Sentiment Analysis of Movie Reviews

Machine Learning Group Project 2
Masters in Computer Science
Department of Computer Science
University of North Carolina at charlotte
NC | 28262

Instructor

Dr. Wlodek Zadrozny

Submitted by
Gopichand Tadapaneni
Id: 801274664

Problem statement:

The goal of the dataset problem is to predict whether a movie review has positive sentiment ("It was a good movie") or negative sentiment ("The film was a waste of time").

Model Built:

1) Loading pre-trained GloVe embeddings:

Each line of the text has multiple values separated by spaces. The first item of each row is the word, and the rest of the items are the values of the vector for each dimension. So, in the 100-dimensional file, each row will have 101 columns. These vectors need to be loaded up in memory. Code for this is:

```
for line in glove_file:
    records = line.split()
    word = records[0]
    vector_dimensions = asarray(records[1:], dtype='float32')
    embeddings_dictionary[word] = vector_dimensions
glove_file.close()
```

2) Creating a pre-trained embedding matrix using GloVe:

we have a dataset, its vocabulary, and a dictionary of GloVe words and their corresponding vectors. However, there is no correlation between these two vocabularies. The way to connect them is through the creation of an embedding matrix. First, let's initialize an embedding matrix of zeros.

```
embedding_matrix = zeros((vocab_size, 100))
for word, index in token.word_index.items():
    embedding_vector = embeddings_dictionary.get(word)
    if embedding_vector is not None:
        embedding_matrix[index] = embedding_vector
Embedding_matrix
```

3) K-fold Validation:

Since we have only 250 reviews we can use K-fold validation to build our model. So that we can get more accurate results. We are using the 10-fold validation that is k=10.

```
k = 10
num_val_samples = len(X_train) // 10
num_epochs = 100
all_scores = []
for i in range(k):
    print(f"Processing fold #{i}")
```

```
    val_data = X_train[i * num_val_samples: (i + 1) *
num_val_samples]
    val_targets = train_label[i * num_val_samples: (i + 1) *
num_val_samples]
    partial_train_data = np.concatenate(
        [X_train[:i * num_val_samples],
         X_train[(i + 1) * num_val_samples:]],
        axis=0)
    partial_train_targets = np.concatenate(
        [train_label[:i * num_val_samples],
         train_label[(i + 1) * num_val_samples:]],
        axis=0)
```

4) Convolutional Neural Network with Sequential Model:

```
model = Sequential()
embedding_layer = Embedding(vocab_size, 100,
weights=[embedding_matrix], input_length=maxlen ,trainable=False)
model.add(embedding_layer)
model.add(Conv1D(64, 5, activation='relu'))
model.add(Conv1D(128,5, activation = 'relu'))
model.add(MaxPooling1D((2)))
model.add(Dropout(0.25))
model.add(Dense(64, activation = 'relu'))
model.add(Flatten())
model.add(Dense(1, activation='sigmoid'))
```

For the purpose of sentiment analysis, I have used a keras Sequential model by using the convolutional neural network. For this neural network, input is an embedding layer. In this model we have two parts: they start with a series of pooling and convolution layers, and they end with a densely connected classifier. The first part is called the Convolution base of the model.

4.1) Convolution part of the model:

In this part we apply kernel filters and pooling to the layers to learn global patterns in their input feature space, whereas convolution layers learn local patterns. In this we are using a 5*5 kernel filters. Also I used a dropout of 0.25(Dropout is one of the most effective and most commonly used regularization techniques for neural networks. It is applied to a layer, consisting of randomly dropping out a number of output features of the layer during training) and relu as an activation function (rectified linear unit is a function meant to zero out negative values).

4.2) Classifier part:

In this part we used only one layer and it is the last layer of the model. In this layer there is sigmoid activation function as the classification is Binary class classification (A sigmoid function placed as the last layer of a machine learning model can serve to convert the model's output into a probability score, which can be easier to work with and interpret.)

Link to my google colab notebook containing the code:
https://drive.google.com/drive/folders/10pp83B_bQ7uga3kb3g8pHjMkeaqoRZIG?usp=sharing