Problem 3: Sentiment Analysis

Overview:

We downloaded the IMDB Movie Review Dataset and copied it in the **data** directory in a folder called **aclImdb**. The data consists of two folders namely **Train** and **Test**. This dataset contains movie reviews along with their associated binary sentiment polarity labels. It is intended to serve as a benchmark for sentiment classification. This report outlines how the dataset was preprocessed, and how the network was trained to classify the review as positive or negative.

Dataset:

The core dataset contains 50,000 reviews split evenly into 25k **train** (12.5k **pos** & 12.5k **neg**) and 25k **test** (12.5k **pos** & 12.5k **neg**) sets. The overall distribution of labels is balanced (25k **pos** and 25k **neg**). In the labeled train/test sets, a negative review has a score <= 4 out of 10 and a positive review has a score >= 7 out of 10. Thus, reviews with more neutral ratings are not included in the train/test sets.

Data Uploading:

The data was uploaded from the local directory by giving path of our directory and reading the text files along with their polarity into train and test dataframes respectively. The ‘os’ and ‘os.path’ modules include many functions to interact with the operating system and the file system. We passed the path in listdir () which returns a list containing the names of the entries in the directory given by path. After this we iterated a for loop to enter only pos and neg folders resp and read files from their by using open() function to open file and f.read() to read it. Further, we appended the reviews and their polarity to our train and test dataframes. We assigned Polarity as 1 for reviews coming from pos folder and Polarity as 0 for reviews coming from neg folder

Data Preprocessing

Part 1. Data Preprocessing done using NLTK ‘stopwords’:

The ‘stopwords’ are a list of words given below in the nltk.corpus library. These words are used in grammar to form sentences but with the sentiment analysis perspective they are of no use therefore, we removed these from training and testing datasets.
{'off', 'too', "shouldn't", 'because', 'them', "you're", 'we', 'on', 'so', 'ain', "wasn't", 'about', 'be', 'is', 'theirs', "weren't", 'your', 'shan', 'she', "haven't", "you've", 'are', 'themselves', 'further', 'ours', 'itself', 'needn', "won't", 'he', 'most', 't', "hasn't", 'having', 'such', 'then', 'aren', 'm', 's', "mightn't", 'it', 'yourself', 'her', "hadn't", 'was', 'haven', 'more', 'after', 'mustn', 'in', 'wasn', "you'll", 'or', 'herself', 'been', 'himself', 'ma', 'his', 'how', 'do', 'now', 'very', 'yourselves', 'any', 'have', 'between', 'above', 'just', 'by', 'but', 'these', "isn't", 'y', 'won', 'll', 'up', "don't", 'does', 'they', "she's", 'no', 'doesn', "wouldn't", 'there', "didn't", 'once', 'my', 'i', 'some', 'that', 'its', 'didn', 'mightn', 'should', 'here', 'why', 've', 'below', "mustn't", 'at', "you'd", 'each', 'of', 'out', 'me', 'both', 'an', 'again', 'yours', 'to', "aren't", "couldn't", 'couldn', 'not', 'than', 'what', 'few', 'has', 'all', 'ourselves', 'against', "that'll", 'this', 'for', 're', 'down', 'were', 'over', 'other', 'if', "shan't", 'through', 'can', 'who', 'did',

'own', 'from', 'wouldn', 'a', 'where', 'myself', 'our', 'nor', 'those', 'am', 'd', 'hers', 'will', 'before', 'only', "should've", 'hasn', 'o', 'their', 'hadn', 'while', 'and', 'with', 'when', 'which', "doesn't", 'you', 'as', 'him', "needn't", 'whom', 'being', 'doing', 'until', 'during', 'don', 'shouldn', 'isn', 'weren', 'the', 'had', 'same', 'under', "it's", 'into'}

Part 2. Data Preprocessing done using regex (Regular Expression):

1. Removed special characters such as . ; : ! ‘ ? , " ( ) [ ] * using regex.
   These special characters create noise in our dataset and are not at all useful while training our model as they play no major role in deciding the polarity of the movie reviews. That’s why we removed them using the basic regular expressions and replaced them with no space in our training and testing reviews dataset.
2. Removed HTML tags such as
 using regex.
   As these movie reviews were gathered from the movies review pages therefore, they had
 which is an HTML tag. As this is not required in the reviews so we removed it using the basic regex expression and replaced it with space.
3. Only english character will remain using regular expression "[^a-zA-Z]".
   As the reviews might be containing numbers therefore, we cleaned that by using the "[^a-zA-Z]" regular expression which looks for non character things in the dataset. We replaced the identified numbers with spaces.

Part 3. Shuffle the Data:

As our review were in order i.e., the first 12.5k reviews were pos and next 12.5k reviews were neg in the train and test dataset. Therefore, we shuffled them using. reset_index on the data frame so that are reviews get shuffled along with their polarities. This is done to train the model in a better manner.

Network Modelling

Part 1. Feed Forward Neural Network:

Tokenization:

- Using Keras, tokenization and mapping to numbers are done on the words.
- When fitting, use all data from train data set, which prevents model from errors.

Padding:

- Max length is set, if length more than max length, zero value will replace that place.
- We need to pad the sequences to get the same length on all of them. i.e. we add zeroes to the small sequences and truncate the larger ones.

Train_Val Split:

- Train_val_test splitting is done where in the 70% of train data is further split into 70-30%. The original 30% test data was kept untouched for testing purpose only.

Model training:

- Sequential model with Adam optimizer and binary cross entropy as a loss function

We added 5 layers:

- Embedded layer
- Convolution layer with global average pooling
    - Filters = 16
    - kernel_size = 2
    - padding = valid
    - activation = relu
- 2 Dense layers with units 32 and 1

The summary of the final chosen model is as below

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_6 (Embedding)      (None, None, 16)          1448624
_________________________________________________________________
dropout_18 (Dropout)         (None, None, 16)          0
_________________________________________________________________
conv1d_6 (Conv1D)            (None, None, 16)          528
_________________________________________________________________
global_average_pooling1d_6 ( (None, 16)                0
_________________________________________________________________
dropout_19 (Dropout)         (None, 16)                0
_________________________________________________________________
dense_12 (Dense)             (None, 32)                544
_________________________________________________________________
dropout_20 (Dropout)         (None, 32)                0
_________________________________________________________________
dense_13 (Dense)             (None, 1)                 33
=================================================================
Total params: 1,449,729
Trainable params: 1,449,729
Non-trainable params: 0
_________________________________________________________________
```

Accuracy Analysis

1. **Feed Forward Neural Network**: Chosen Model
    a. Training accuracy: 96.73%
    b. Validation accuracy (30% of training data): 89.35%
    c. Testing accuracy: 87.88%

2. Logistic Regression with regularization parameter = 0.01:
    Logistic Regression is a good baseline model for us to use for several reasons. They are easy to interpret, linear models tend to perform well on sparse datasets like this one, and they learn very fast compared to other algorithms.
    a. Testing accuracy: 87.72%

3. CNN:
    a. Training accuracy: 54.16%
    b. Validation accuracy (30% of training data): 51.78%
    c. Testing accuracy: 51.64%

4. LSTM:
    a. Training accuracy: 50.03%
    b. Validation accuracy: 49%

All the above models were executed in a separate ipynb file to conclude on the best model.

Conclusion and Final Comments:

While training the Feed Forward Neural Network we encountered the loss of up to 0.10 which is a positive sign towards achieving good accuracy. As mentioned in Table 2 of "Learning Word Vectors for Sentiment Analysis" (the research paper) where we can find accuracies for different features against Our dataset falls in the range from 87-89% comparing our model's accuracy with this information we are almost on track with a test accuracy of around 87%.