CHAPTER 7
Recurrent Neural Networks

So far in this book, we've introduced you to the use of deep learning to process various types of inputs. We started from simple linear and logistic regression on fixed dimensional feature vectors, and then followed up with a discussion of fully connected deep networks. These models take in arbitrary feature vectors with fixed, predetermined sizes. These models make no assumptions about the type of data encoded into these vectors. On the other hand, convolutional networks place strong assumptions upon the structure of their data. Inputs to convolutional networks have to satisfy a locality assumption that allows for the definition of a local receptive field.

How could we use the networks that we've described thus far to process data like sentences? Sentences do have some locality properties (nearby words are typically related), and it is indeed possible to use a one-dimensional convolutional network to process sentence data. That said, most practitioners resort to a different type of architecture, the recurrent neural network, in order to handle sequences of data.

Recurrent neural networks (RNNs) are designed natively to allow deep networks to process sequences of data. RNNs assume that incoming data takes the form of a sequence of vectors or tensors. If we transform each word in a sentence into a vector (more on how to do this later), sentences can be fed into RNNs. Similarly, video (considered as a sequence of images) can similarly be processed by an RNN. At each sequence position, an RNN applies an arbitrary nonlinear transformation to the input at that sequence location. This nonlinear transformation is shared for all sequence steps.

The description in the previous paragraph is a little abstract, but turns out to be immensely powerful. In this chapter, you will learn more details about how RNNs are structured and about how to implement RNNs in TensorFlow. We will also discuss how RNNs can be used in practice to perform tasks like sampling new sentences or generating text for applications such as chatbots.

The case study for this chapter trains a recurrent neural network language model on the Penn Treebank corpus, a body of sentences extracted from *Wall Street Journal* articles. This tutorial was adapted from the TensorFlow official documentation tutorial on recurrent networks. (We encourage you to access the original tutorial on the TensorFlow website if you're curious about the changes we've made.) As always, we recommend that you follow along with the code in the GitHub repo associated with this book.

Overview of Recurrent Architectures

Recurrent architectures are useful for modeling very complex time varying datasets. Time varying datasets are traditionally called *time-series*. Figure 7-1 displays a number of time-series datasets.

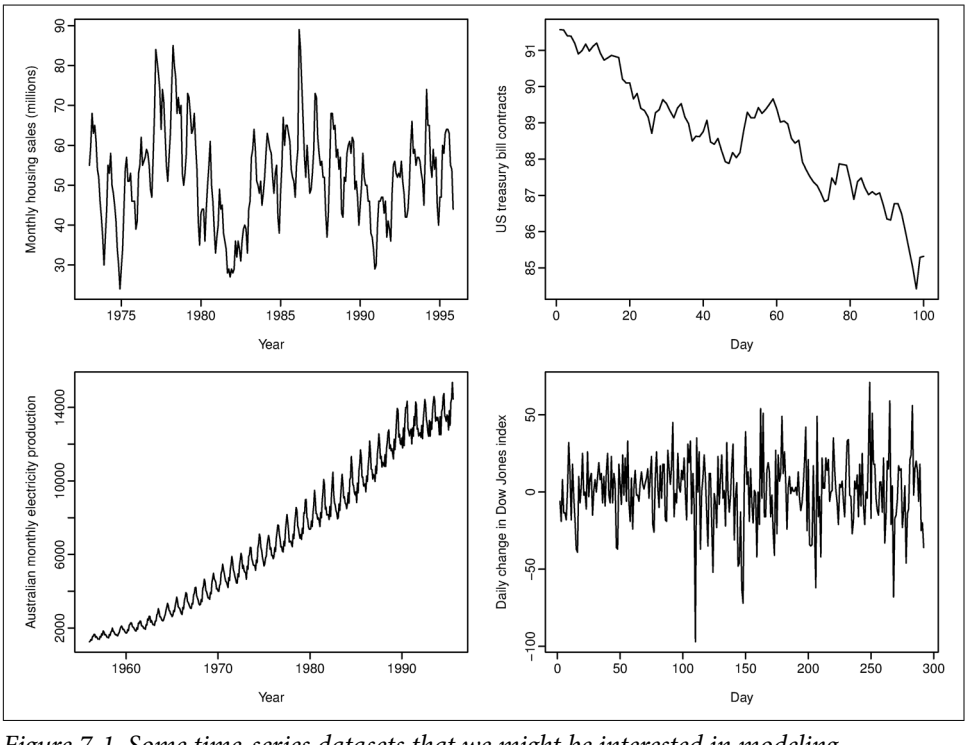

Figure 7-1. Some time-series datasets that we might be interested in modeling.

In time-series modeling, we design learning systems that are capable of learning the evolution rule that models how the future of the system at hand evolves depending on the past. Mathematically, let's suppose that at each time step, we receive a datapoint x_t where t is the current time. Then, time-series methods seek to learn some function f such that