CHAPTER 1

INTRODUCTION

1.1 OVERVIEW OF THIS BOOK

This is a practical book that introduces the key ideas of text mining. It assumes that you have electronic texts to analyze and are willing to write programs using the programming language Perl. Although programming takes effort, it allows a researcher to do exactly what he or she wants to do. Interesting texts often have many idiosyncrasies that defy a software package approach.

Numerous, detailed examples are given throughout this book that explain how to write short programs to perform various text analyses. Most of these easily fit on one page, and none are longer than two pages. In addition, it takes little skill to copy and run code shown in this book, so even a novice programmer can get results quickly.

The first programs illustrating a new idea use only a line or two of text. However, most of the programs in this book analyze works of literature, which include the 68 short stories of Edgar Allan Poe, Charles Dickens's *A Christmas Carol*, Jack London's *The Call of the Wild*, Mary Shelley's *Frankenstein*, and Johann Wolfgang von Goethe's *Die Leiden des jungen Werthers*. All of these are in the public domain and are available from the Web for free. Since all the software to write the programs is also free, you can reproduce all the analyses of this book on your computer without any additional cost.

This book is built around the programming language Perl for several reasons. First, Perl is free. There are no trial or student versions, and anyone with access to the Web can download it as many times and on as many computers as desired. Second, Larry Wall created Perl to excel in processing computer text files. In addition, he has a background in

Practical Text Mining with Perl. By Roger Bilisoly

linguistics, and this influenced the look and feel of this computer language. Third, there are numerous additions to Perl (called modules) that are also free to download and use. Many of these process or manipulate text. Fourth, Perl is popular and there are numerous online resources as well as books on how to program in Perl. To get the most out of this book, download Perl to your computer and, starting in chapter 2, try writing and running the programs listed in this book.

This book does not assume that you have used Perl before. If you have never written any program in any computer language, then obtaining a book that introduces programming with Perl is advised. If you have never worked with Perl before, then using the free online documentation on Perl is useful. See sections 2.8 and 3.9 for some Perl references.

Note that this book is not on Perl programming for its own sake. It is devoted to how to analyze text with Perl. Hence, some parts of Perl are ignored, while others are discussed in great detail. For example, process management is ignored, but regular expressions (a text pattern methodology) is extensively discussed in chapter 2.

As this book progresses, some mathematics is introduced as needed. However, it is kept to a minimum, for example, knowing how to count suffices for the first four chapters. Starting with chapter 5, more of it is used, but the focus is always on the analysis of text while minimizing the required mathematics.

As noted in the preface, there are three underlying ideas behind this book. First, much text mining is built upon counting and text pattern matching. Second, although language is complex, there is useful information gained by considering the simpler properties of it. Third, combining a computer's ability to follow instructions without tiring and a human's skill with language creates a powerful team that can discover interesting properties of text. Someday, computers may understand and use a natural language to communicate, but for the present, the above ideas are a profitable approach to text mining.

1.2 TEXT MINING AND RELATED FIELDS

The core goal of text mining is to extract useful information from one or more texts. However, many researchers from many fields have been doing this for a long time. Hence the ideas in this book come from several areas of research.

Chapters 2 through 8 each focus on one idea that is important in text mining. Each chapter has many examples of how to implement this in computer code, which is then used to analyze one or more texts. That is, the focus is on analyzing text with techniques that require little or modest knowledge of mathematics or statistics.

The sections below describe each chapter's highlights in terms of what useful information is produced by the programs in each chapter. This gives you an idea of what this book covers.

1.2.1 Chapter 2: Pattern Matching

To analyze text, language patterns must be detected. These include punctuation marks, characters, syllables, words, phrases, and so forth. Finding string patterns is so important that a pattern matching language has been developed, which is used in numerous programming languages and software applications. This language is called regular expressions.

Literally every chapter in this book relies on finding string patterns, and some tasks developed in this chapter demonstrate the power of regular expressions. However, many tasks that are easy for a human require attention to detail when they are made into programs.

For example, section 2.4 shows how to decompose Poe's short story, "The Tell-Tale Heart," into words. This is easy for someone who can read English, but dealing with hyphenated words, apostrophes, conventions of using single and double quotes, and so forth all require the programmer's attention.

Section 2.5 uses the skills gained in finding words to build a concordance program that is able to find and print all instances of a text pattern. The power of Perl is shown by the fact that the result, program 2.7, fits within one page (including comments and blank lines for readability).

Finally, a program for detecting sentences is written. This, too, is a key task, and one that is trickier than it might seem. This also serves as an excellent way to show several of the more advanced features of regular expressions as implemented in Perl. Consequently, this program is written more than once in order to illustrate several approaches. The results are programs 2.8 and 2.9, which are applied to Dickens's *A Christmas Carol*.

1.2.2 Chapter 3: Data Structures

Chapter 2 discusses text patterns, while chapter 3 shows how to record the results in a convenient fashion. This requires learning about how to store information using indices (either numerical or string).

The first application is to tally all the word lengths in Poe's "The Tell-Tale Heart," the results of which are shown in output 3.4. The second application is finding out how often each word in Dickens's *A Christmas Carol* appears. These results are graphed in figure 3.1, which shows a connection between word frequency and word rank.

Section 3.7.2 shows how to combine Perl with a public domain word list to solve certain types of word games, for example, finding potential words in an incomplete crossword puzzle. Here is a chance to impress your friends with your superior knowledge of lexemes.

Finally, the material in this chapter is used to compare the words in the two Poe stories, "Mesmeric Revelations" and "The Facts in the Case of M. Valdemar." The plots of these stories are quite similar, but is this reflected in the language used?

1.2.3 Chapter 4: Probability

Language has both structure and unpredictability. One way to model the latter is by using probability. This chapter introduces this topic using language for its examples, and the level of mathematics is kept to a minimum. For example, Dickens's *A Christmas Carol* and Poe's "The Black Cat" are used to show how to estimate letter probabilities (see output 4.2).

One way to quantify variability is with the standard deviation. This is illustrated by comparing the frequencies of the letter *e* in 68 of Poe's short stories, which is given in table 4.1, and plotted in figures 4.3 and 4.4.

Finally, Poe's "The Unparalleled Adventures of One Hans Pfaall" is used to show one way that text samples behave differently from simpler random models such as coin flipping. It turns out that it is hard to untangle the effect of sample size on the amount of variability in a text. This is graphically illustrated in figures 4.5, 4.6, and 4.7 in section 4.6.1.

1.2.4 Chapter 5: Information Retrieval

One major task in information retrieval is to find documents that are the most similar to a query. For instance, search engines do exactly this. However, queries are short strings of

text, so even this application compares two texts: the query and a longer document. It turns out that these methods can be used to measure the similarity of two long texts.

The focus of this chapter is the comparison of the following four Poe short stories: "Hop Frog," "A Predicament," "The Facts in the Case of M. Valdemar," and "The Man of the Crowd." One way to quantify the similarity of any pair of stories is to represent each story as a vector. The more similar the stories, the smaller the angle between them. See output 5.2 for a table of these angles.

At first, it is surprising that geometry is one way to compare literary works. But as soon as a text is represented by a vector, and because vectors are geometric objects, it follows that geometry can be used in a literary analysis. Note that much of this chapter explains these geometric ideas in detail, and this discussion is kept as simple as possible so that it is easy to follow.

1.2.5 Chapter 6: Corpus Linguistics

Corpus linguistics is empirical: it studies language through the analysis of texts. At present, the largest of these are at a billion words (an average size paperback novel has about 100,000 words, so this is equivalent to approximately 10,000 novels). One simple but powerful technique is using a concordance program, which is created in chapter 2. This chapter adds sorting capabilities to it.

Even something as simple as examining word counts can show differences between texts. For example, table 6.2 shows differences in the following texts: a collection of business emails from Enron, Dickens's *A Christmas Carol*, London's *The Call of the Wild*, and Shelley's *Frankenstein*. Some of these differences arise from narrative structure.

One application of sorted concordance lines is comparing how words are used. For example, the word *body* in *The Call of the Wild* is used for live, active bodies, but in *Frankenstein* it is often used to denote a dead, lifeless body. See tables 6.4 and 6.5 for evidence of this.

Sorted concordance lines are also useful for studying word morphology (see section 6.4.3) and collocations (see section 6.5). An example of the latter is phrasal verbs (verbs that change their meaning with the addition of a word, for example, *throw* versus *throw up*), which is discussed in section 6.5.2.

1.2.6 Chapter 7: Multivariate Statistics

Chapter 4 introduces some useful, core ideas of probability, and this chapter builds on this foundation. First, the correlation between two variables is defined, and then the connection between correlations and angles is discussed, which links a key tool of information retrieval (discussed in chapter 5) and a key technique of statistics.

This leads to an introduction of a few essential tools from linear algebra, which is a field of mathematics that works with vectors and matrices, a topic introduced in chapter 5. With this background, the statistical technique of principal components analysis (PCA) is introduced and is used to analyze the pronoun use in 68 of Poe's short stories. See output 7.13 and the surrounding discussion for the conclusions drawn from this analysis.

This chapter is more technical than the earlier ones, but the few mathematical topics introduced are essential to understanding PCA, and all these are explained with concrete examples. The payoff is high because PCA is used by linguists and others to analyze many measurements of a text at once. Further evidence of this payoff is given by the references in section 7.6, which apply these techniques to specific texts.

1.2.7 Chapter 8: Clustering

Chapter 7 gives an example of a collection of texts, namely, all the short stories of Poe published in a certain edition of his works. One natural question to ask is whether or not they form groups. Literary critics often do this, for example, some of Poe's stories are considered early examples of detective fiction. The question is how a computer might find groups.

To group texts, a measure of similarity is needed, but many of these have been developed by researchers in information retrieval (the topic of chapter 5). One popular method uses the PCA technique introduced in chapter 7, which is applied to the 68 Poe short stories, and results are illustrated graphically. For example, see figures 8.6, 8.7 and 8.8.

Clustering is a popular technique in both statistics and data mining, and successes in these areas have made it popular in text mining as well. This chapter introduces just one of many approaches to clustering, which is explained with Poe's short stories, and the emphasis is on the application, not the theory. However, after reading this chapter, the reader is ready to tackle other works on the topic, some of which are listed in the section 8.4.

1.2.8 Chapter 9: Three Additional Topics

All books have to stop somewhere. Chapters 2 through 8 introduce a collection of key ideas in text mining, which are illustrated using literary texts. This chapter introduces three shorter topics.

First, Perl is popular in linguistics and text processing not just because of its regular expressions, but also because many programs already exist in Perl and are freely available online. Many of these exist as modules, which are groups of additional functions that are bundled together. Section 9.2 demonstrates some of these. For example, there is one that breaks text into sentences, a task also discussed in detail in chapter 2.

Second, this book focuses on texts in English, but any language expressed in electronic form is fair game. Section 9.3 compares Goethe's novel *Die Leiden des jungen Werthers* (written in German) with some of the analyses of English texts computed earlier in this book.

Third, one popular model of language in information retrieval is the so-called bag-of-words model, which ignores word order. Because word order does make a difference, how does one quantify this? Section 9.4 shows one statistical approach to answer this question. It analyzes the order that character names appear in Dickens's *A Christmas Carol* and London's *The Call of the Wild*.

1.3 ADVICE FOR READING THIS BOOK

As noted above, to get the most out of this book, download Perl to your computer. As you read the chapters, try writing and running the programs given in the text. Once a program runs, watching the computer print out results of an analysis is fun, so do not deprive yourself of this experience.

How to read this book depends on your background in programming. If you never used any computer language, then the subsequent chapters will require time and effort. In this case, buying one or more texts on how to program in Perl is helpful because when starting out, programming errors are hard to detect, so the more examples you see, the better. Although learning to program is difficult, it allows you to do exactly what you want to do, which is critical when dealing with something as complex as language.

If you have programmed in a computer language other than Perl, try reading this book with the help of the online documentation and tutorials. Because this book focuses on a subset of Perl that is most useful for text mining, there are commands and functions that you might want to use but are not discussed here.

If you already program in Perl, then peruse the listings in chapters 2 and 3 to see if there is anything that is new to you. These two chapters contain the core Perl knowledge needed for the rest of the book, and once this is learned, the other chapters are understandable.

After chapters 2 and 3, each chapter focuses on a topic of text mining. All the later chapters make use of these two chapters, so read or peruse these first. Although each of the later chapters has its own topic, these are the following interconnections. First, chapter 7 relies on chapters 4 and 5. Second, chapter 8 uses the idea of PCA introduced in chapter 7. Third, there are many examples of later chapters referring to the computer programs or output of earlier chapters, but these are listed by section to make them easy to check.

The Perl programs in this book are divided into *code samples* and *programs*. The former are often intermediate results or short pieces of code that are useful later. The latter are typically longer and perform a useful task. These are also boxed instead of ruled. The results of Perl programs are generally called *outputs*. These are also used for R programs since they are interactive.

Finally, I enjoy analyzing text and believe that programming in Perl is a great way to do it. My hope is that this book helps share my enjoyment to both students and researchers.