1 Introduction

Sentiments analysis is a data mining technique that is used to measure and understand people's opinion and inclination through NLP(Natural Language Processing). Computational linguistics and text analysis are used to analyze the infromation.

The most popular word embedding algorithms are Google's Word2Vec, Stanford's GloVe. In this we use the famous tensorflow libraary to create a model to identify an intent behind a piece of text which in this case is a movie review on IMDb a popular online database of information related to films, television programs, home videos, video games, and streaming content.

2 Data

This dataset contains movie reviews along with their associated binary sentiment polarity labels. It is intended to serve as a benchmark for sentiment classification.

The core dataset contains 25,000 reviews split evenly. The overall distribution of labels is balanced (12.5k pos and 12.5k 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.

There are two top-level directories $[train/, test/]$ corresponding to the training and test sets. Each contains $[pos/, neg/]$ directories for the reviews with binary labels positive and negative. Within these directories, reviews are stored in text files named following the convention $[[id]\_[rating].txt]$ where $[id]$ is a unique id and [rating] is the star rating for that review on a 1-10 scale.

For example, the file $[test/pos/200\_8.txt]$ is the text for a positive-labeled test set example with unique id 200 and star rating 8/10 from IMDb. The $[train/unsup/]$ directory has 0 for all ratings because the ratings are omitted for this portion of the dataset.

The positive reviews are extracted into a $IMDB\_POS.csv$ file and the negative reviews are extracted to $IMDB\_NEG.csv$ file. Both files are then combined into a single dataset called the $IMDB\_Movie.csv$ file after the review goes through filtering techniques mentioned in the following section.

Below is the sample screen shot of the dataset. The review column has the movie review by a critiqe the idex is the unique ID for the neagative or positive review , the rating column mentions the rating given out of 10 and the score is the one which classifies the movie review data into either positive or negative.

	A	B	C	D	E
1		review	index	rating	score
2	0	stori man unnatur feel pig start o	0	3	negative
3	1	airport start brand new luxuri pla	10000	4	negative
4	2	film lack someth couldnt put fing	10001	4	negative
5	3	sorri everyon know suppos art fil	10002	1	negative
6	4	littl parent took along theater se	10003	1	negative
7	5	appear mani critic find idea wooo	10004	3	negative

Figure 1: IMDb movie review extracted to .csv file

3 Text filtering

The data or the movie reviews have a lot of noise and has to be filtered for further analysis.

Following are the techniques used:

1. Removing special characters, white spaces, numbers, brackets, HTML tags etc.

```
def filter_text(review):
    # parsing html content
    soup = BeautifulSoup(review, "html.parser")
    filter_one_text = soup.get_text()
    # convert caps to lower case
    filter_one_text = filter_one_text.lower()
    # remove brackets and other characters
    filter_one_text = re.sub(
        '\[[^]]*\]', '', filter_one_text)
    # remove quotes double quotes
    filter_one_text = re.sub(
        '['''"",,,]', '', filter_one_text)
    # removing whitespaces
    filter_one_text = re.sub('\w*\d\w*', '', filter_one_text)
    # removing next line
    filter_one_text = re.sub('\n', '', filter_one_text)
    # removing numbers
    filter_one_text = re.sub(
        '[^a-zA-z0-9\s]', '', filter_one_text)
        .
        .
```

2. Removing Stopwords which refers to the most common words in a language that might not add much value to the meaning of the review. Ex("the", "is", "in", "for", "where", "when", "to", "at" etc).

```
from nltk.corpus import stopwords
nltk.download('stopwords')
stop_words = stopwords.words('english')

def stopwords_removal(review):
    new_sentence = [word for word in review.split() if word not in stop_words]
    return " ".join(new_sentence)
```

3. Word Stemming a process of removing prefixes and suffixes from words so that they are reduced to simpler forms which are called stems.It helps to reduce our vocabulary and dimensionality for NLP tasks and improves speed , efficiency in information retrieval and information processing tasks.

```
from nltk.stem.snowball import SnowballStemmer

def word_stemming(review):
    new_sentence = [stemm_words.stem(word) for word in review.split()]
    return " ".join(new_sentence)
```

4. Word lemmatization is the process through which several different forms of the same word are mapped to one single form, which we can call the root form or the base form. In more technical terms, the root form is called a lemma, this helps us ignore morphological variations on a single word.

```
from nltk.stem import WordNetLemmatizer
nltk.download('wordnet')

def words_lemmatizer(review):
    new_sentence = [lemmatizer.lemmatize(word) for word in review.split()]
    return " ".join(new_sentence)
```

All the above functions are called in a single function shown below that is applied to each and every review of a movie from the IMDb dataset.

```
def filter_text(review):
    # Parsing reviews of each movie
    .
    .
    return filter_one_text
```

5. Loading data into CSV file

```
def load_data_to_csv():
   .
   .
   .
   try:
        all_files = glob.glob(os.path.join(final_data, "IMDB_*.csv"))
        df_from_each_file = (pd.read_csv(f, sep=',') for f in all_files)
        df_merged = pd.concat(df_from_each_file, ignore_index=True)
        df_merged.to_csv("IMDB_Movie.csv")
    except:
        print("Unable to merge the CSV files")
```

4 Training

The model define is as follows

1. **Word Embedding**: They are fundamental to measure word similarity and distance at a mathematical view, and it allows us to compute relationships of their vector representation in space.

2. **LSTM**: LSTM networks are a specialized type of recurrent neural network (RNN)—a neural network architecture used for modeling sequential data and often applied to natural language processing (NLP) tasks. The advantage of LSTMs over traditional RNNs is that they retain information for long periods of time, allowing for important information learned early in the sequence to have a larger impact on model decisions made at the end of the sequence.

3. **Flatten**: This function is used to flatten the input, without affecting the batch size. A Flatten layer flattens each batch in the inputs to 1-dimension.

4. **Dense**: This function is used to create fully connected layers, in which every output depends on every input.

```
model = tf.keras.Sequential([
          tf.keras.layers.Embedding(20000, 16, input_length=max_length),
          tf.keras.layers.Bidirectional(
              tf.keras.layers.LSTM(64, return_sequences=True)),
          tf.keras.layers.Flatten(),
          tf.keras.layers.Dense(24, activation='relu'),
          tf.keras.layers.Dense(1, activation='sigmoid')
      ])
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_5 (Embedding)      (None, 120, 16)           320000
_________________________________________________________________
bidirectional_6 (Bidirection (None, 120, 128)          41472
_________________________________________________________________
flatten_3 (Flatten)          (None, 15360)             0
_________________________________________________________________
dense_11 (Dense)             (None, 24)                368664
_________________________________________________________________
dense_12 (Dense)             (None, 1)                 25
=================================================================
```

The training of the model was performed on the training data set that includes equal number of positive and negative reviews. Data was split into 80:20 for training and validation. The reviews under the test folder was never seen by the model.

5 Results

1. **Training v/s Validation Loss for 10 Epochs**

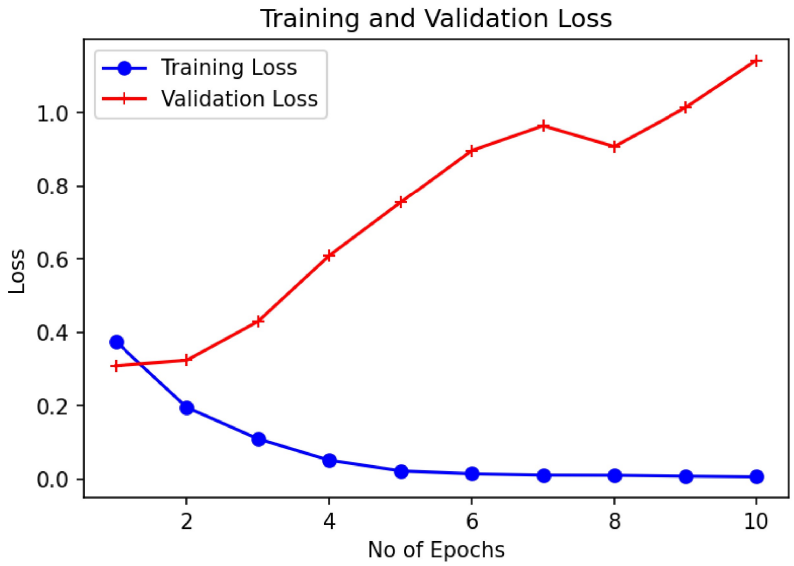

Figure 2: Training v/s Validation loss

2. **Training v/s Validation Accuracy for 10 Epochs**

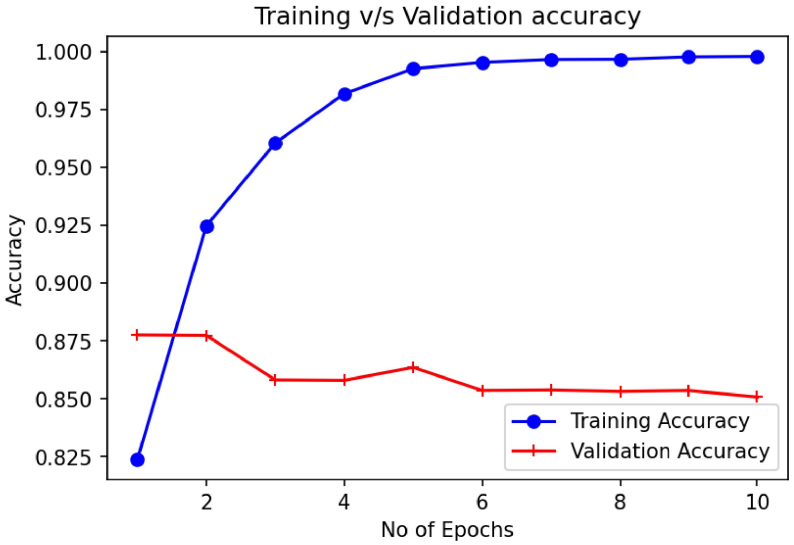

Figure 3: Training v/s Validation accuracy

Running the model on the test data was able to classify **10857** of **12500** negative reviews and **9409** of **12500** positive reviews correctly resulting in an overall accuracy of **81%** on unseen movie reviews.