```
Epoch 1/3
16750/16750 [==============================] - 112s - loss: 0.6623 - acc: 0.5935
Epoch 2/3
16750/16750 [==============================] - 113s - loss: 0.5159 - acc: 0.7484
Epoch 3/3
16750/16750 [==============================] - 113s - loss: 0.4502 - acc: 0.7981
Accuracy: 82.82%
```

Listing 26.13: Output from Running the LSTM Model with Dropout on the Gates.

We can see that the LSTM specific dropout has a more pronounced effect on the convergence of the network than the layer-wise dropout. As above, the number of epochs was kept constant and could be increased to see if the skill of the model can be further lifted. Dropout is a powerful technique for combating overfitting in your LSTM models and it is a good idea to try both methods, but you may bet better results with the gate-specific dropout provided in Keras.

26.3 LSTM and CNN For Sequence Classification

Convolutional neural networks excel at learning the spatial structure in input data. The IMDB review data does have a one-dimensional spatial structure in the sequence of words in reviews and the CNN may be able to pick out invariant features for good and bad sentiment. This learned spatial features may then be learned as sequences by an LSTM layer. We can easily add a one-dimensional CNN and max pooling layers after the Embedding layer which then feed the consolidated features to the LSTM. We can use a smallish set of 32 features with a small filter length of 3. The pooling layer can use the standard length of 2 to halve the feature map size. For example, we would create the model as follows:

```
model = Sequential()
model.add(Embedding(top_words, embedding_vecor_length, input_length=max_review_length))
model.add(Convolution1D(nb_filter=32, filter_length=3, border_mode='same',
    activation='relu'))
model.add(MaxPooling1D(pool_length=2))
model.add(LSTM(100))
model.add(Dense(1, activation='sigmoid'))
```

Listing 26.14: Define LSTM and CNN Model.

The full code listing with a CNN and LSTM layers is listed below for completeness.

```
# LSTM and CNN for sequence classification in the IMDB dataset
import numpy
from keras.datasets import imdb
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers.convolutional import Convolution1D
from keras.layers.convolutional import MaxPooling1D
from keras.layers.embeddings import Embedding
from keras.preprocessing import sequence
# fix random seed for reproducibility
numpy.random.seed(7)
# load the dataset but only keep the top n words, zero the rest
top_words = 5000
```

```
(X_train, y_train), (X_test, y_test) = imdb.load_data(nb_words=top_words)
# truncate and pad input sequences
max_review_length = 500
X_train = sequence.pad_sequences(X_train, maxlen=max_review_length)
X_test = sequence.pad_sequences(X_test, maxlen=max_review_length)
# create the model
embedding_vecor_length = 32
model = Sequential()
model.add(Embedding(top_words, embedding_vecor_length, input_length=max_review_length))
model.add(Convolution1D(nb_filter=32, filter_length=3, border_mode='same',
    activation='relu'))
model.add(MaxPooling1D(pool_length=2))
model.add(LSTM(100))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
print(model.summary())
model.fit(X_train, y_train, nb_epoch=3, batch_size=64)
# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

Listing 26.15: LSTM and CNN Model for the IMDB Dataset.

Running this example provides the following output.

```
Epoch 1/3
16750/16750 [==============================] - 58s - loss: 0.5186 - acc: 0.7263
Epoch 2/3
16750/16750 [==============================] - 58s - loss: 0.2946 - acc: 0.8825
Epoch 3/3
16750/16750 [==============================] - 58s - loss: 0.2291 - acc: 0.9126
Accuracy: 86.36%
```

Listing 26.16: Output from Running the LSTM and CNN Model.

We can see that we achieve similar results to the first example although with less weights and faster training time. I would expect that even better results could be achieved if this example was further extended to use dropout.

26.4 Summary

In this project you discovered how to develop LSTM network models for sequence classification predictive modeling problems. Specifically, you learned:

- How to develop a simple single layer LSTM model for the IMDB movie review sentiment classification problem.
- How to extend your LSTM model with layer-wise and LSTM-specific dropout to reduce overfitting.
- How to combine the spatial structure learning properties of a Convolutional Neural Network with the sequence learning of an LSTM.