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/*.

In[2]:

```
!tree -L 2 data/aclImdb
```

Out[2]:

```
data/aclImdb
├── test
│   ├── neg
│   └── pos
└── train
    ├── neg
    └── pos

6 directories, 0 files
```

The *pos* folder contains all the positive reviews, each as a separate text file, and similarly for the *neg* folder. There is a helper function in `scikit-learn` to load files stored in such a folder structure, where each subfolder corresponds to a label, called `load_files`. We apply the `load_files` function first to the training data:

In[3]:

```
from sklearn.datasets import load_files

reviews_train = load_files("data/aclImdb/train/")
# load_files returns a bunch, containing training texts and training labels
text_train, y_train = reviews_train.data, reviews_train.target
print("type of text_train: {}".format(type(text_train)))
print("length of text_train: {}".format(len(text_train)))
print("text_train[1]:\n{}".format(text_train[1]))
```

Out[3]:

```
type of text_train:  <class 'list'>
length of text_train:  25000
text_train[1]:
b'Words can\'t describe how bad this movie is. I can\'t explain it by writing
  only. You have too see it for yourself to get at grip of how horrible a movie
  really can be. Not that I recommend you to do that. There are so many
  clich\xc3\xa9s, mistakes (and all other negative things you can imagine) here
  that will just make you cry. To start with the technical first, there are a
  LOT of mistakes regarding the airplane. I won\'t list them here, but just
  mention the coloring of the plane. They didn\'t even manage to show an
  airliner in the colors of a fictional airline, but instead used a 747
  painted in the original Boeing livery. Very bad. The plot is stupid and has
  been done many times before, only much, much better. There are so many
  ridiculous moments here that i lost count of it really early. Also, I was on
  the bad guys\' side all the time in the movie, because the good guys were so
  stupid. "Executive Decision" should without a doubt be you\'re choice over
  this one, even the "Turbulence"-movies are better. In fact, every other
  movie in the world is better than this one.'
```

You can see that `text_train` is a list of length 25,000, where each entry is a string containing a review. We printed the review with index 1. You can also see that the review contains some HTML line breaks (`<br />`). While these are unlikely to have a

large impact on our machine learning models, it is better to clean the data and remove this formatting before we proceed:

In[4]:

```
text_train = [doc.replace(b"<br />", b" ") for doc in text_train]
```

The type of the entries of `text_train` will depend on your Python version. In Python 3, they will be of type `bytes` which represents a binary encoding of the string data. In Python 2, `text_train` contains strings. We won't go into the details of the different string types in Python here, but we recommend that you read the Python 2 and/or Python 3 documentation regarding strings and Unicode.

The dataset was collected such that the positive class and the negative class balanced, so that there are as many positive as negative strings:

In[5]:

```
print("Samples per class (training): {}".format(np.bincount(y_train)))
```

Out[5]:

```
Samples per class (training): [12500 12500]
```

We load the test dataset in the same manner:

In[6]:

```
reviews_test = load_files("data/aclImdb/test/")
text_test, y_test = reviews_test.data, reviews_test.target
print("Number of documents in test data: {}".format(len(text_test)))
print("Samples per class (test): {}".format(np.bincount(y_test)))
text_test = [doc.replace(b"<br />", b" ") for doc in text_test]
```

Out[6]:

```
Number of documents in test data: 25000
Samples per class (test): [12500 12500]
```

The task we want to solve is as follows: given a review, we want to assign the label "positive" or "negative" based on the text content of the review. This is a standard binary classification task. However, the text data is not in a format that a machine learning model can handle. We need to convert the string representation of the text into a numeric representation that we can apply our machine learning algorithms to.

Representing Text Data as a Bag of Words

One of the most simple but effective and commonly used ways to represent text for machine learning is using the *bag-of-words* representation. When using this representation, we discard most of the structure of the input text, like chapters, paragraphs, sentences, and formatting, and only count *how often each word appears in each text* in