Bag of Words Meets Bags of Popcorn

Sentiment Analysis of IMDB Dataset

Name: Shahriar Shayesteh
College: University Of British Columbia

Abstract

Capstone Project

Bag of Words Meets Bags of Popcorn

by Shahriar SHAYESTEH

This project first has been done as a capstone project for Udacity Mashine Learning program and Then it got expanded to what it is today…

Contents

Chapter 1

Definition

1.1 Project Overview

This problem is selected from one of the Kaggle's competitions[1]. This competition is related to sentiment analysis ;an important subfield of Natural Language Processing(NLP). Sentiment analysis is a series of methods, techniques, and tools about detecting and extracting subjective information, such as opinions and attitudes, from language. Traditionally, sentiment analysis has been about opinion polarity, i.e., whether someone has positive, neutral, or negative opinion towards something and there have been introduced many different approaches to extract meaning of words and find the semantic relationships among them in recent years. It is useful to know that the object of sentiment analysis has typically been a product or a service whose review has been made public on the Internet[2], and there have been introduced many different approaches to extract meaning of words and find the semantic relationship among them in recent years.
In this project, we focused on some of the most remarkable methods that made a breakthrough in this field in the time that they were introduced. First, we will start with Bags of Word method as the main benchmark model of the sentiment analysis and then analyze the result using Random Forest Classifier and Linear Regression Classifier. Moreover, we applied Word2vec model, introduced by a team of researchers led by Tomas Mikolov at Google. Finally, we employed Convolutional Neural Networks(CNNs) ;The model architecture is a slight variant of the CNNs architecture of Collobert et al. (2011); to compare the results with the benchmark models. In the coming sections, we go more into details and try to clarify all aspects of this problem.

1.2 Problem Statement

In this problem, we are trying to do sentiment analysis by focusing on the meaning of words as comments and reviews for different movies in IMDB website. For this project, two original benchmark models has been applied . Original benchmark models are Bags of word (BoW) features of data being classified by using a Random Forest classifier and a Linear Logistic Regression classifier. Another one employes the Word2vec feature representation of data to classify a random forest and a Linear Logistic Regression classifier. Finally, a new method is compared to our benchmark models is introduced and its performance is going to be compared to the benchmark models. Our new method employs Convolutional Neural Network to address the problem. This method, Actually, emplyes CNNs to embed words into low dimensional vectors and also uses it to classify the embedded words to some outputs.

At the end, we are willing to compare these methods and we are expecting to have a better performance on our final model (CNNs architecture).

1.3 Evaluation Metrics

Our evaluation is based on the AUC ;Area Under the Curve; value of ROC curve. The graph below shows three ROC; Receiver operating characteristic; curves representing excellent, good, and worthless tests plotted on the same graph. Accuracy is measured by the area under the ROC curve. An area of 1 represents a perfect test; an area of .5 represents a worthless test [4].

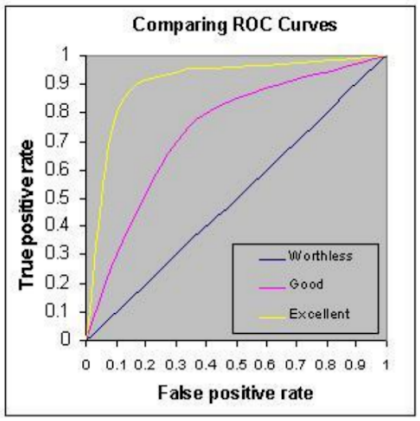

FIGURE 1.1: ROC Curves. If area under a curve is : 90-1 is a excellent (A) test, 80-.90 is a good test (B), 70-.80 is a fair test (C), .60-.70 is a poor test (D), .50-.60 is a fail test (F).

1.3.1 What Is ROC ?

ROC or Receiver Operating Characteristic Curve tells us about how good the model can distinguish between two things (e.g If a patient has a disease or no). Better models can accurately distinguish between the two. Whereas, a poor model will have difficulties in distinguishing between the two.[5]

1.3.2 Area Under the Curve

The AUC is the area under the ROC curve. This score gives us a good idea of how well the model performances. As we see in figure 1.1, the first model the AUC score is 0.9 as the area under the ROC curve is large so it does quite a good job of distinguishing the positive and negative values. Whereas, if we see the last model, we get the AUC score of 0.5. This means that the model is performing poorly and its predictions are almost random.

Chapter 2

Analysis

2.1 Data Exploration and Exploratory Visualization

The Stanford Large Movie Review (IMDB) Dataset consists of 50,000 ”highly polar”, binary labeled reviews from IMDB. These reviews are split 50:50 into training and testing sets. The distribution of labels within each subset of data is balanced. The dataset also includes a further 50,000 unlabeled reviews which may be used for unsupervised training.
Also, It is interesting to know some Statistical properties related to the length of reviews.

Percentile	10%	50%	75%	90%	Average Length	Standard Deviation
Length	521.00	991.50	1639.75	2660.29	1343.76	994.51

TABLE 2.1: Statistical properties related to the length of reviews in dataset

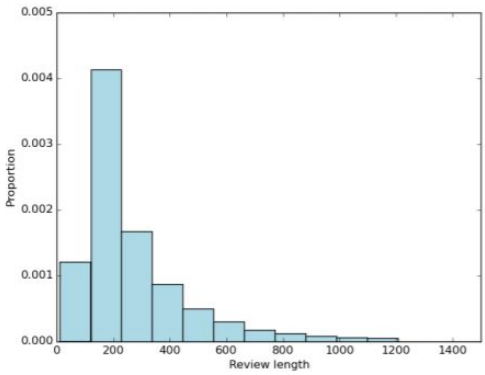

FIGURE 2.1: shows our dataset distribution over length of each comment.

After tokenizing the reviews, It is going to give us a more compact representation of reviews. We found that reviews averaged 267.9 tokens in length with a standard deviation of 198.8 tokens; the precise distribution of review lengths is shown in figure 2.1.

2.2 Algorithms and Techniques

In this work, we are trying to use a variety of algorithms to analyze the sentiment of each review. First, there are two algorithms being used to vectorize reviews; Bags of word representation, Word2vec. Moreover, two algorithms have been used for binary classification purpose; Random Forest, Linear Logistic Regression classifier. Finally, we used a Convolutional Neural Network architecture to do vectorization and also classification on reviews.

2.2.1 Features from a Bag of Words

We convert the sentences to some kind of numeric representation for machine learning. One common approach is called a Bag of Words. The Bag of Words model learns a vocabulary from all of the documents, then models each document by counting the number of times each word appears. In the IMDB data, we have a very large number of reviews, which will give us a large vocabulary. To limit the size of the feature vectors, we should choose some maximum vocabulary size. In this project, we use the 5000 most frequent words (remembering that stopwords; the words that do not convey any meanings; have already been removed). We will have 25,000 rows and 5,000 features (one for each vocabulary word)[1]. At this point, we have numeric training features from the Bag of Words and the original sentiment labels for each feature vector.

2.2.2 Features from a Word2vec

We train the Word2vec model. That is, by counting the co-occurrence of words in the same sentence, we create a vector representation of each word. We then use this matrix to train our prediction models. The training phase has several parameters that can affect the performance on the test set[1]:

1. **Architecture**:Architecture options are skip-gram (default) or continuous bag of words.
2. **Training algorithm**: Hierarchical softmax (default) or negative sampling.
3. **Word vector dimensionality**: Dimensionality of each word being represented as a feature vector in an embedding layer.
4. **Context/window size**: Window size over the center word in order to find its corresponding context words.
5. **Minimum word count**: This helps limit the size of the vocabulary to meaningful words.

2.2.3 Convolutional Neural Networks

In this model we use Convolutional Neural Network architecture; a class of neural networks; using a variation of multilayer perceptrons designed to require minimal preprocessing. These are inspired by animal visual cortex. In our proposed model, the first layers embed words into low-dimensional vectors. The next layer performs convolutions over the embedded word vectors using multiple filter sizes.

2.2.4 Random forests

Random forests or random decision forests are an ensemble learning methods for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Also, basic parameters to Random Forest Classifier can be the total number of trees to be generated and decision tree related parameters like minimum split, split criteria etc.

2.2.5 Linear Logistic Regression classifier

Logistic regression is the most famous machine learning algorithm after linear regression. In a lot of ways, linear regression and logistic regression are similar. It is another technique borrowed by machine learning from the field of statistics. Also, It is the go-to method for binary classification problems (problems with two class values).

Chapter 3

Methodology

3.1 Data Preprocessing

In this part, data preprocessing for our first benchmark model(Bags of word) is explained and in the end, the difference between the preprocessing of different methods due to similarities will be described.

1. **Removing HTML Markup**:using the BeautifulSoup Package, first, we remove the HTML tags. For doing this, calling $get - text()$ gives you the text of the review, without tags or markup.
2. **Dealing with Punctuation, Numbers**: Although punctuation and numbers may carry sentiment and should be treated as words. In our problem for simplicity, we remove them. To remove them we use built-in with Python package called re.
3. **Tokenization**: We will also convert our reviews to lower case and split them into individual words.
4. **Stop Words**: words that don't carry much meaning are called stop words. Conveniently, there are Python packages that come with stopword lists built in. we used a stop word list from the Python Natural Language Toolkit (NLTK).

   In the end, we add two more steps to our data preprocessing for making it more efficient

5. we converted the stop word list to a different data type, a set. This is for speed; since we'll be calling this function tens of thousands of times, it needs to be fast, and searching sets in Python is much faster than searching lists.
6. we joined the words back into one paragraph. This is to make the output easier to use in our Bag of Words.

Other methods: Word2vec and CNNs architecture use a similar approach for preprocessing the data but there are some small differences. First, to train Word2vec it is better not to remove stop words because the algorithm relies on the broader context of the sentence in order to produce high-quality word vectors. For this reason, we will not use stopword removal. Second, we want a specific input format. Word2vec expects single sentences, each one as a list of words. In other words, the input format is a list of lists. It is not at all straightforward how to split a paragraph into sentences. There are all kinds of gotchas in natural language. English sentences can end with "?", "!", """, or ".", among other things, and spacing and capitalization are not reliable guides either. For this reason, we'll use NLTK's Punkt tokenizer for sentence splitting.

3.2 Implementation

In this part, after preprocessing the data, we analyze implementation of our models more in details.

3.2.1 Model using Bags of word features

Now, after preparing our training reviews, we need to convert them to some numeric representation that is called feature vectors. One common approach is called a Bag of Words. The Bag of Words model learns a vocabulary from all of the documents, then models each document by counting the number of times each word appears. For example, consider the following two sentences:

Sentence 1: "The cat sat on the hat"

Sentence 2: "The dog ate the cat and the hat"

From these two sentences, our vocabulary is as follows:

$the, cat, sat, on, hat, dog, ate, and$

To get our bags of words, we count the number of times each word occurs in each sentence. In Sentence 1, "the" appears twice, and "cat", "sat", "on", and "hat" each appear once, so the feature vector for Sentence 1 is:

$the, cat, sat, on, hat, dog, ate, and$

Sentence 1: $2, 1, 1, 1, 1, 0, 0, 0$

Similarly, the features for Sentence 2 are: $3, 1, 0, 0, 1, 1, 1, 1$

In the IMDB data, we have a very large number of reviews, which will give us a large vocabulary. To limit the size of the feature vectors, we should choose some maximum vocabulary size. Below, we use the 5000 most frequent words (remembering that stop words have already been removed).

We will be using the feature-extraction module from scikit-learn to create bag-of-words features. As a result, our training data has 25,000 rows and 5,000 features (one for each vocabulary word). After converting each training review to a feature vector, we can train our supervised models (Random Forest classifier and Logistic Regression). In this classification problem, we set the number of trees for Random Forest to be the default value or 100 and also we set linear regression classifier to default value too.[1]

3.2.2 Model using Word2vec features

This method represents words as high dimensional vectors so that words that are semantically similar will have similar vectors. It comes in two flavors: Continuous Bag of Words (CBOW) and Skip-Gram. CBOW trains a network to predict a word from its context, while Skip-Gram does the exact opposite, predicting the context based on a specific target word.

1. **Architecture**: Architecture options are skip-gram (default) or continuous bag of words. We found that skip-gram was very slightly slower but produced better results.

2. **Training algorithm**: Hierarchical softmax (default) or negative sampling. For us, the default worked well. Downsampling of frequent words: The Google documentation recommends values between .00001 and .001. For us, values closer 0.001 seemed to improve the accuracy of the final model.

3. **Word vector dimensionality**: More features result in longer runtimes, and often, but not always, result in better models. Reasonable values can be in the tens to hundreds; we used 300.

4. **Context/window size**: How many words of context should the training algorithm take into account? 10 seems to work well for hierarchical softmax (more is better, up to a point).

5. **Minimum word count**: This helps limit the size of the vocabulary to meaningful words. Any word that does not occur at least this many times across all documents are ignored. Reasonable values could be between 10 and 100. In this case, since each movie occurs 30 times, we set the minimum word count to 40, to avoid attaching too much importance to individual movie titles. This resulted in an overall vocabulary size of around 15,000 words. Higher values also help limit run time.

Finally, after training our unsupervised model to extract Word2vec features of the training Reviews. We need to train two introduced classifiers to classify reviews based on their Word2vec features.[1]

3.2.3 Convolutional Neural Network model

The network we will build in this post looks roughly as follows:

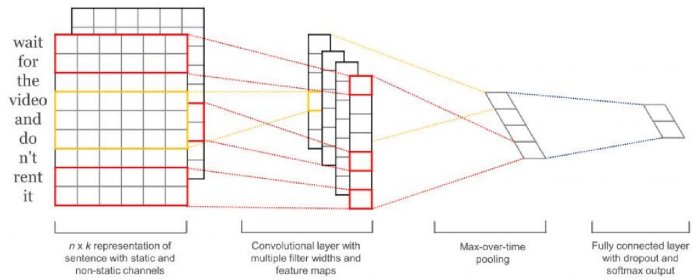

FIGURE 3.1: CNNs architecture for this task. [3]

The first layers embeds words into low-dimensional vectors. The next layer performs convolutions over the embedded word vectors using multiple filter sizes. For example, sliding over 3, 4 or 5 words at a time. Next, we max-pool the result of the convolutional layer into a long feature vector, add dropout regularization, and classify the result using a softmax layer.

- Pixels are made of embedding vectors of each word in a sentence .
- Convolutions are performed based on word-level
- Classify each sentence as positive (1) or negative (0)

So now we will see the implementation part. I will implement this using this method:
CNNs using 1D convolution and pooling with defined embedding layer[3].

Chapter 4

Refinement

As it is mentioned before, this project has one main benchmark model and one other benchmark models, comparing with CNNs architecture model. For each model, cross validation method has been run to tune the hyper parameters of models. In this part we go into details of tuning hyper-parameters and comparing the change in the score and accuracy of each model.

4.1 Features from a Bag of Words

In this model, we used two classifiers first Random Forest and second Linear Logistic Regression and Benchmark model uses Random Forest with 100 number of estimators and random number of trees and it gives us accuracy of 0.84. Then we train it two times using cross validation method to tune the Random Forest parameters. First time to tune the number of estimators(random trees) and second time to tune the depth of the tree.

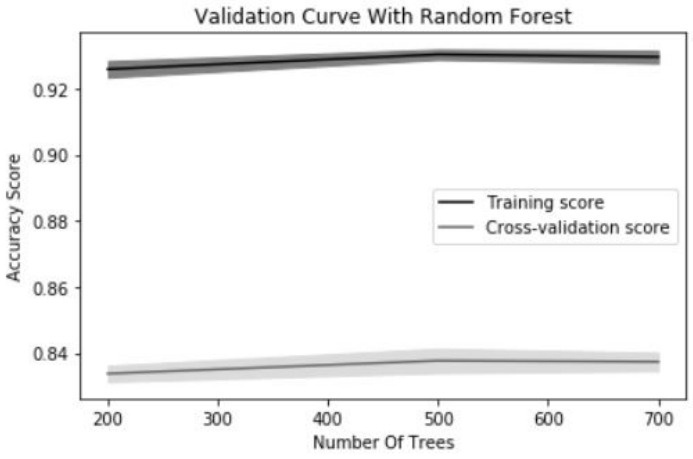

FIGURE 4.1: Validation and Training accuracy scores of Random Forest respect to Number of trees.

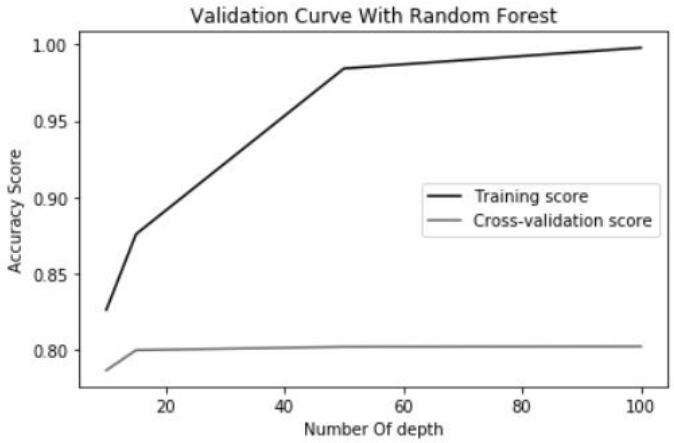

FIGURE 4.2: Validation and Training accuracy scores of Random Forest respect to Depth of trees(between 20 to 100).

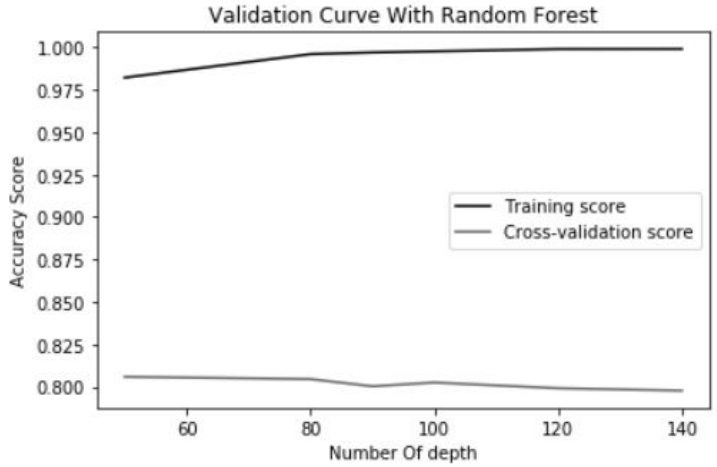

FIGURE 4.3: Validation and Training accuracy scores of Random Forest respect to Depth of trees(between 60 to 140).

It can be seen in the Figure 6 that increasing number of trees to 500, increase validation accuracy slightly and after that it is flatten out. So for Random Forest model we choose the number of estimators to be 500. By looking at Figure 7 and Figure 8, we can see validation accuracy peak at 80 so we choose depth of each tree to be 80. The result of this change will be shown in the conclusion part.

Doing the same procedure for Logistic Regression to tune C(Inverse of regularization strength) One time using L2 regularization as default setting .For that we got the same response.

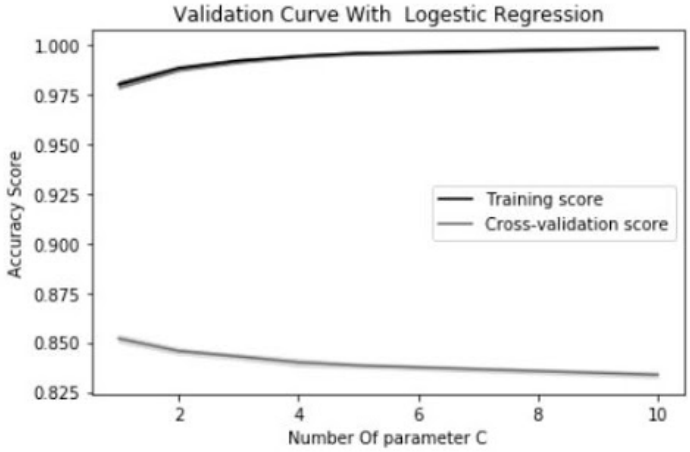

FIGURE 4.4: Validation and Training accuracy scores of Logistic Regression respect to C(between 1 to 10).

It turns out $C = 1$ that is default value is the best choice for this classifier. Then classifier is trained using $C = 1$ and L-2 regularization and $C = 1$ L-1 regularization and the result will be discussed in the next part.

4.2 Features from a Word2vec

In this model, again everything is the same as the previous part. we used two classifiers first Random Forest and second Logistic Regression and Benchmark model uses Random Forest with 100 number of estimators and random number of trees and it gives us accuracy of 0.83 for Random Forest and 0.84 for Logistic Regression . Then we train it two times using cross validation method to tune the Random Forest parameters. First time to tune the number of estimators(random trees) and second time to tune the depth of the tree.

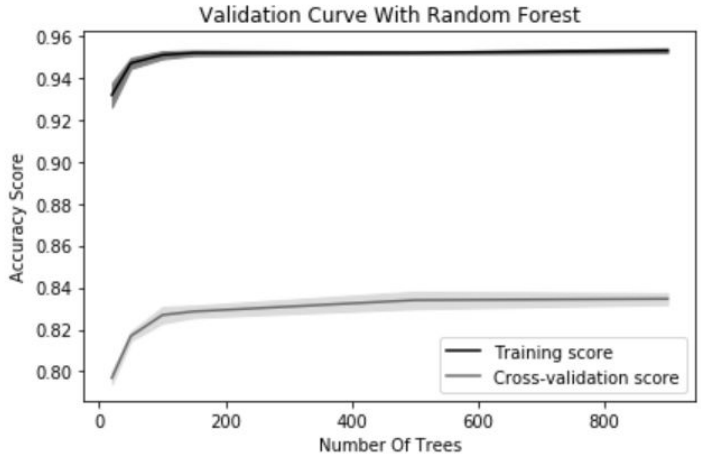

FIGURE 4.5: Validation and Training accuracy scores of Random Forest respect to Number of trees .

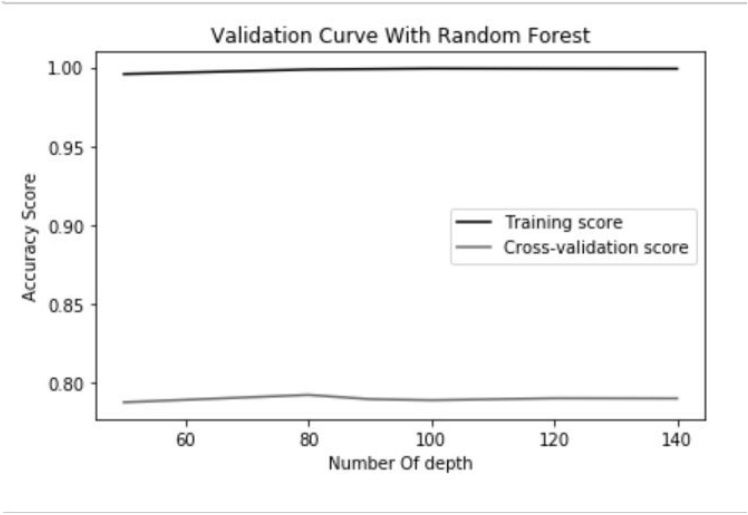

FIGURE 4.6: Validation and Training accuracy scores of Random Forest respect to Depth of trees(between 20 to 100).

It can be seen in Figure 10 that increasing number of trees to 700, increase validation accuracy slightly and after that it is flatten out. So for Random Forest model we choose the number of estimators to be 500. By looking at Figure 7 and Figure 8, we can see validation accuracy peak at 80 so we choose depth of each tree to be 80. The result of this change will be shown in the conclusion part.

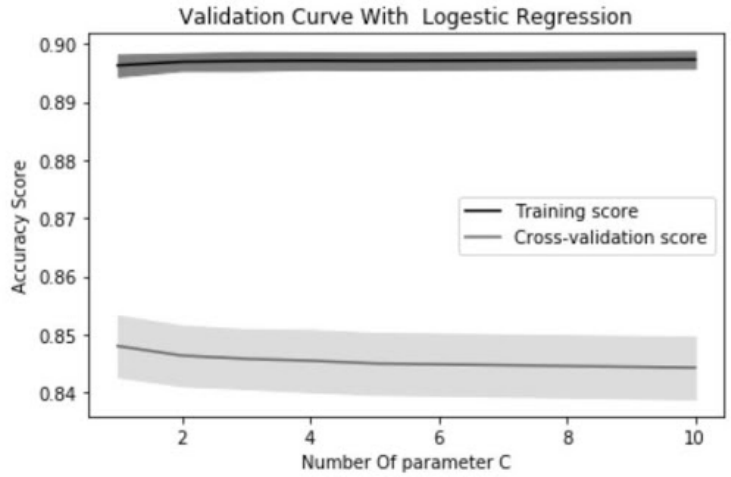

FIGURE 4.7: Validation and Training accuracy scores of Logistic Regression respect to C(between 1 to 10).

It turns out C= 1 that is default value is the best choice for this classifier. Then classifier is trained using C=1 and L2 regularization and C= 1 L1 regularization and it turns out that L1 regularization due to robustness to outliers has a better accuracy than L2 regularization for this model.

4.3 CNNs Architecture

For this method, we used the introduced method in the previous part and to tune the hyper-parameters of our CNNs based architecture, we used GridSearchCV method in scikit-learn to improve the accuracy of our models by tuning our type of optimizer and number of epochs and batches . After analysis turns out that the best model for our architecture should use ‘adam ’ optimization method for loss with batch size = 300 and epochs = 15 to get the AUC = 0.93.

Chapter 5

Results

Our evaluation is based on the AUC and ROC values. For BoW benchmark model. We limited our vocabulary to 5000 and used 500 trees in Random Forest, and obtained a baseline AUC of 0.85 and using Logistic Regression classifier of parameter C = 1 then we tried to tune the hyper- parameters of the classifiers even more to improve the classification accuracy if it is possible.The result of this represented in Table 5.1.

For Word2vec classifier, we tried to tune hyper-parameters of Word2vec method based on two hyper parameters. First, number of features.Second, number of context words. We analyze the changes which these two hyper-parameters make in ROC in Table 5.1 and we show the result of changing this parameters on AUC curve in figure 5.1 and 5.2.

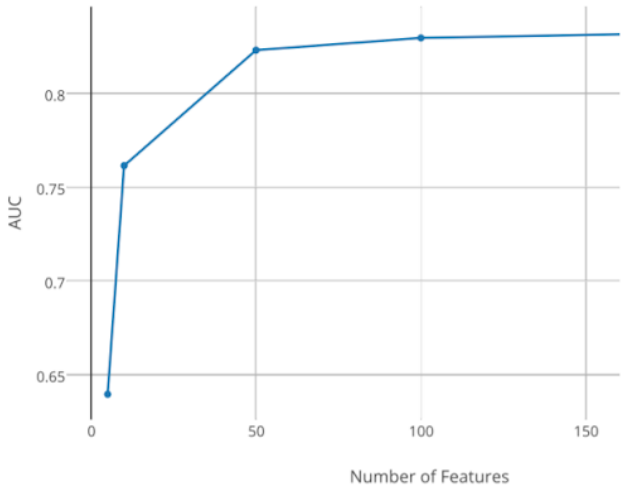

FIGURE 5.1: Area under curve (AUC) vs number of features[1].

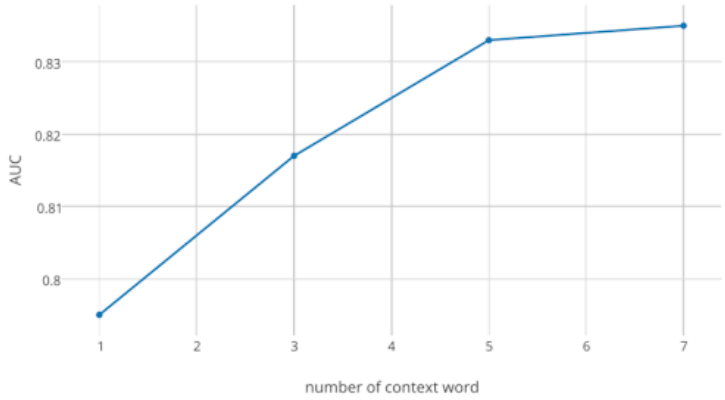

FIGURE 5.2: AUC vs number of context words[1].

Finally, The introduction of CNNs architecture causes a significant increase in the AUC, compared to our benchmark models. We described the architecture of this model in details in previous parts. The result for baseline model and after justification is shown in Table 5.2.

Models	Baseline AUC of Random Forest	Baseline AUC of Logistic Regression	Justified AUC of Random Forest	Justified AUC of Logistic Regression
BOW	0.84	0.85	0.85	0.85
Word2vec	0.83	0.86	0.85	0.86

TABLE 5.1

Model	BaseLine AUC	Justified AUC
CNNs	0.93	0.93

TABLE 5.2

Chapter 6

Conclusion and Improvement

6.1 Free-Form Visualization

In this chapter, the ROC curve of each sentiment analysis methods using Random Forest classifier and Logistic Regression classifier and also the ROC curve of CNN's based archeture is represented. The disscusion about how the hyper parameters have been chosen and also the performance of each method has been done in the previous chapter.

6.1.1 BOW

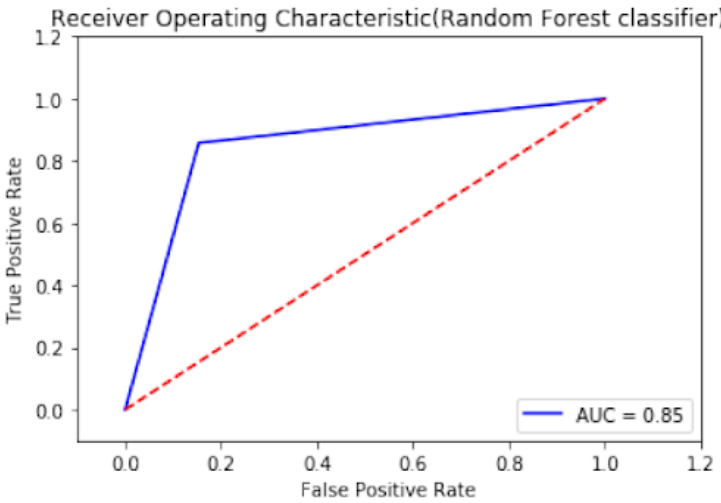

FIGURE 6.1: ROC curve For Random Forest Classifier (n-estimators = 500,max-depth = 60).

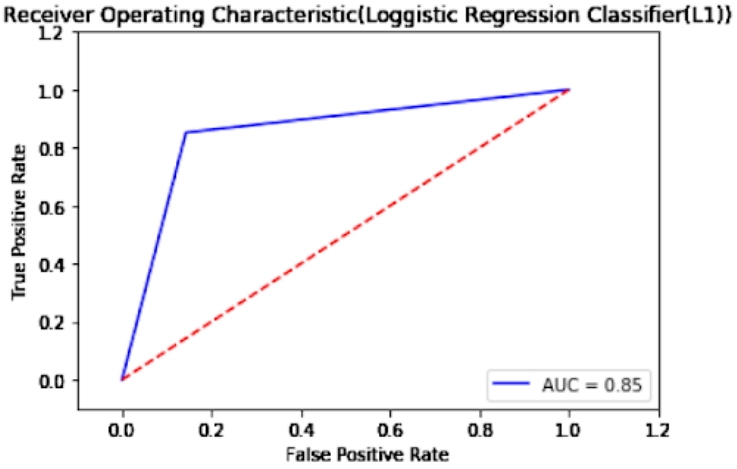

FIGURE 6.2: ROC curve For Logistic Regression(penalty='$l1$',C=1.0).

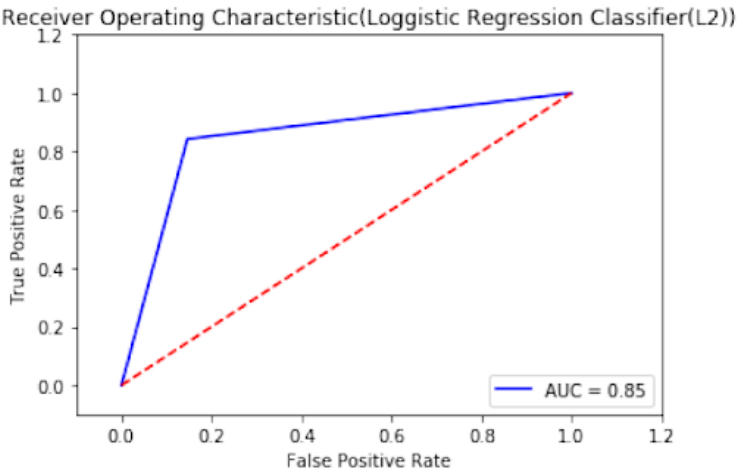

FIGURE 6.3: ROC curve For Logistic Regression(penalty='$l2$',C=1.0).

6.1.2 Word2Vec

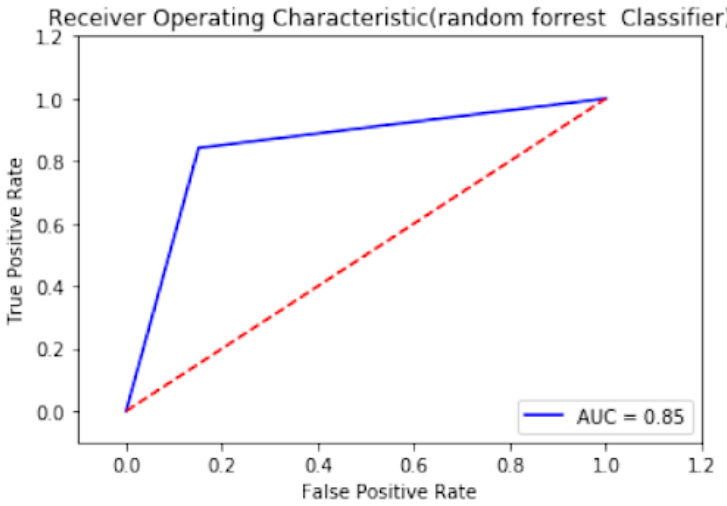

FIGURE 6.4: ROC curve For Random Forest Classifier (n-estimators = 700,max-depth=80).

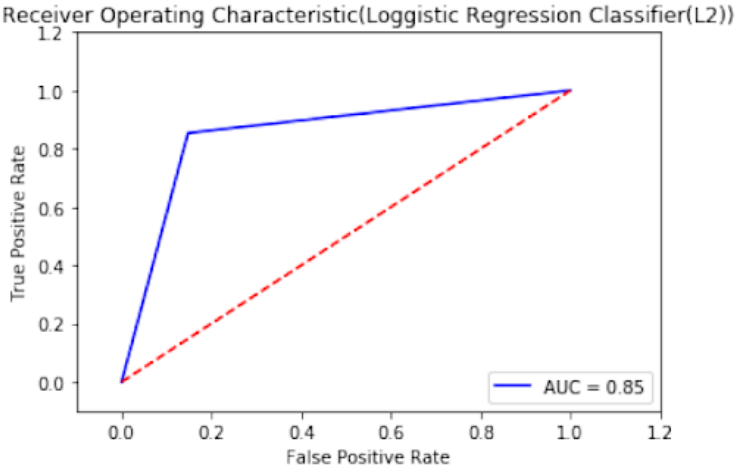

FIGURE 6.5: ROC curve For Logistic Regression(penalty='l2',C=1.0).

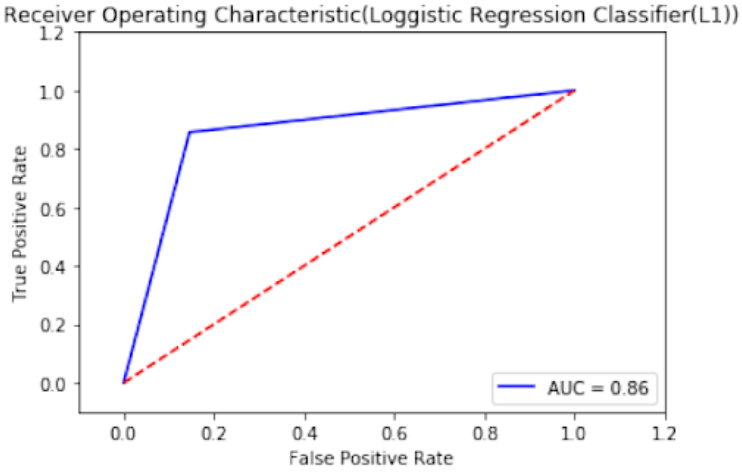

FIGURE 6.6: ROC curve For Logistic Regression(penalty='l1',C=1.0).

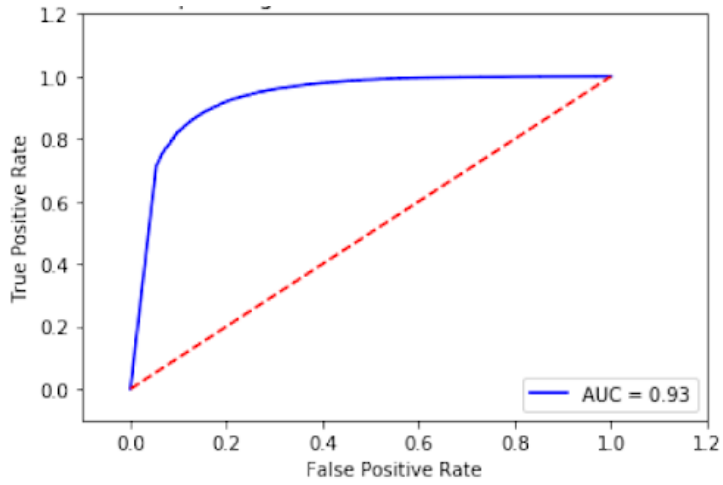

FIGURE 6.7: ROC curve For CNNs architecture.

6.1.3 Reflection

In this project, we have try to analyze different common sentiment analysis methods using IMDB dataset. In other words, In this project, we experience introduced methods by Kaggle for Bag of Words meets Bags of Popcorn competition. The benchmark models which are Introduced in this competition is using Bag of Words(BOW) method and also Word2vec. After presenting these methods, we are encouraged to use Deep Learning based architecture and we decided to use CNNs based architecture. After analysing the result of these methods, We observed that the Word2vec model slightly outperforms the vanilla bag-of-words model. Also, the result shows that using deep learning methods like CNNs architecture can increase the accuracy significantly.

6.1.4 Improvement

We have a few recommendations for future work. We removed non-alphabet characters in our analysis, but it may be useful to also take those into account. For instance,

emojis and a repetition of punctuation (e.g., multiple exclamation marks) may infer the sentiment of the review significantly. Other classification methods may also be useful, such as regression and k-nearest neighbor. However, due to significant performance of deep learning methods and reinforcement learning algorithms on NLP tasks especially LSTM architecture based models employing Deep learning methods can significantly revolutionised solving sentiment analysis problems.

Chapter 7

Reference

[1]: Bag of Words Meets Bags of Popcorn, Amir Sadeghian, Ali Reza Sharafat

[2]: Learning Word Vectors for Sentiment Analysis, Andrew L. Maas, Raymond E. Daly, Peter T.Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts

[3]:https://towardsdatascience.com/sentence-classification-using-cnn-with-deep-learningstudio-fe54eb53e24

[4]: Interpreting Diagnostic Tests,Thomas G. Tape, MD, University of Nebraska Medical Center

[5]: https://medium.com/greyatom/lets-learn-about-auc-roc-curve-4a94b4d88152