satisfactory translation of a Shakespeare sonnet. The chapter concludes with a discussion of the likely consequences of automated translation for the career prospects and training choices of human translators.

Chapter 8. Epilogue: Impact of Language Technology  The final chapter takes a look at the impact of language technology on society and human self-perception, as well as some of the ethical issues involved. We raise questions about how computers and language technology change the way information can be accessed and what this means for a democratic society in terms of control of information and privacy, how this changes learning and teaching as well as our jobs through upskilling and
Overview for Instructors

deskilling, and the impact on human self-perception when faced with machines capable of communicating using language. The goal of the chapter is to raise awareness of such issues arising through the use of computers and language technology in the context of real life.

How to use the book

A typical way to use the material in this book is in a quarter-length course assuming no mathematical or linguistic background beyond normal high-school experience. For this kind of course, instructors may choose to cover only a subset of the chapters. Each chapter is a stand-alone package, with no strict dependencies between the topics. We have found this material to be accessible to the general student population at the Ohio State University (OSU), where we originally developed the course.

To support the use of the book for longer or more advanced courses, the book also includes Under the Hood sections providing more detail on selected advanced topics, along with development of related analytical and practical skills. This kind of use is more appropriate as part of a Linguistics, Computer Science, or Communications major, or as an overview course at the nonspecialist graduate level. The Under the Hood topics have been useful in semester-length courses at Georgetown University and Indiana University, as well as honors versions at OSU.

Accompanying the book there is a website containing course materials, such as presentation slides, at http://purl.org/lang-and-comp/teaching.
This book grew out of a course that was offered for the first time in the winter quarter of 2004 at the Ohio State University (OSU): Linguistics 384, Language and Computers. A special thanks to the chair of the Department of Linguistics at the time, Peter Culicover, as well as the Director of Undergraduate Studies in Linguistics, Beth Hume, for having the foresight to recognize the potential for such a course and supporting its development and approval as a general education requirement course at OSU.

There would not be a book were it not for Danielle Descoteaux, who heard about the course and our ideas to turn it into a book and encouraged us to realize this with Wiley-Blackwell. Her sustained enthusiasm for the project made sure we stayed on the ball. We are also very grateful to our project editor Julia Kirk, who continued to be supportive and friendly despite our slow progress.

Drafts of this book benefited significantly from feedback on the course at OSU, as well as similar courses taught at Georgetown University (GU) and Indiana University (IU). The instructors shared many good ideas and pointers to relevant materials, for which we would like to thank particularly Stacey Bailey, Xiaofei Lu (who also provided the Chinese characters for this book), Anna Feldman, DJ Hovermale, Jon Dehdari, Rajakrishnan Rajkumar, Michael White, Sandra Kübler, Ross Israel, and the computational linguistics community at our universities for encouragement and neat ideas for the course and book. Thanks also to Lwin Moe for providing figures of the Burmese writing system; to Tony Meyer and Ayelet Weiss for help on the Hebrew examples; and to Wes Collins for providing Mam examples in chapter 7.

We wish to specially acknowledge Jason Baldridge at the University of Texas, who has continually tested book chapters, provided insightful suggestions for the book and associated courses, and diligently encouraged us to get the book completed.

While it would take too long to name them individually, the students at OSU, GU, and IU who took these courses have been a joy to teach, and their feedback on particular exercises and requests for clarifications on material have definitely made the book better.
A number of people read drafts or partial drafts and provided useful comments. Thanks to Amber Smith for her comments and discussion on integrating the book material into a real course, to Johannes Widmann for his comments on the Language Tutoring Systems chapter, and Keelan Evanini and Michael Heilman at Educational Testing Service (ETS) for extensive and extremely helpful comments on every aspect of the book, from the structure of the chapters through the best way to talk about neural networks to correcting typographical mistakes (including the especially awkward ones in the sentences advocating the use of spell checkers). Sheena Phillips, Jason Quinley, and Christl Glauder also helped us improve the book by carefully proofreading the final version – thanks!

Speaking individually, Markus Dickinson would like to thank Stephanie Dickinson for her encouragement and support during the last few years of this project, and also for her willingness to discuss specificity, sensitivity, and other classification metrics at the dinner table. Lynn Weddle deserves credit for responding to a Facebook post and suggesting Bart and Lisa Simpson for an example in the Searching chapter – although she had no idea it had to do with a book or would lead to an acknowledgment.

Chris Brew thanks Sheena Phillips for everything, and specifically for being exactly the right person to answer the question: “Is this too British?” Matthew and Jamie Brew gave helpful advice on the design of the cover, pointing out things that the older generation just did not see.

Detmar Meurers would like to thank Markus and Chris for the excellent collaboration and being such reliable colleagues and friends, Walter Meurers for emphasizing the importance of connecting research issues with the real world, and Kordula De Kuthy, Marius, Dario, and Cora for being around to remind him that life has a meaning beyond deadlines.
1

Prologue

Encoding Language on Computers

1.1 Where do we start?

One of the aims of this book is to introduce you to different ways that computers are able to process natural language. To appreciate this task, consider how difficult it is to describe what happens when we use language. As but one example, think about what happens when a friend points at a book and says: “He’s no Shakespeare!” First of all, there is the difficulty of determining who is meant by “he”. Your friend is pointing at a book, not at a person, and although we can figure out that “he” most likely refers to the author of the book, this is not obvious to a computer (and sometimes not obvious to the person you are talking with). Secondly, there is the difficulty of knowing who “Shakespeare” is. Shakespeare is the most famous writer in the English language, but how does a computer know that? Or, what if your friend had said “He’s no Lessing!”? English majors with an interest in science-fiction or progressive politics might take this as a reference to Doris Lessing; students of German literature might suspect a comparison to G.E. Lessing, the elegant Enlightenment stylist of German theater; but in the absence of background knowledge, it is hard to know what to make of this remark.

Finally, even if we unpack everything else, consider what your friend’s statement literally means: the author of this book is not William Shakespeare. Unless there is a serious possibility that the book was written by Shakespeare, this literal meaning is such a crushingly obvious truth that it is difficult to see why anyone would bother to express it. In context, however, we can tell that your friend is not intending to express the literal meaning, but rather to provide a negative evaluation of the author relative to Shakespeare, who is the standard benchmark for good writing in English. You could do the same thing for a slim book of mystical poetry by saying “She’s no Dickinson!”, provided the hearer was going to understand that the reference was to American poet Emily Dickinson.
Or consider a different kind of statement: “I’m going to the bank with a fishing pole.” Most likely, this means that the speaker is going to a river bank and is carrying a fishing pole. But it could also mean that the speaker is going to a financial institution, carrying a fishing pole, or it could mean that the speaker is going to a financial institution known for its fishing pole – or even that the river bank the speaker is going to has some sort of notable fishing pole on it. We reason out a preferred meaning based on what we know about the world, but a computer does not know much about the world. How, then, can it process natural language?

From the other side of things, let us think for a moment about what you may have observed a computer doing with natural language. When you get a spam message, your email client often is intelligent enough to mark it as spam. Search for a page in a foreign language on the internet, and you can get an automatic translation, which usually does a decent job informing you as to what the site is about. Your grammar checker, although not unproblematic, is correct a surprising amount of the time. Look at a book’s listing on a site that sells books, like Amazon, and you may find automatically generated lists of keywords; amazingly, many of these words and phrases seem to give a good indication of what the book is about.

If language is so difficult, how is it that a computer can “understand” what spam is, or how could it possibly translate between two languages, for example from Chinese to English? A computer does not have understanding, at least in the sense that humans do, so we have to wonder what technology underlies these applications. It is these very issues that we delve into in this book.

1.1.1 Encoding language

There is a fundamental issue that must be addressed here before we can move on to talking about various applications. When a computer looks at language, what is it looking at? Is it simply a variety of strokes on a piece of paper, or something else? If we want to do anything with language, we need a way to represent it.

This chapter outlines the ways in which language is represented on a computer; that is, how language is encoded. It thus provides a starting point for understanding the material in the rest of the chapters.

If we think about language, there are two main ways in which we communicate – and this is true of our interactions with a computer, too. We can interact with the computer by writing or reading text or by speaking or listening to speech. In this chapter, we focus on the representations for text and speech, while throughout the rest of the book we focus mainly on processing text.

1.2 Writing systems used for human languages

If we only wanted to represent the 26 letters of the English alphabet, our task would be fairly straightforward. But we want to be able to represent any language in any writing system, where a writing system is “a system of more or less permanent
marks used to represent an utterance in such a way that it can be recovered more or less exactly without the intervention of the utterer” (Daniels and Bright, 1996).

And those permanent marks can vary quite a bit in what they represent. We will look at a basic classification of writing systems into three types: alphabetic, syllabic, and logographic systems. There are other ways to categorize the world’s writing systems, but this classification is useful in that it will allow us to look at how writing systems represent different types of properties of a language by means of a set of characters. Seeing these differences should illustrate how distinct a language is from its written representation and how the written representation is then distinct from the computer’s internal representation (see Section 1.3).

For writing English, the idea is that each letter should correspond to an individual sound, more or less, but this need not be so (and it is not entirely true in English). Each character could correspond to a series of sounds (e.g., a single character for *str*), but we could also go in a different direction and have characters refer to meanings. Thus, we could have a character that stands for the meaning of “dog”. Types of writing systems vary in how many sounds a character represents or to what extent a meaning is captured by a character. Furthermore, writing systems differ in whether they even indicate what a word is, as English mostly does by including spaces; we will return to this issue of distinguishing words in Section 3.4.

One important point to remember is that these are systems for writing down a language; they are not the language itself. The same writing system can be used for different languages, and the same language in principle could be written down in different writing systems (as is the case with Japanese, for example).

### 1.2.1 Alphabetic systems

We start our tour of writing systems with what should be familiar to any reader of English: alphabets. In **alphabetic systems**, a single character refers to a single sound. As any English reader knows, this is not entirely true, but it gives a good working definition.

We will look at two types of alphabetic systems. First, there are the **alphabets**, or phonemic alphabets, which represent all sounds with their characters; that is, both consonants and vowels are represented. Many common writing systems are alphabets: Etruscan, Latin, Cyrillic, Runic, and so forth. Note that English is standardly written in the Latin, or Roman, alphabet, although we do not use the entire repertoire of available characters, such as those with accents (e.g., è) or **ligatures**, combinations of two or more characters, such as the German ß, which was formed from two previous versions of s.

As an example of an alphabet other than Latin, we can look at Cyrillic, shown in Figure 1.1. This version of the alphabet is used to write Russian, and slight variants are used for other languages (e.g., Serbo-Croatian). Although some characters correspond well to English letters, others do not (e.g., the letter for [n]). The characters within brackets specify how each letter is said – that is, pronounced; we will return to these in the discussion of phonetic alphabets later on.
Prologue

Some alphabets, such as the Fraser alphabet used for the Lisu language spoken in Myanmar, China, and India, also include diacritics to indicate properties such as a word’s tone (how high or low pitched a sound is). A diacritic is added to a regular character, for example a vowel, indicating in more detail how that sound is supposed to be realized. In the case of Fraser, for example, M: refers to an[m] sound (written as M), which has a low tone (written as :).

Our second type of alphabetic system also often employs diacritics. Abjads, or consonant alphabets, represent consonants only; some prime examples are Arabic, Aramaic, and Hebrew. In abjads, vowels generally need to be deduced from context, as is illustrated by the Hebrew word for “computer”, shown on the left-hand side of Figure 1.2.

The Hebrew word in its character-by-character transliteration bšxm contains no vowels, but context may indicate the [a] and [e] sounds shown in the pronunciation of the word [mexav]. (Note that Hebrew is written right to left, so the m as the rightmost character of the written word is the first letter pronounced.) As shown in the middle and right-hand side of Figure 1.2, the context could also indicate different pronunciations with different meanings.

The situation with abjads often is a little more complicated than the one we just described, in that characters sometimes represent selected vowels, and often vowel diacritics are available.

A note on letter–sound correspondence As we have discussed, alphabets use letters to encode sounds. However, there is not always a simple correspondence between a word’s spelling and its pronunciation. To see this, we need look no further than English.
Writing systems used for human languages

English has a variety of non-letter–sound correspondences, which you probably labored through in first grade. First of all, there are words with the same spellings representing different sounds. The string *ough*, for instance, can be pronounced at least five different ways: “cough”, “tough”, “through”, “though”, and “hiccough”. Letters are not consistently pronounced, and, in fact, sometimes they are not pronounced at all; this is the phenomenon of silent letters. We can readily see these in “knee”, “debt”, “psychology”, and “mortgage”, among others. There are historical reasons for these silent letters, which were by and large pronounced at one time, but the effect is that we now have letters we do not speak.

Aside from inconsistencies of pronunciation, another barrier to the letter–sound correspondence is that English has certain conventions where one letter and one sound do not cleanly map to one another. In this case, the mapping is consistent across words; it just uses more or less letters to represent sounds. Single letters can represent multiple sounds, such as the *x* in “tax”, which corresponds to a *k* sound followed by an *s* sound. And multiple letters can consistently be used to represent one sound, as in the *th* in “the” or the *ti* in “revolution”.

Finally, we can alternate spellings for the same word, such as “doughnut” and “donut”, and homophones show us different words that are spelled differently but spoken the same, such as “colonel” and “kernel”.

Of course, English is not the only language with quirks in the letter–sound correspondences in its writing system. Looking at the examples in Figure 1.3 for Irish, we can easily see that each letter does not have an exact correspondent in the pronunciation.

<table>
<thead>
<tr>
<th>Spelling</th>
<th>Pronunciation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>samhradh</td>
<td>[sauruh]</td>
<td>‘summer’</td>
</tr>
<tr>
<td>scri’obhaim</td>
<td>[slgrIm]</td>
<td>‘I write’</td>
</tr>
</tbody>
</table>

Figure 1.3 Some Irish expressions

The issue we are dealing with here is that of *ambiguity* in natural language, in this case a letter potentially representing multiple possible sounds. Ambiguity is a recurring issue in dealing with human language that you will see throughout this book. For example, words can have multiple meanings (see Chapter 2); search queries can have different, often unintended meanings (see Chapter 4); and questions take on different interpretations in different contexts (see Chapter 6). In this case, writing systems can be designed that are unambiguous; phonetic alphabets, described next, have precisely this property.

**Phonetic alphabets** You have hopefully noticed the notation used within the brackets ([[]]). The characters used there are a part of the International Phonetic Alphabet (IPA). Several special alphabets for representing sounds have been developed, and probably the best known among linguists is the IPA. We have been discussing problems with letter–sound correspondences, and phonetic alphabets help us discuss these problems, as they allow for a way to represent all languages unambiguously using the same alphabet.
Each phonetic symbol in a phonetic alphabet is unambiguous: the alphabet is designed so that each speech sound (from any language) has its own symbol. This eliminates the need for multiple symbols being used to represent simple sounds and one symbol being used for multiple sounds. The problem for English is that the Latin alphabet, as we use it, only has 26 letters, but English has more sounds than that. So, it is no surprise that we find multiple letters like \textit{th} or \textit{sh} being used for individual sounds.

The IPA, like most phonetic alphabets, is organized according to the articulatory properties of each sound, an issue to which we return in Section 1.4.2. As an example of the IPA in use, we list some words in Figure 1.4 that illustrate the different vowels in English.

\begin{center}
\begin{tabular}{ll}
bead: & [bid] \\
bid: & [bid] \\
bade: & [be(i)d] \\
bed: & [brd] \\
bad: & [bæd] \\
boot: & [but] \\
book: & [bʊk] \\
bud: & [bad] \\
bode: & [bo(o)d] \\
bought: & [bɔt] \\
body: & [bædi] \\
\end{tabular}
\end{center}

\textbf{Figure 1.4} Example words for English vowels (varies by dialect)

At \url{http://purl.org/lang-and-comp/ipa} you can view an interactive IPA chart, provided by the University of Victoria’s Department of Linguistics. Most of the English consonants are easy to figure out, e.g., [b] in “boy”, but some are not obvious. For example, [θ] stands for the \textit{th} in “thigh”; [ð] for the \textit{th} in “thy”; and [ʃ] for the \textit{sh} in “shy”.

\subsection*{1.2.2 Syllabic systems}

\textbf{syllabic system} Syllabic systems are like alphabetic systems in that they involve a mapping between characters and sounds, but the units of sound are larger. The unit in question is called the \textit{syllable}. All human languages have syllables as basic building blocks of speech, but the rules for forming syllables differ from language to language. For example, in Japanese a syllable consists of a single vowel, optionally preceded by at most one consonant, and optionally followed by [m], [n], or [ŋ]. Most of the world’s languages, like Japanese, have relatively simple syllables. This means that the total number of possible syllables in the language is quite small, and that syllabic writing systems work well. But in English, the vowel can also be preceded by a sequence of several consonants (a so-called \textit{consonant cluster}), and there can also be a consonant cluster after the vowel. This greatly expands the number of possible syllables. You could design a syllabic writing system for English, but it would be unwieldy and difficult to learn, because there are so many different possible syllables.

\textbf{abugida} There are two main variants of syllabic systems, the first being \textit{abugidas} (or \textit{alphasyllabaries}). In these writing systems, the symbols are organized into families.