CHAPTER 7

Working with Text Data

In Chapter 4, we talked about two kinds of features that can represent properties of the data: continuous features that describe a quantity, and categorical features that are items from a fixed list. There is a third kind of feature that can be found in many applications, which is text. For example, if we want to classify an email message as either a legitimate email or spam, the content of the email will certainly contain important information for this classification task. Or maybe we want to learn about the opinion of a politician on the topic of immigration. Here, that individual's speeches or tweets might provide useful information. In customer service, we often want to find out if a message is a complaint or an inquiry. We can use the subject line and content of a message to automatically determine the customer's intent, which allows us to send the message to the appropriate department, or even send a fully automatic reply.

Text data is usually represented as strings, made up of characters. In any of the examples just given, the length of the text data will vary. This feature is clearly very different from the numeric features that we've discussed so far, and we will need to process the data before we can apply our machine learning algorithms to it.

Types of Data Represented as Strings

Before we dive into the processing steps that go into representing text data for machine learning, we want to briefly discuss different kinds of text data that you might encounter. Text is usually just a string in your dataset, but not all string features should be treated as text. A string feature can sometimes represent categorical variables, as we discussed in Chapter 5. There is no way to know how to treat a string feature before looking at the data.

There are four kinds of string data you might see:

- Categorical data
- Free strings that can be semantically mapped to categories
- Structured string data
- Text data

Categorical data is data that comes from a fixed list. Say you collect data via a survey where you ask people their favorite color, with a drop-down menu that allows them to select from "red," "green," "blue," "yellow," "black," "white," "purple," and "pink." This will result in a dataset with exactly eight different possible values, which clearly encode a categorical variable. You can check whether this is the case for your data by eyeballing it (if you see very many different strings it is unlikely that this is a categorical variable) and confirm it by computing the unique values over the dataset, and possibly a histogram over how often each appears. You also might want to check whether each variable actually corresponds to a category that makes sense for your application. Maybe halfway through the existence of your survey, someone found that "black" was misspelled as "blak" and subsequently fixed the survey. As a result, your dataset contains both "blak" and "black," which correspond to the same semantic meaning and should be consolidated.

Now imagine instead of providing a drop-down menu, you provide a text field for the users to provide their own favorite colors. Many people might respond with a color name like "black" or "blue." Others might make typographical errors, use different spellings like "gray" and "grey," or use more evocative and specific names like "midnight blue." You will also have some very strange entries. Some good examples come from the xkcd Color Survey, where people had to name colors and came up with names like "velociraptor cloaka" and "my dentist's office orange. I still remember his dandruff slowly wafting into my gaping yaw," which are hard to map to colors automatically (or at all). The responses you can obtain from a text field belong to the second category in the list, *free strings that can be semantically mapped to categories*. It will probably be best to encode this data as a categorical variable, where you can select the categories either by using the most common entries, or by defining categories that will capture responses in a way that makes sense for your application. You might then have some categories for standard colors, maybe a category "multicolored" for people that gave answers like "green and red stripes," and an "other" category for things that cannot be encoded otherwise. This kind of preprocessing of strings can take a lot of manual effort and is not easily automated. If you are in a position where you can influence data collection, we highly recommend avoiding manually entered values for concepts that are better captured using categorical variables.

Often, manually entered values do not correspond to fixed categories, but still have some underlying *structure*, like addresses, names of places or people, dates, telephone

numbers, or other identifiers. These kinds of strings are often very hard to parse, and their treatment is highly dependent on context and domain. A systematic treatment of these cases is beyond the scope of this book.

The final category of string data is freeform *text data* that consists of phrases or sentences. Examples include tweets, chat logs, and hotel reviews, as well as the collected works of Shakespeare, the content of Wikipedia, or the Project Gutenberg collection of 50,000 ebooks. All of these collections contain information mostly as sentences composed of words.[1] For simplicity's sake, let's assume all our documents are in one language, English.[2] In the context of text analysis, the dataset is often called the *corpus*, and each data point, represented as a single text, is called a *document*. These terms come from the *information retrieval* (IR) and *natural language processing* (NLP) community, which both deal mostly in text data.

Example Application: Sentiment Analysis of Movie Reviews

As a running example in this chapter, we will use a dataset of movie reviews from the IMDb (Internet Movie Database) website collected by Stanford researcher Andrew Maas.[3] This dataset contains the text of the reviews, together with a label that indicates whether a review is "positive" or "negative." The IMDb website itself contains ratings from 1 to 10. To simplify the modeling, this annotation is summarized as a two-class classification dataset where reviews with a score of 6 or higher are labeled as positive, and the rest as negative. We will leave the question of whether this is a good representation of the data open, and simply use the data as provided by Andrew Maas.

After unpacking the data, the dataset is provided as text files in two separate folders, one for the training data and one for the test data. Each of these in turn has two subfolders, one called *pos* and one called *neg*:

1 Arguably, the content of websites linked to in tweets contains more information than the text of the tweets themselves.

2 Most of what we will talk about in the rest of the chapter also applies to other languages that use the Roman alphabet, and partially to other languages with word boundary delimiters. Chinese, for example, does not delimit word boundaries, and has other challenges that make applying the techniques in this chapter difficult.

3 The dataset is available at *http://ai.stanford.edu/~amaas/data/sentiment/*.