Jahnavi Mesa Mohammed Ather Ahmed Shareef

**Quora Question Pairs**

[jxm169230@utdallas.edu](mailto:jxm169230@utdallas.edu) [mxs166331@utdallas.edu](mailto:mxs166331@utdallas.edu)

**Abstract**

Quora [1] is a platform where people post questions which are answered by the Quora community (people who can share their insights with experiences). To make this process effective, a suggested approach was to limit the users for posting questions which are like the existing questions – referred as duplicate questions in this report, thereby helping users to find the answers to their questions over the same page instead of referring multiple pages.

1. **Introduction**

Considering the growing world of dialects, it is critical to understand the actual meaning of the sentences to compare them. This project targets to understand the semantic similarity between sentences and effectively use them for solving our duplicate questions problem [2]. We have used word embedding techniques which works with the help of spaCy [3] which works with underlying concepts of tokenization, lemmatization, speech tagging and sentence recognition.

In the Quora dataset, the positive questions would be the associated questions which should be having the same answer. Alternatively, negative questions should not be having same answers tagged to them.

1. **Problem Definition**

Given two questions (sentences) q1 and q2, we aim to classify it as either 0 or 1 i.e. find the similarity/dissimilarity between q1 and q2 which is represented as

Where 0 represents the questions, which are not similar and 1 represents them to be similar or to have same answers in context of the question. This is an interesting problem, as it’s not just a simple comparison of questions which can be achieved by tokenizing and processing the words of the questions, conversely it requires semantic matching of words understanding the dialect and usage of words (as same words used in another context means a different meaning in English.)

The goal of this project is to use Machine learning/NLP techniques to predict whether two given questions have the same meaning.

1. **Background/Related Work**

There is a similar problem where the category of given words is identified and based on document of words they are assigned to a class, for ex: document classifier, where we strip the sentences and remove stop-words and non-meaningful characters. Later, using lemmatizing and stemming techniques to identify the similarity of sentences.

Above problem can be solved using Naïve Bayes classifier. The same solution cannot be used with our duplicate question problem as there can be two sentences which vary just by a single word and we cannot figure out the semantical meaning of the sentence.

1. **Dataset and Features**

Quora has released its first dataset [4] which is also available at Kaggle in order to contest the built model based on the data. Data contains 6 columns and 404290 rows, where the below figure 1 depicts the first 5 rows. *question1* and *question2* are checked for duplicacy and *is\_duplicate* is the target.

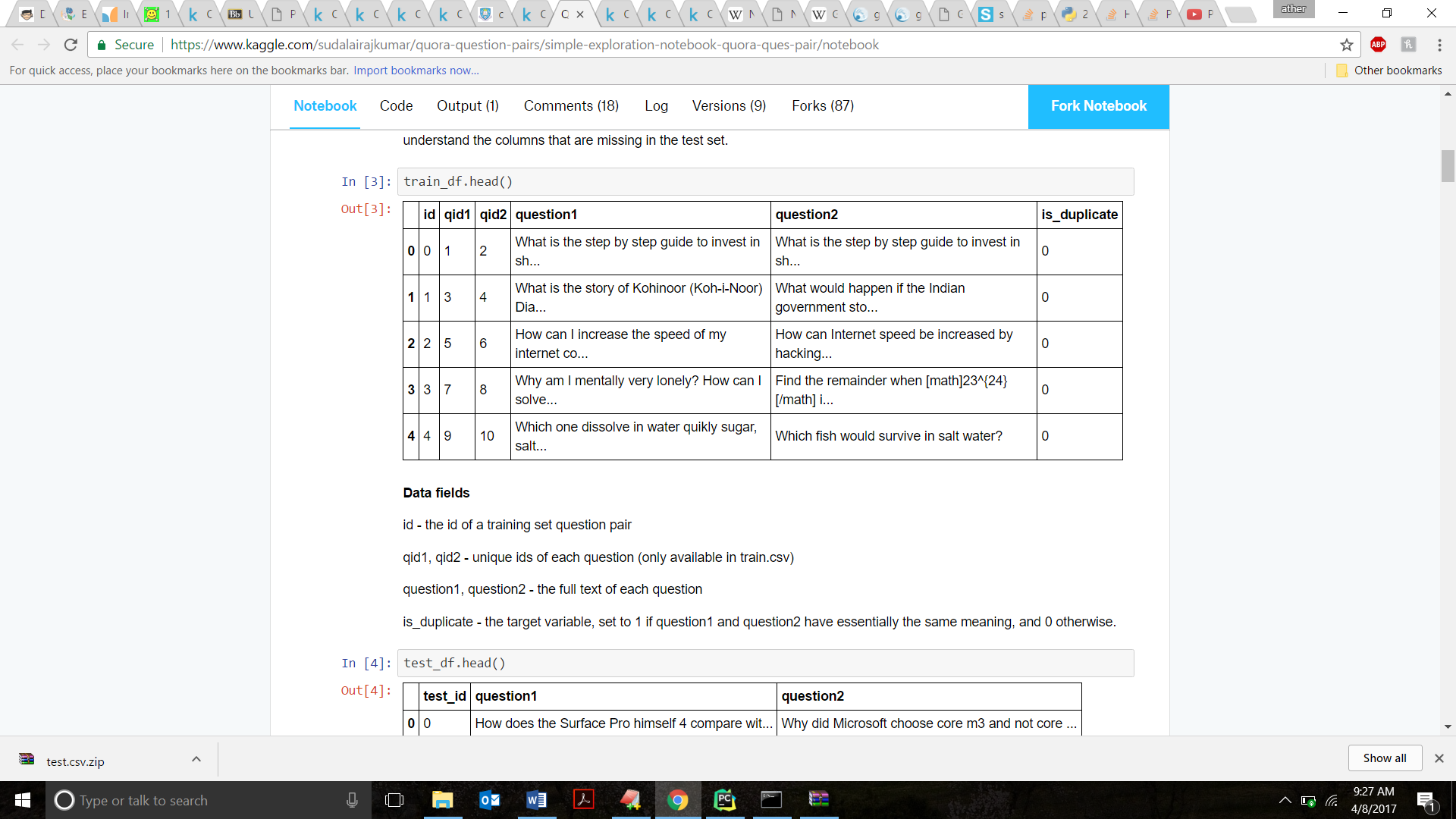


Figure 1 Train Data: 404290 rows and 6 columns

Due to the hardware limitations, we have used first 100,000 rows for training. Below is the plot which depicts the frequency of number of words on the dataset.

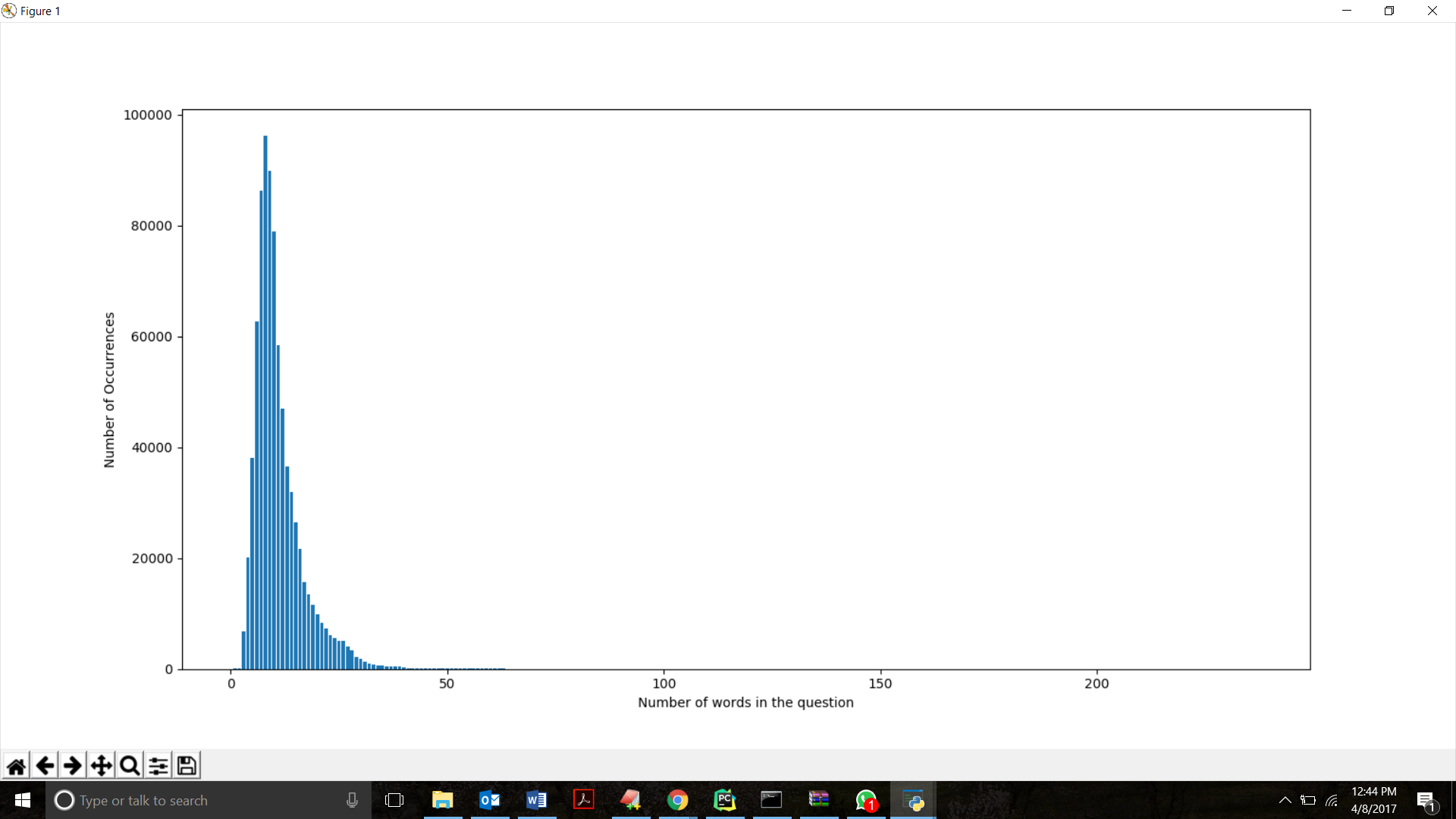


Figure 2 Bar plot of Number of words in the Question

1. **Pre-processing techniques**

Beginning with our data, it contains words or questions which are processed by spaCy’s which tokenize, lemmatize and remove the un meaningful words while constructing the 300-dimensional representation of the question. In our project, we have used scikit-learn’s StandardScaler to transform the spaCy’s 300-dimensional vector values to values with mean 0 and standard deviation of 1 i.e. the data will be distributed normally using the below equation.

1. **Proposed Solution**
   1. **Word Embedding with spaCy:**

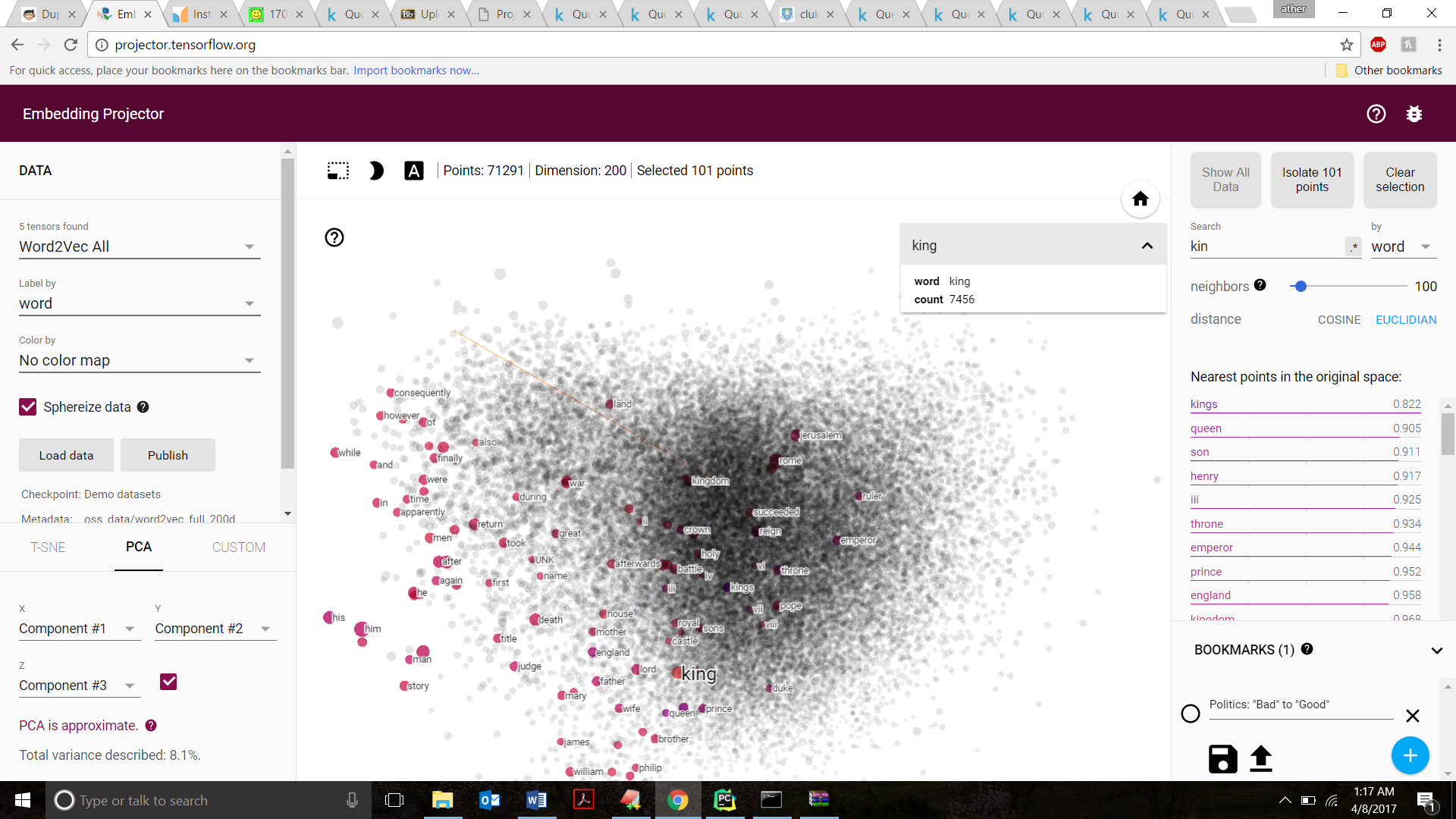
Word embedding is a set of techniques where word or sentences of English are mapped to real values vectors. There are several ways we can obtain the vectorization task for ex: Gensim, GLoVe, spaCy. We have used spaCy over gensim as we gensim works with word2vec and it needs huge training data to perform well and achieve the required accuracy.

On a contrary, spaCy is a pre- trained model which is trained on Wikipedia content. Thus, almost all the words used in questions by the users are present in spaCy which simplifies the issue of handling the missing words representation in spaCy.

* 1. **Proposed Model:**

Based on the experiments (which is explained in detail in next section), we decided to go with spaCy with deep layer network to get the required accuracy. We achieved an accuracy of 71% with 10k cross validation.

As explained in 6.1, we used spaCy to vectorize (300 dimension) the words of the sentence and combine them to represent a sentence. For example, using Word2Vec, below is the Embedding projector [5] view of king search. It can be observed that the cosine/Euclidian distance similar words to king as very less.



Based on the above concept, we are finding a vector representation of a sentence. The flow of data is depicted in the graphical model below.

Questions 1 and Question 2 are passed to spaCy which tokenizes and process the questions and creates the learning vectors of 300 dimension each of a question. Features like length of question 1 and 2, sum of lengths, difference in question numeric vectors and sum of question numeric vectors are extracted from the spaCy layer. This data is combined using pandas dataframe and pre-processed using scikit learn.

Embedding using spaCy for Q2

Embedding using spaCy for Q1

Feature Extraction Q2

Feature Extraction Q1

Pre-processing and combining features to Feed to Deep Layers

Figure 3 Graphical Representation of Proposed Model

Multiple Deep layers

Deep Network Input Layer

Deep Net Output Layer

Pre-processed data is then fed to deep layers i.e. deep neural network. In our project, we have used 5 deep layers each 11, 15, 29, 30, 21 neurons each and ‘relu’ activation function to avoid diminishing deltas. We used scikit learn to implement the deep network. The model is trained using processed train data and validated using 10 k cross validation over 100k rows. We have achieved an average of 71% accuracy using the above model.

* 1. **Results:**

\* Confusion Matrix \*

[[49183 13563]

[15425 21829]]

\* Classification Report \*

precision recall f1-score support

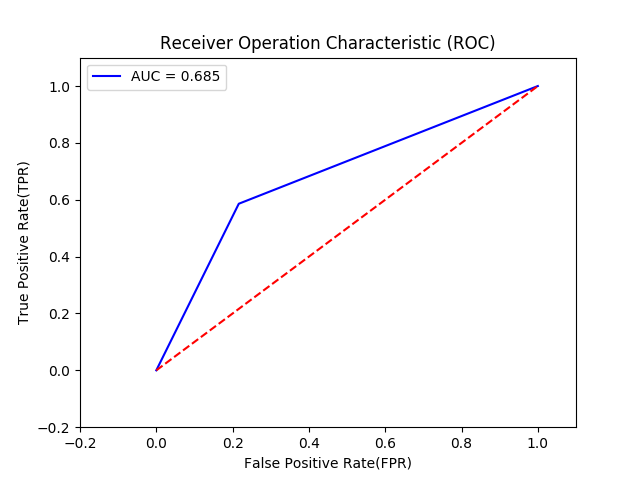
0 0.76 0.78 0.77 62746

1 0.62 0.59 0.60 37254

avg / total 0.71 0.71 0.71 100000

Accuracy of the Model is: 0.71012

Area under the ROC curve of the Model is: 0.68489664836



1. **Experimental Evaluation Methodology**
   1. **Word2vec and Random Forest Tree:**

Word2vec [6] available in Gensim library is an algorithm that embeds words into a 300-dimensional vector space. A vocabulary is constructed from the training data and then represented into vector and similar words will be grouped together in the vector space. These word vectors for each question are used as features to build the required model.

The problem with this approach is that we have very less attributes to train the model and end up with low accuracy.

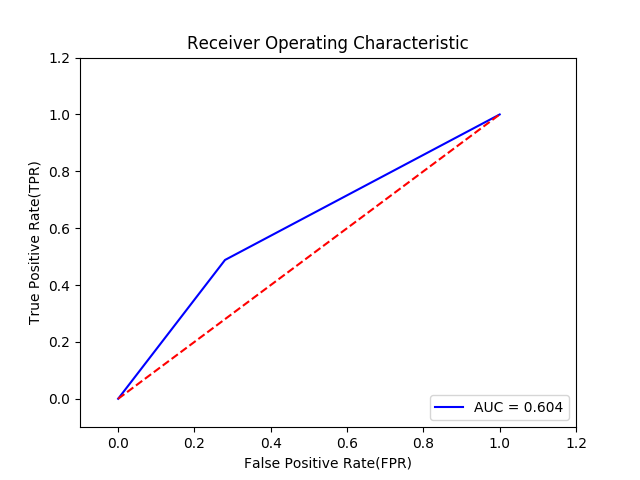
Word2vec used with combination of Random Forest Tree Results:

Accuracy: 63.4666666667

precision recall f1-score support

0 0.71 0.72 0.71 1896

1 0.50 0.49 0.50 1104

avg / total 0.63 0.63 0.63 3000 

* 1. **Word2vec with Deep Learning**

We tried using Deep learning in place of Random forest trees to check if the performance improves by tuning the parameters of the deep learning network. The activation function used for the network is ‘relu’.

MLPClassifier from scikit library has been used with adam as the solver for weight optimization. The accuracy and Roc improved when compared to Random Forest tree model but still not up to the mark.

Word2vec with Deep Learning results:

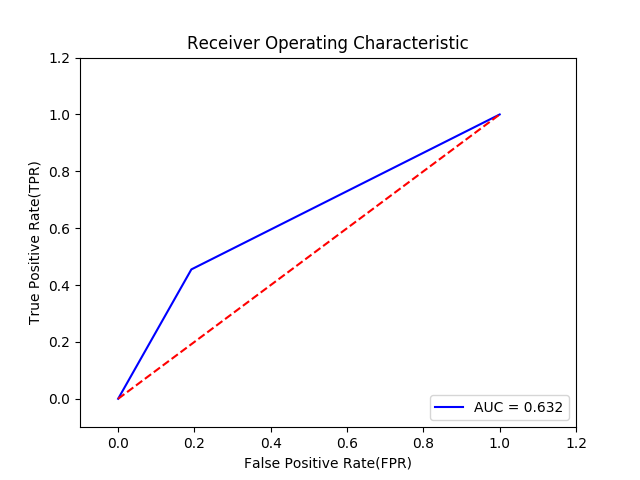
Accuracy: 0.676933333333

precision recall f1-score support

0 0.71 0.81 0.76 18854

1 0.58 0.46 0.51 11146

avg / total 0.67 0.68 0.67 30000



1. **Future Work**

Combination of different models can be used to concentrate on few specific scenarios in the data which were misclassified by our model. Performance can also be improved by using Spacy with TD-IDF to convert a sentence into a weighted average of the word2vec vectors.

The concept of Siamese network can also be used to bring closer the similar sentences and disperse any dissimilar sentences in the space, which requires more time and understanding of Siamese network implementation as we are having a problem where there is no developed network.

1. **Conclusion**

Deep learning technique was used to classify the question pairs with combination of Spacy which yielded better results when compared to other word embedding techniques. The best accuracy for this natural language processing problem with our model is 71.04%. The data set provided for this problem has been labeled by human experts. Thus, the true meaning and the similarity between the sentences cannot be predicted with certainty.

1. **Bibliography**
2. Quora: <https://www.quora.com/>
3. Problem: <https://www.kaggle.com/c/quora-question-pairs>
4. Spacy: <https://spacy.io/>
5. Data: <https://www.kaggle.com/c/quora-question-pairs/data>
6. Tensor Flow Projector: <http://projector.tensorflow.org/>
7. Word2vec: <https://en.wikipedia.org/wiki/Word2vec>
8. Kaggle Discussion: <https://www.kaggle.com/c/quora-question-pairs/discussion/30340>
9. Semantic similarity paper: <https://aclweb.org/anthology/K15-1013>
10. Word embedding: <https://en.wikipedia.org/wiki/Word_embedding>
11. Plots: <http://matplotlib.org/examples/pylab_examples/barchart_demo.html>
12. Genism: <https://radimrehurek.com/gensim/models/word2vec.html>
13. Blog: <https://explosion.ai/blog/quora-deep-text-pair-classification>
14. Bilateral Multi-Perspective Matching for Natural Language Sentences <https://arxiv.org/pdf/1702.03814.pdf>
15. spaCy tutorial: <https://www.analyticsvidhya.com/blog/2017/04/natural-language-processing-made-easy-using-spacy-%E2%80%8Bin-python/>
16. More about spaCy: <https://explosion.ai/blog/how-spacy-works>
17. Use of spaCy: <https://nicschrading.com/project/Intro-to-NLP-with-spaCy/>
18. Deep learning with Questions: <https://engineering.quora.com/Semantic-Question-Matching-with-Deep-Learning>