Imdb Movie Review Sentiment Analysis

(By: Shyam Kumar Sodankoor(19230735) and Harshith Shankar Tarikere Ravikumar(19230323))

Sentiment Analysis is one of the important applications of Natural Language Processing. Many of the E-commerce websites use this for understanding the quality and the popularity of a product. It is also used in the movie industry for understanding the popularity of a movie, actor, director and so on. In this assignment the Imdb Movie Review Dataset is considered for creating a neural network model for predicting the sentiment based on the review.

Section 1 Data Preprocessing

Imdb Movie Review Dataset is a dataset of 50000 movie reviews which consists of 25000 training data and 25000 testing data, each of which consists of 12500 positive reviews and 12500 negative reviews.

Each review is in a separate text file. The file structure of positive reviews training data is aclImdb/train/pos/<files> and negative reviews training data is aclImdb/train/neg/<files>. Similarly for the testing data they are under test folder instead of train folder.

All the reviews are read one by one from the files and stored in a dataframe with the corresponding sentiments. They are stored in the dataframe in the following order

0 to 12499 -> Positive Training data

12500 to 24999 -> Negative Training data

25000 to 37499 -> Positive Testing data

37500 to 49999 -> Negative Testing Data

In addition, for each of the reviews, some additional preprocessing is done. They include removing html tags, removing punctuations and numbers, removal single character words, removing multiple spaces, removing special characters and filtering out stop words.

Section 2 Choice of Neural Network

The most general approach of architecture selection followed in the NLP applications is the use of the Sequential Models.

Recurrent Neural Network is the basic sequential model which finds good application in the NLP domain for multiple NLP tasks. They consider not only the words in a sentence but also the sequence in which they are present to understand a sentence.

The output of any given hidden node is a function of input vector X_t and the state S of the node as can be seen from Figure 2.1

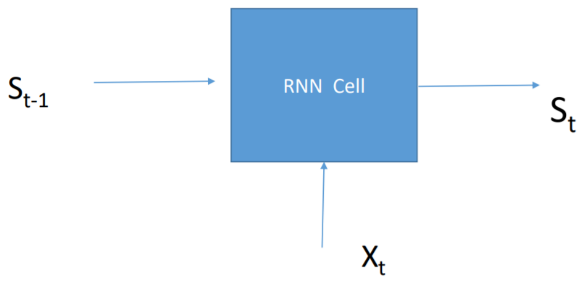

Figure 2.1 Structure of any given node in a Recurrent Neural Network

$$S_t = g(WS_{t-1} + UX_t + b)$$

It is governed by the above equation where W, U and b are learnable parameters.

They are pretty good in NLP applications but they have some drawbacks due to which other Neural Network Architectures perform better than a traditional RNN. Some of them include the vanishing gradient problem where the value of weights almost vanish for large sentences. Another problem with RNN is Information Morphing, where the architecture lacks the ability to hold memory for more than a few time steps

Long-Short Term Memory(LSTM) architecture solves these two issues by introducing a memory cell along with the sequence model as seen in the RNN. This is done by adding new gate vectors: Read gate vector, Forget gate Vector and Write gate vector.

The states are defined as given below

New Candidate State := f (Read gate vector ◦ Current state, Input$_t$)

Current state := (Forget gate vector ◦ Previous state) + (Write gate vector ◦ New Candidate State)

All the gates are function of current input and the previous state and the activation functions for the gate vectors is the sigmoid function

Read gate vector = f_g (Curr Input, Prev State)

Forget gate vector = f_f (Curr Input, Prev State)

A single LSTM function is as shown below

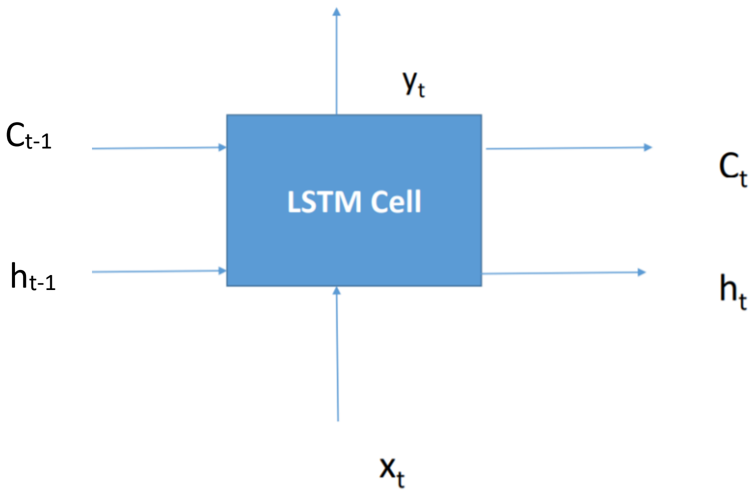

Figure 2.2 LSTM Cell

where C_t is the memory cell and h_t is the shadow state(gate)

As LSTM resolves the issues the RNN faces, LSTM is used for this application

Section 3 LSTM implementation using Keras

The neural network architecture needs the sentences or reviews to be converted into a numerical format to be able to work. Each of these reviews is converted to a numerical form by using a Tokenizer object of the Keras library which takes the parameter of num_words. So it creates an array of size num_words which represent the top num_words in the corpus of reviews. This array is created for each of the reviews with array values based on the words in the review.

Then each of these reviews need to be converted to word embeddings which is a set of numbers in a fixed size array. A word embedding consists of several features and corresponding value of each word in the corresponding feature tells how much that word is important with regards to that feature for each review.

LSTM model is created by using keras library. As we process the data sequentially. We initially build a sequential model using the Sequential() function. Then an embedding layer is added with 5000 features and 128 nodes in the next layer. LSTM layer is added as the next layer. The next layer is dense layer with 64 nodes with the Relu as the activation function and the final layer consists of a single nodes with sigmoid activation function.

The model is characterized with the binary crossentropy loss function and Adam optimizer. Finally the model is trained using the training data(25000 reviews).

The accuracy of the model is calculated based on the predicted output of the testing data in comparison with the actual output of the test data.

Section 4 Evaluation

Evaluation is done by running the LSTM for 10 epochs and the training and validation loss is plotted against the number of epochs. The plot is as shown in Figure 4.1

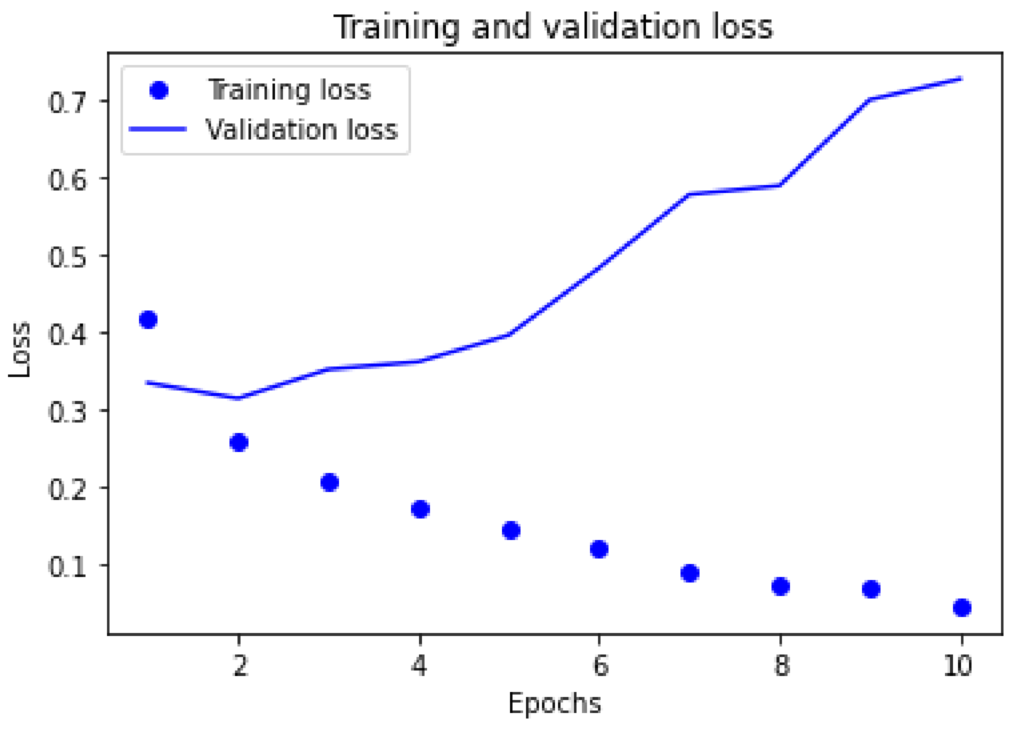

Figure 4.1 Plot of Training and Validation Loss vs Number of Epochs

The training loss goes on decreasing with increase in the number of epochs. Validation loss decreases as epochs increase from 1 to 2 but then it goes on increasing with increase in epochs. So we obtain the highest accuracy of around 87%.

Section 5 Comparison with Naïve Implementation

The performance of the LSTM is compared with the Naïve Bayes algorithm for the Imdb sentiment analysis task

Naïve Bayes gives an accuracy of 83.5% for this task

LSTM using Keras gives an accuracy of 87% for 2 epochs

This suggests that LSTM gives us a 3.5% increase in accuracy in comparison to Naïve Bayes.

The disadvantage with using a Neural Network architecture it is more compute intensive. It takes more time in learning the model.

Contributions

- Shyam Kumar Sodankoor: Data Preprocessing, Choice of Neural Network Explanation and Naïve Bayes
- Harshith Shankar Tarikere Ravikumar: LSTM Implementation using Keras, Model evaluation using the Training and Validation Loss vs Number of Epochs

References

1] Equations and Figures in Section 2 used from *Lecture 2: Sequential Data Processing by Colm O' Riordan*