Quora Question Pairs
A Novel Approach for Question Pair Similarity Identification

A project report submitted for the partial fulfillment of the
Bachelor of Technology Degree
in
Computer Science & Engineering
under
Maulana Abul Kalam Azad University of Technology
by

Sudipta Halder
Roll No: 10400116059, Registration Number: 161040110188
Smita Bandyopadhyay
Roll No: 10400116078, Registration Number: 161040110169

Academic Session: 2016-2020

Under the Supervision of
Prof. Amit Kumar Das

Department of Computer Science and Engineering
Institute of Engineering & Management
Y-12, Salt Lake, Sector 5, Kolkata, Pin 700091, West Bengal, India Affiliated

To

Maulana Abul Kalam Azad University of Technology
BF 142, BF Block, Sector 1, Kolkata, West Bengal 700064

May 2020

IEM
INSTITUTE OF ENGINEERING & MANAGEMENT
श्रद्धावान लभते ज्ञानम्
Salt Lake Electronics Complex, Kolkata - 700091, WB, INDIA

Phone : (033) 2357 2969/2059/2995
(033) 2357 8189/8908/5389
Fax : 91 33 2357 8302
E-mail : director@iemcal.com
Website : www.iemcal.com

CERTIFICATE

TO WHOM IT MAY CONCERN

This is to certify that the project report titled **"Quora Question Pairs A Novel Approach for Question Pair Similarity Identification"**, submitted by **Sudipta Halder, Roll No: 10400116059, Registration Number: 161040110188**, **Smita Bandyopadhyay, Roll No: 10400116078, Registration Number: 161040110169**, students of **Institute of Engineering & Management** in partial fulfillment of requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering**, is a bona fide work carried out under the supervision of **Prof. Amit Kumar Das** during the final year of the academic session of 2016-2020. The content of this report has not been submitted to any other university or institute for the award of any other degree. It is further certified that the work is entirely original and performance has been found to be satisfactory.

____

Prof. Amit Kumar Das
Assistant Professor
Department of Computer Science and Engineering
Institute of Engineering & Management

____

Prof. Sourav Saha
H.O.D.
Department of Computer Science and Engineering
Institute of Engineering & Management

____

Prof. (Dr.) Amlan Kusum Nayak
Principal
Institute of Engineering & Management

A Novel Approach for Question Pair Similarity Identification

Gurukul Campus: Y-12, Salt Lake Electronics Complex, Sector-V, Kolkata 700091, Phone: (033) 2357 2969
Management House: D-1, Salt Lake Electronics Complex, Sector-V, Kolkata 700091, Phone: (033) 2357 8908
Ashram Building: GN-34/2, Salt Lake Electronics Complex, Sector-V, Kolkata 700091, Phone: (033) 2357 2059/2995

INSTITUTE OF ENGINEERING & MANAGEMENT

IEM
श्रद्धावान लभते ज्ञानम्
INSTITUTE OF ENGINEERING & MANAGEMENT
Good Education, Good Jobs

DECLARATION
FOR NON-COMMITMENT OF PLAGIARISM

We, **Sudipta Halder**, **Smita Bandyopadhyay**, students of B.Tech. in the Department of Computer Science and Engineering, Institute of Engineering & Management have submitted the project report in partial fulfillment of the requirements to obtain the above noted degree. We declare that we have not committed plagiarism in any form or violated copyright while writing the report and have acknowledged the sources and/or the credit of other authors wherever applicable. If subsequently it is found that we have committed plagiarism or violated copyright, then the authority has full right to cancel/reject/revoke our degree.

Name of the Student: SUDIPTA HALDER

Full Signature: ________________________________

Name of the Student: SMITA BANDYOPADHYAY

Full Signature: ________________________________

Date: ________________________________

Abstract

Quora is a social media website where questions are asked, answered, edited and organized by its community of users and everyday millions of people visit Quora. Users can contribute by editing questions, answering questions that have been posted by others. This whole question-answer contribution is displayed as a thread on a single question with a list of semantically related questions so that users do not have to answer questions which are similar to it or duplicate. Quora wanted to improve their similarity detection system. So, they released their dataset publicly and later launched a Kaggle competition in 2017. At that time, Quora used a Random Forest model to identify duplicate questions. This model does not work very efficiently with large amount of data [27]. They wanted Kagglers to apply advanced techniques to improve the similarity detection system. The prize money of the competition was $25000.[26] The main aim of this work is to apply various Natural Language Processing (NLP) techniques for feature engineering from the given dataset and apply and compare some machine learning models such as Logistic Regression with Stochastic Gradient Descent, Linear Support Vector Machine with Stochastic Gradient Descent, Decision Tree, Random Forest, Gradient Boost Decision Tree, Extra Trees, Adaptive Boosting, Stacking Classifier to predict the similarity.

Acknowledgements

We must not forget to acknowledge everyone who has provided constant support to us during our B.Tech course. First and foremost, we would like to express sincere gratitude to our supervisor **Prof. Amit Kumar Das** for his continuous support and motivation in fueling the pursuance of carrying out this project endeavor. Without his guidance and persistent encouragement, this project work would not have been possible. He has been a tremendous mentor for us throughout this academic journey. Many of his academic advises about our career growth have been priceless.

We would like to convey sincere gratitude to **Prof. Sourav Saha** for providing us constant inspiration to stand firm against several setbacks throughout the course. Additionally, we would like to thank all the technical, non-technical and office staffs of our department for extending facilitating cooperation wherever required. We also express gratitude to all of our friends in the department for providing the friendly environment to work on the project work.

We would also like to thank our Director **Prof. Satyajit Chakraborti** for providing us an outstanding platform in order to develop our academic career. In addition, we also preserve a very special thankful feeling about our Principal **Prof. Amlan Kusum Nayak** for being a constant source of inspiration.

A special thank is due to our family. Words cannot express how grateful we are to our parents for all the sacrifices that they have made while giving us necessary strength to stand on our own feet.

Finally, we would like to thank everybody who has provided assistance, in whatever little form, towards successful realization of this project but with an apology that we could not mention everybody's name individually.

Contents

List of Figures

List of Tables

1 Introduction

Quora is a very popular website for Question Answer forum among users and a large number of people from around the globe visit it every day. It is a platform for asking questions and answering different questions of other people. As of 16 November 2018, **almost 38 MILLION** questions have been asked on Quora.[25] Since the number of questions is too high, there are always such questions which are semantically similar to other questions and hence redundant. For example, questions like “How do I read and find my YouTube comments?” and “How can I see all my YouTube comments?" are identical because both have the same meaning. Some questions, like “How old are you?” and “What is your age?” do not share the same words but they are of same meaning semantically. Therefore, such questions are also considered duplicate. These redundant questions do create problems in several ways. One is it unnecessarily creates repetition and hence takes more storage than required in database servers. Another problem is user have to answer duplicate questions which is both waste of time and energy. It would be highly beneficial if the redundant or duplicate question can be reduced. This will help users to get correct and crisp information in very short amount of time. Machine Learning techniques can be used for tackling this problem. We aim on deploying some techniques that would help to judge the similarity between two questions in a more meaningful sense. Also, we then aim to decide the similarity between a pair of question using various machine learning algorithm and compare the efficiency of different algorithm in tackling the problem.
Quora Question Pairs Database source: https://www.kaggle.com/c/quora-question-pairs/data

2
Literature Survey

2.1 Related Works

How can a question be a duplicate of another question or semantically same? Classifying question pairs as duplicate or non-duplicate is a tedious task since it is difficult for machine to interpret the true underlined meaning of two question pairs. In this dataset, labels have been done by domain experts of this area. But it is also tough for human experts to find out the underlying meaning of each and every pair because there is always a chance of ambiguity in our language. We may not be able to identify the mood of the user's question every time correctly. Hence, this dataset should not be considered as 100% accurate, it may contain improper labeling and outliers.
There have been many contributions of researchers in this field such as Gabrilovich and Markovitch, 2007; Mihalcea et al., 2006; Greedy String Tiling Wise, 1996. Classifying short texts as duplicate is similar to the problem of record linkage, deduplication etc. Databases often have same records and field values which are not syntactically identical but refer to the same entity. This is known as record linkage and it doesn't let data mining algorithms work efficiently. (Torsten Zesch et al., 2012)
Feature engineering has been the core area of focus for most of the well-known traditional methods developed by several data scientists. The common features used are bag of words (BOW), word-to-vector, term frequency and inverse document frequency (TF IDF), unigrams, bigrams, n-grams along with different machine learning models such as Logistic Regression, SVM, Random Forest etc.[28]
With the renaissance of neural networks, deep learning models have been able to achieve performance boost across different NLP techniques especially in semantic Text similarity.[29]-[31]. A research [32] proposed supervised and semi-supervised methods based on LSTM that used region embedding method for embedding the text regions of adjustable dimensions. Another work [33], proposed a Neural Network model and studied documents represented in form of vectors in an integrated manner. First, the model used CNN or LSTM to study the vector form of the sentences. Then, the context of sentences and their relations, of a given document, was determined in the distributed vector representation with recurrent neural network (RNN). Another research [34], proposed a Tree based LSTM model and used it to predict the similarity between two sentences. Skip-thought based approach was proposed which used skip-gram approach of word2vec from the word to sentence level [35].

2.2 Our Approach

First of all, we will analyze the dataset. In the training dataset we have 404290 rows and 6 columns. The percentage of questions which are duplicate is 36.92% and percentage of non-duplicate questions are 63.08%. Hence our dataset is 60/40, which indicates it is almost balanced.
Then, we check whether there are any duplicate question pairs or not. After that we check for any NULL values in dataset. If found, we fix the NULL values.
Then, we extract some basic features like frequency of questions, length of questions, number of common words in question pairs, total number of words in questions, share of words in question pairs, sum of frequency of qid1 and qid2, absolute difference of frequency of qid1 and qid2.
Next, we extract some NLP features like ratio of common-word-count to min length of word count of Q1 and Q2, ratio of common-stop-count to min length of stop count of Q1 and Q2, ratio of common-token-count to min length of token count of Q1 and Q2, if first or last word is same in both questions or not, absolute difference of length of Q1 and Q2, longest substring ratio etc.
After that, we extract some fuzzy features like fuzz-ratio[1], fuzz-partial-ratio[1], token-sort-ratio[1], token-set-ratio[1] etc. using the fuzzywuzzy[1] library.
Adding all the features in the database, we do some univariate analysis like plotting Log Histogram, violin plots, probability density function and bivariate analysis like plotting pair-plot to understand how the features are performing in order to distinguish the duplicate and non-duplicate pairs of questions. We then emphasize on the features which are giving good results.
After that, we download Google-News-Vector[2] (3 billion running words) word vector model (3 million 300-dimension English word vectors) and convert our words into vectors of real numbers. After that we calculate average TF-IDF[3] average word to vector for every sentence by calculating weighted TF-IDF[3] average of words in a sentence.
Next, we head towards calculating word-mover-distance[4] and normalized-word-mover-distance[4] for each question pairs. We also calculate distance metrices like cosine-distance[5], city-block-distance[6], jaccard-distance[7], Canberra-distance[8], Euclidean-distance[9], minkowski-distance[10] etc. for each question pairs.
We add all previous features into dataset and create a final dataset for training our all Machine Learning Models.
Before applying our Machine Learning models, we need to scale the data to normalize the data within a particular range. Sometimes, it also helps in speeding up the calculations in an algorithm.
After scaling, we first build a random model which blindly predicts yes or no for every pair of questions. This model has a worst log loss which we will be use as a benchmark for comparing other models.

We need to fix a probabilistic threshold. If the probability of a question pair being duplicate is greater than that threshold, then we can call that pair as duplicate otherwise, not duplicate. We can change that threshold according to our observation of performance.
Here, we apply Log Loss and Binary Confusion Matrix as our performance metrices. We use log loss because it deals with probability scores and binary confusion matrix because it is a binary classification problem.
Now, we apply different machine learning models such as Logistic Regression with Stochastic Gradient Descent, Linear Support Vector Machine with Stochastic Gradient Descent and hinge loss, Random Forest Classifier, Extra Tree Classifier, Gradient Boost Decision Tree, Stacking Classifier and Adaptive Boosting.
We then compare which model(s) are performing best according to our performance metrices.

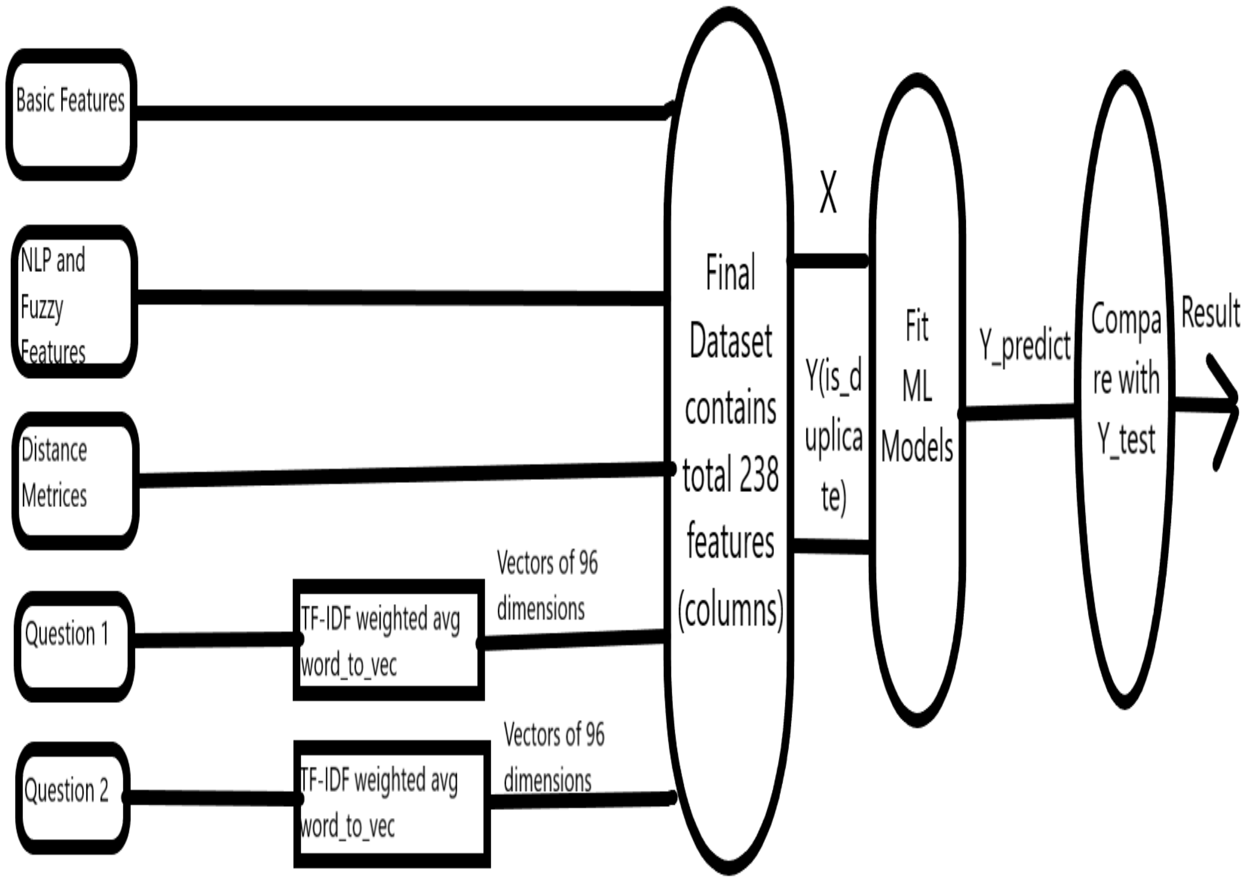


Figure 1 Our Solution Approach in Diagram

3 Dataset

The data is hosted by Kaggle. We are given 404351 number of data fields here, consisting of:

1. id: Looks like a simple rowID
2. qid {1,2}: The unique ID of each question in the pair
3. question {1,2}: The actual textual contents of the questions
4. is-duplicate {0,1}: The label that we are trying to predict - whether the two questions are duplicates of each other.

The labels of whether questions are duplicate or not have been supplied by human experts. Thus, the labelling is prone to noise and hence, there may be some outliers. Here goes some examples of duplicate and non-duplicate questions from the data set.

The question pairs below are duplicates:

- How can I be a good geologist?
- What should I do to be a great geologist?

The question pairs below are non-duplicates:

- How can I increase the speed of my internet connection while using a VPN?
- How can Internet speed be increased by hacking through DNS?

4 Feature Extraction

4.1 Basic Features

- **freq_qid1** = Frequency of qid1's
- **freq_qid2** = Frequency of qid2's
- **q1len** = Length of q1
- **q2len** = Length of q2
- **q1-count-words** = Number of words in Question 1
- **q2-count-words** = Number of words in Question 2
- **word-common-count** = (Number of common unique words in Question 1 and Question 2)
- **word-Total** = (Total number of words in Question 1 + Total number of words in Question 2)
- **word-share** = (word-common)/(word-Total) (Figure 1)
- **frequency_q1+frequency_q2** = sum of frequency of qid1 and qid2
- **frequency_q1-frequency_q2** = absolute difference in frequency of qid1 and qid2

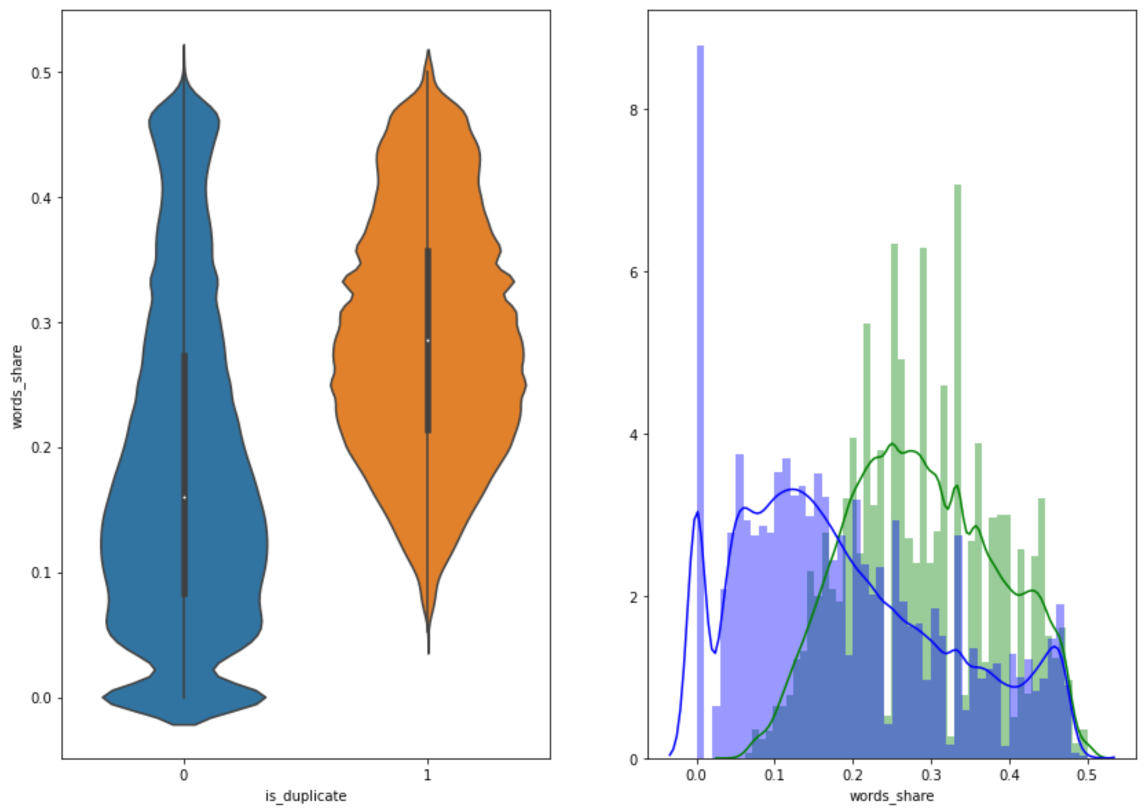


Figure 2 word_share vs is_dupliacte violin plot and pdf

4.2 NLP and Fuzzy Features

Definition:

- **Token**: You get a Token by splitting sentence with the basis of a space
- **Stop-Word**: stop words as per NLTK. [11]
- **Word**: A token that is not a stop-word

Features:

- **Common-word-count-minimum**: Ratio of common_word_count to min length of word count of Q1 and Q2
  cwc_min = common_word_count / (min(len(q1_words), len(q2_words))

- **Common-word-count-maximum**: Ratio of common_word_count to max lenghth of word count of Q1 and Q2
  cwc_max = common_word_count / (max(len(q1_words), len(q2_words))

- **Common-stop-count-minimum**: Ratio of common_stop_count to min lenghth of stop count of Q1 and Q2
  csc_min = common_stop_count / (min(len(q1_stops), len(q2_stops))

- **Common-stop-count-maximum**: Ratio of common_stop_count to max lenghth of stop count of Q1 and Q2
  csc_max = common_stop_count / (max(len(q1_stops), len(q2_stops))

- **Common-token-count-minimum**: Ratio of common_token_count to min length of token count of Q1 and Q2
  ctc_min = common_token_count / (min(len(q1_tokens), len(q2_tokens))

- **Common-token-count-maximum**: Ratio of common_token_count to max length of token count of Q1 and Q2
  ctc_max = common_token_count / (max(len(q1_tokens), len(q2_tokens))

- **Last-word-equivalent**: Check if Last word of both questions is equal or not. Returns Boolean value
  last_word_eq = int(q1_tokens[-1] == q2_tokens[-1])

- **First-word-equivalent**: Check if First word of both questions is equal or not. Returns Boolean value
  first_word_eq = int(q1_tokens[0] == q2_tokens[0])

- **absolute-length-difference**: Abs. length difference
  abs_len_diff = abs(length(q1_tokens) - length(q2_tokens))

- **mean-length**: Average Token Length of both Questions
  mean_length = (length(q1_tokens) + length(q2_tokens))/2

- **longest-substring-ratio:** Ratio of length longest common substring to min length of token count of Q1 and Q2 longest-substring-ratio = length (longest common substring) / (min(length(q1_tokens), length(q2_tokens))

We need to install the fuzzywuzzy[1] library to use the next features:

‘pip install fuzzywuzzy’[1]

- **fuzz_ratio**: [1] [13] (Figure2) shows a violin plot and probability distribution function between fuzz_ratio and is_duplicate

- **fuzz_partial_ratio**: [1] [13] (Figure3) shows a violin plot and probability distribution function between fuzz_partial_ratio and is_duplicate

- **token_sort_ratio**: [1] [13] (Figure4) shows a violin plot and probability distribution function between token_sort_ratio and is_duplicate

- **token_set_ratio**: [1] [13]

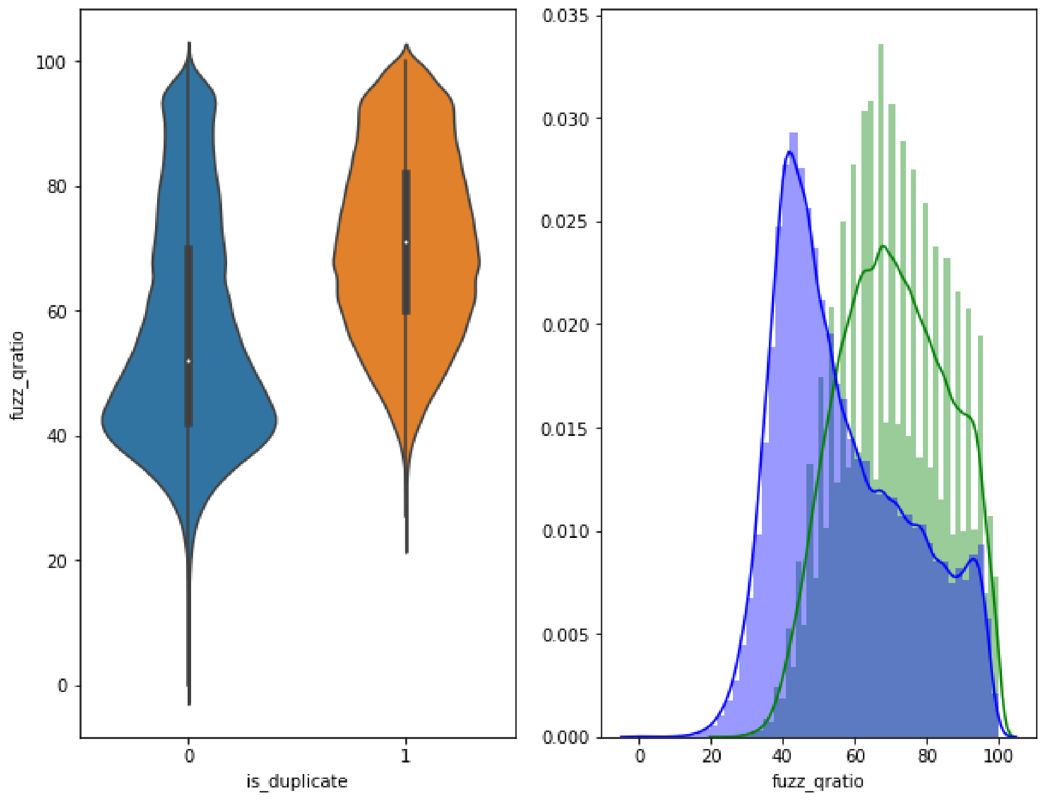


Figure 3 fuzz_ratio vs is_dupliacte violin plot and pd

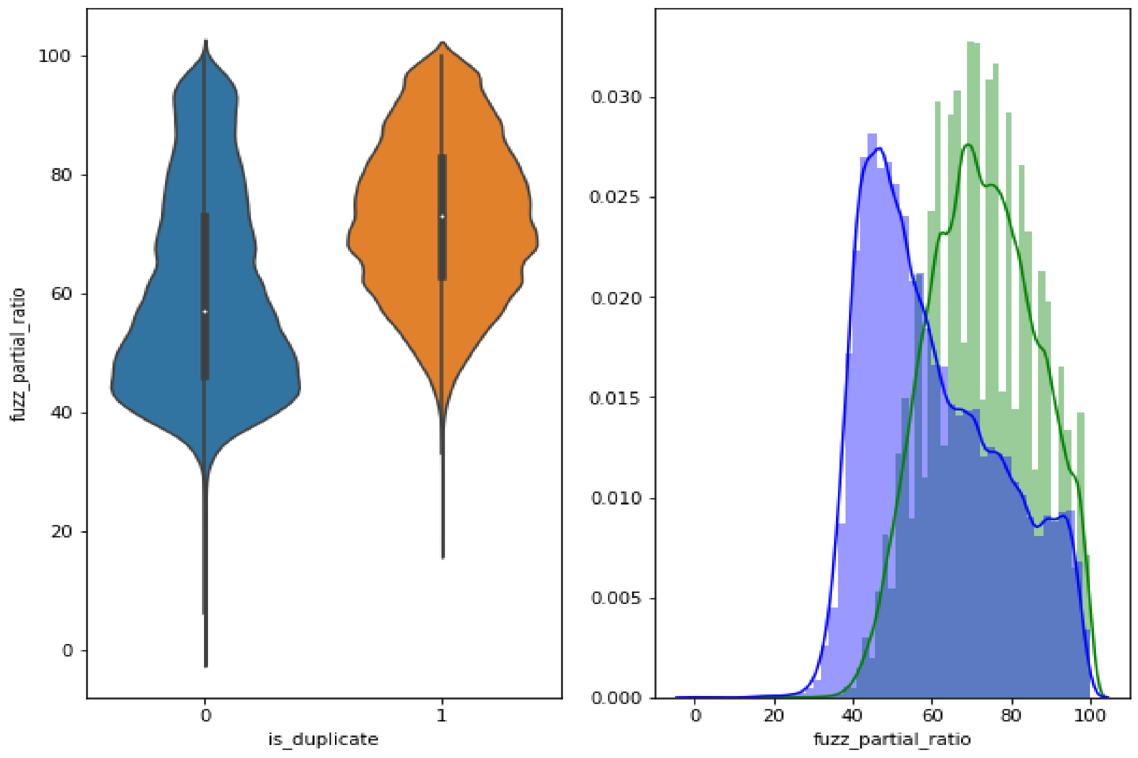


Figure 4 fuzz_partial_ratio vs is_duplicate violin plot and pdf

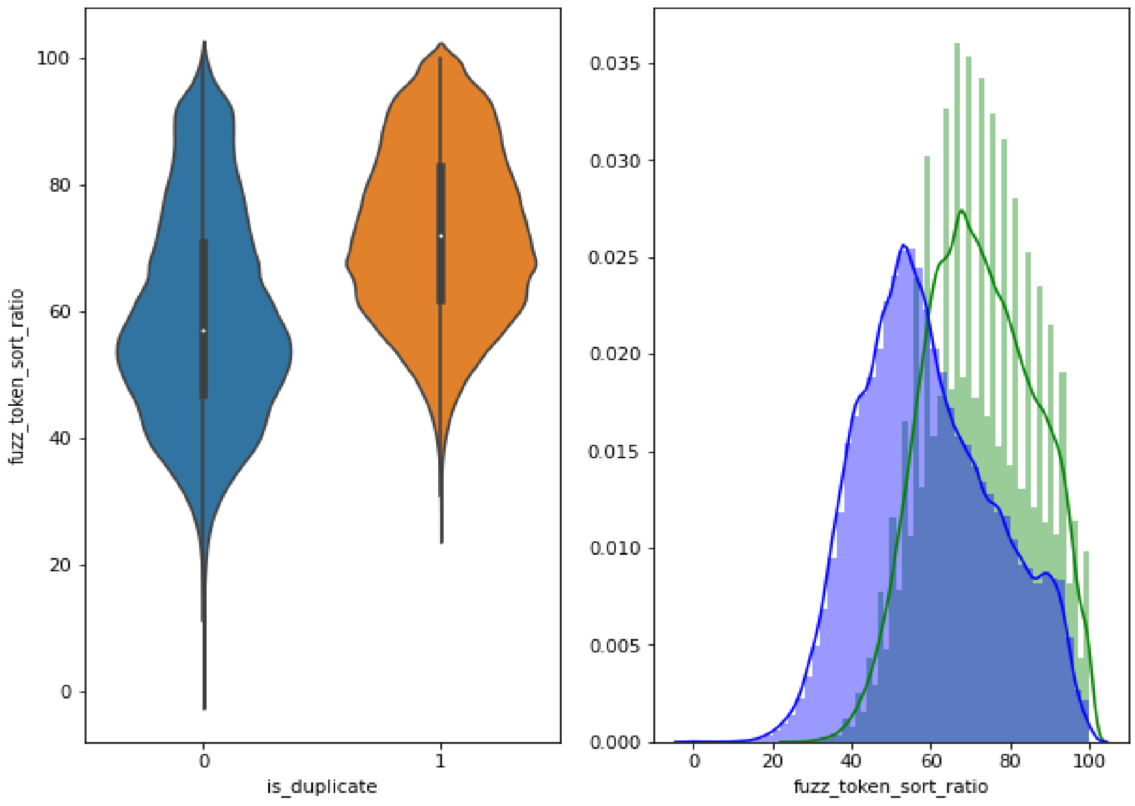


Figure 5 fuzz_token_sort_ratio vs is_duplicate violin plot and pdf

Here goes a pair plot between some of the basic features and fuzzy features.

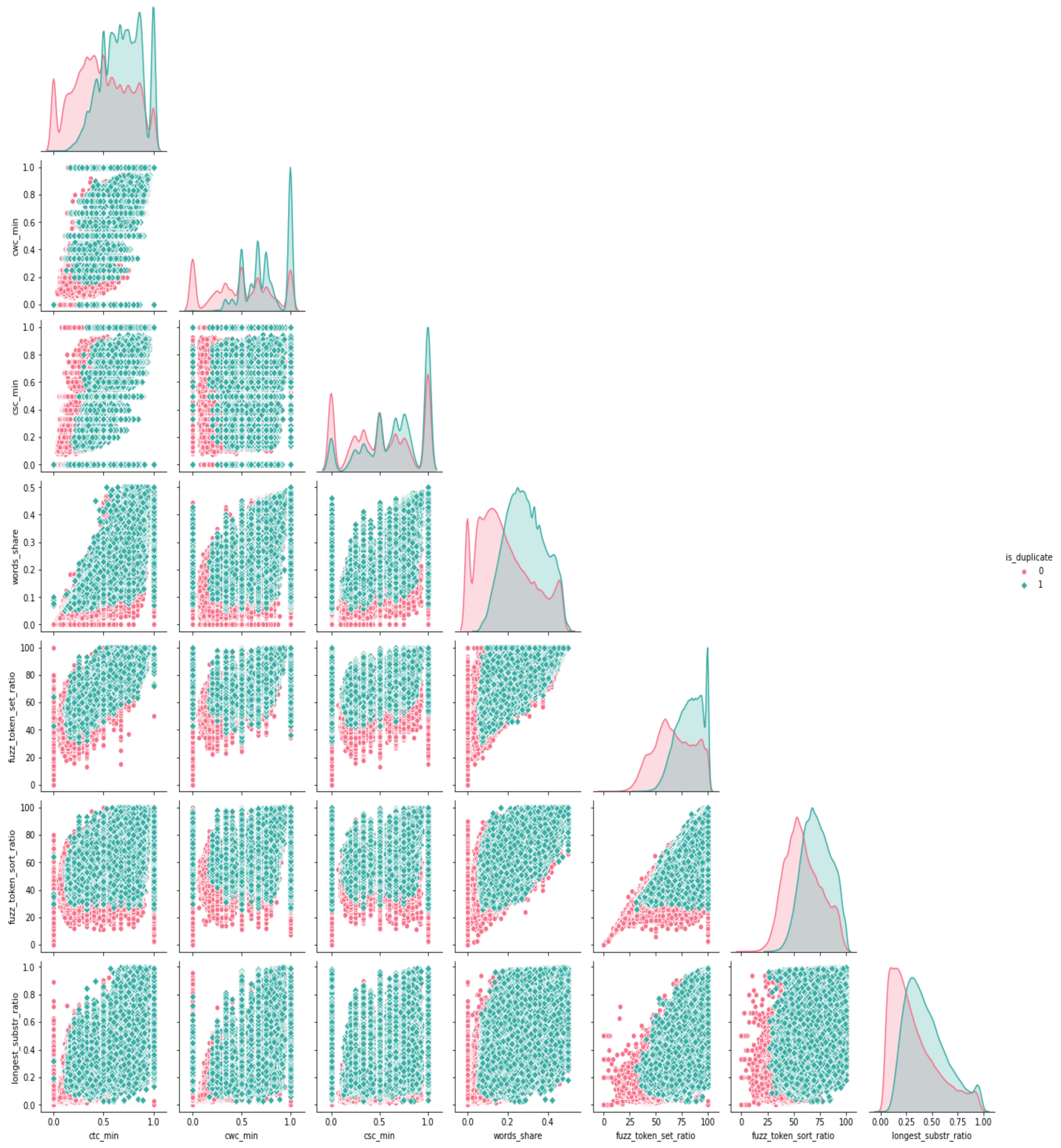


Figure 6 Pair plot of different basic and fuzzy features

4.3 Word Embedding

Word embedding is a very popular NLP technique to map words to vectors of real numbers. Word embedding is one of the most popular representation of document vocabulary. It is capable of capturing context of a word in a document, semantic and syntactic similarity, relation with other words. From this work, it is much easier for people to find synonym of one word and distance of two words using vectors. Word embedding has been proved to boost the performance in NLP problem such as sentiment analysis, duplicate detection. I find that pre-trained **Google News** corpus[2] word vector model (3 million 300-dimension English word vectors) performs excellent for this dataset. It is a deep learning algorithm for obtaining vector representations for words. Here, I use the one called GoogleNews-vectors-negative300.bin.gz.[2]

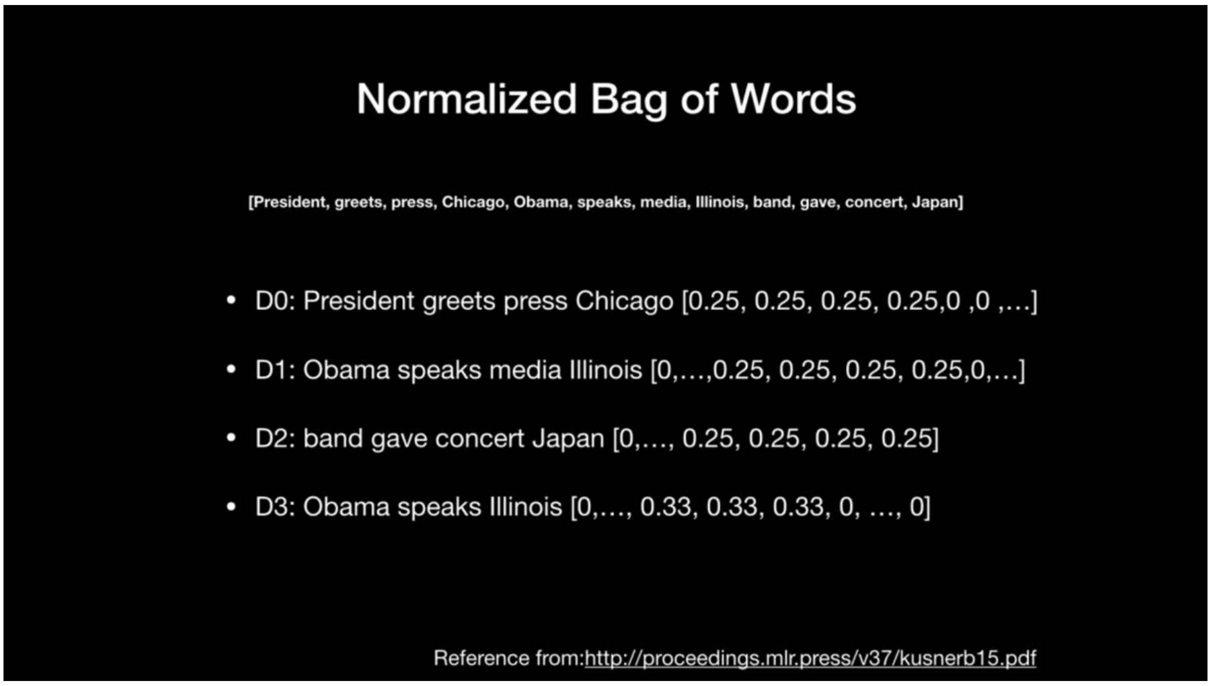
Code for downloading the Google-News-Vector:

```
!wget https://s3.amazonaws.com/dl4j-distribution/GoogleNews-vectors-negati
ve300.bin.gz
```

After loading the vector into dataset, I convert all the words in training and test set into vectors of real numbers and d dimensions. Then I calculate the TF-IDF[3] weighted average word to vector for all the sentences from word vectors. So, now all the sentence gets converted to a particular d-dimension vector space.

```
vectors1 = []
for ques1 in tqdm(list(data['question1'])):
    doc1 = nlp(ques1)
    mean_vec1 = np.zeros([len(doc1), len(doc1[0].vector)])
    for word1 in doc1:
        vec1 = word1.vector
        try:
            idf = word2tfidf[str(word1)]
        except:
            idf = 0
        mean_vec1 += vec1 * idf
    mean_vec1 = mean_vec1.mean(axis=0)
    vectorss1.append(mean_vec1)
data['q1_feats_m'] = list(vectorss1)
```

I compute **word_mover_distance**[4] and **normalized_word_mover_distance**[4] for each word vector. Figure 6-9 shows how word_mover_disatnce is calculated.



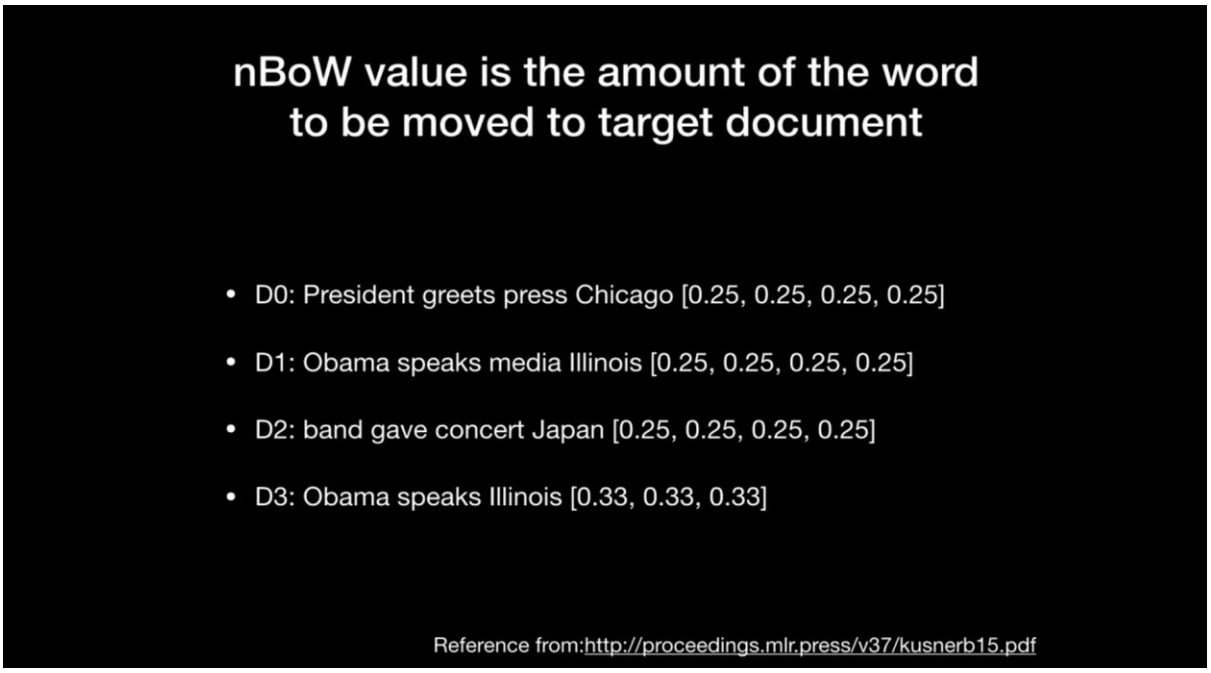
Normalized Bag of Words

[President, greets, press, Chicago, Obama, speaks, media, Illinois, band, gave, concert, Japan]

- D0: President greets press Chicago [0.25, 0.25, 0.25, 0.25,0 ,0 ,…]
- D1: Obama speaks media Illinois [0,…,0.25, 0.25, 0.25, 0.25,0,…]
- D2: band gave concert Japan [0,…, 0.25, 0.25, 0.25, 0.25]
- D3: Obama speaks Illinois [0,…, 0.33, 0.33, 0.33, 0, …, 0]

Reference from:http://proceedings.mlr.press/v37/kusnerb15.pdf

Figure 7 word_mover_distance



nBoW value is the amount of the word to be moved to target document

- D0: President greets press Chicago [0.25, 0.25, 0.25, 0.25]
- D1: Obama speaks media Illinois [0.25, 0.25, 0.25, 0.25]
- D2: band gave concert Japan [0.25, 0.25, 0.25, 0.25]
- D3: Obama speaks Illinois [0.33, 0.33, 0.33]

Reference from:http://proceedings.mlr.press/v37/kusnerb15.pdf

Figure 8 word_mover_distance

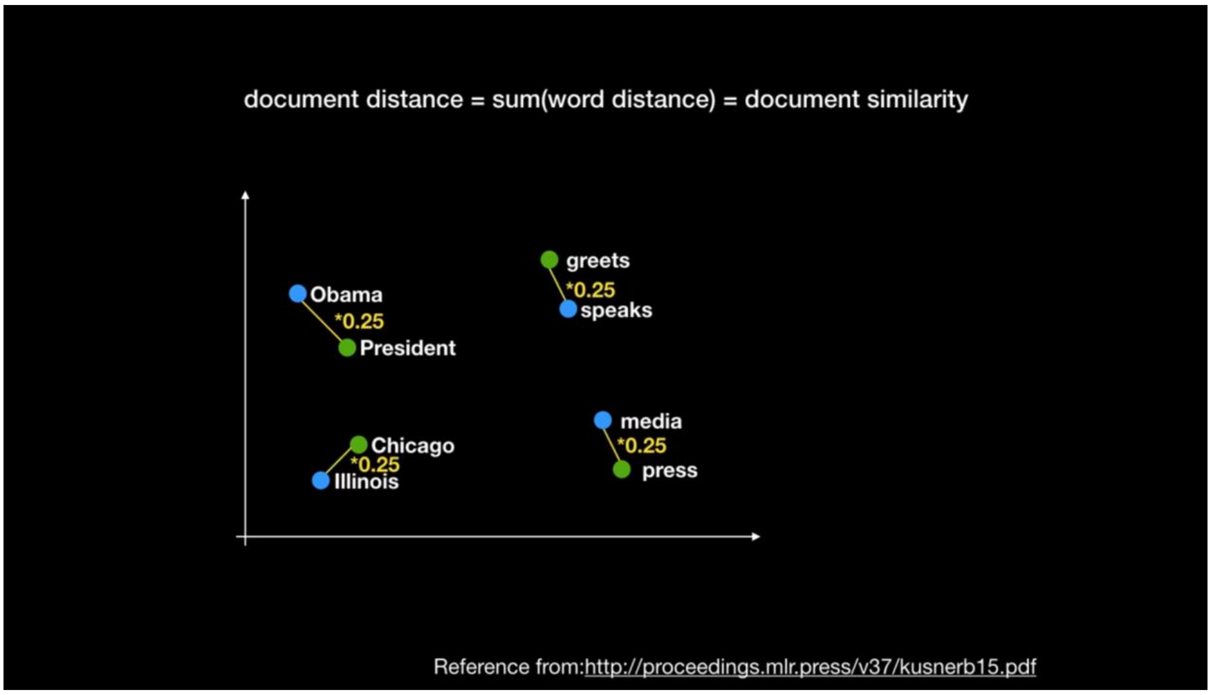
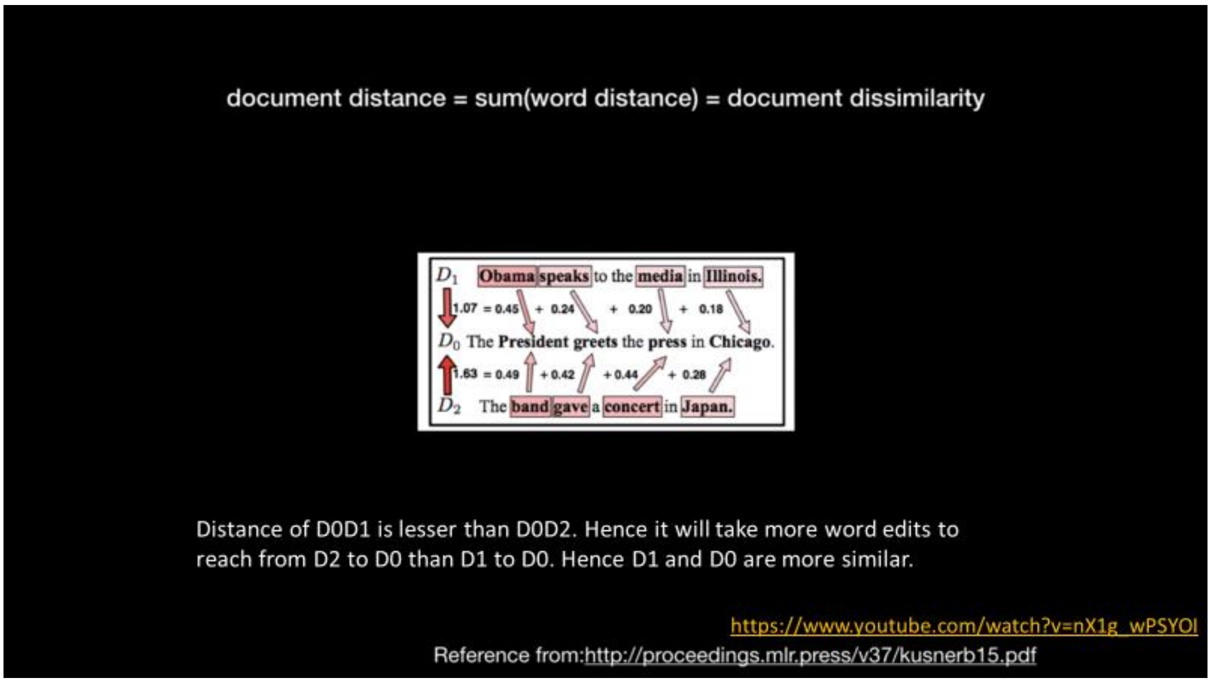

document distance = sum(word distance) = document similarity
Obama
*0.25
President
greets
*0.25
speaks
Chicago
*0.25
Illinois
media
*0.25
press
Reference from:http://proceedings.mlr.press/v37/kusnerb15.pdf

Figure 9 word_mover_distance


document distance = sum(word distance) = document dissimilarity
D_1 Obama speaks to the media in Illinois.
1.07 = 0.45 + 0.24 + 0.20 + 0.18
D_0 The President greets the press in Chicago.
1.63 = 0.49 + 0.42 + 0.44 + 0.28
D_2 The band gave a concert in Japan.
https://www.youtube.com/watch?v=nX1g_wPSYOI
Reference from:http://proceedings.mlr.press/v37/kusnerb15.pdf

Distance of D0D1 is lesser than D0D2. Hence it will take more word edits to reach from D2 to D0 than D1 to D0. Hence D1 and D0 are more similar.

Figure 10 word_mover_distance

```
model = gensim.models.KeyedVectors.load_word2vec_format('GoogleNews-vec
tors-negative300.bin.gz', binary=True)
data['wmd'] = data.apply(lambda x: word_mover_distance(x['question1'],
x['question2']), axis=1) #'word_mover_distance' added to data columns
[13]
```

id	Question1	Question2	Word_mover_distance
0	what is the step by step guide to invest in sh...	what is the step by step guide to invest in sh...	0.640008
1	what is the story of kohinoor kohinoor dia...	what would happen if the Indian government sto...	2.472493
2	how can I increase the speed of my internet co...	how can internet speed be increased by hacking...	1.922139

Table 1 Word_Mover_Distance

```
norm_model = gensim.models.KeyedVectors.load_word2vec_format('GoogleNew
s-vectors-negative300.bin.gz', binary=True)
norm_model.init_sims(replace=True)
data['norm_wmd'] = data.apply(lambda x: normalized_word_mover_distance(
x['question1'], x['question2']), axis=1)
[13]
```

id	Question1	Question2	Normalized_Word_mover_distance
0	what is the step by step guide to invest in sh...	what is the step by step guide to invest in sh...	0.198042
1	what is the story of kohinoor kohinoor dia...	what would happen if the Indian government sto...	0.877940
2	how can i increase the speed of my internet co...	how can internet speed be increased by hacking...	0.694896

Table 2 Normalized_Word_Mover_Distance

Then, I compute various distance metrics from those d-dimensional vectors, such as **cosine_distance**[5], **cityblock_distance**[6], **jaccard_distance**[7], **canberra_distance**[8], **euclidean_distance**[9], **minkowski_distance**[10].

```
data['cosine_distance'] = [cosine(x, y) for (x, y) in zip(np.nan_to_num
(q1_vectors), np.nan_to_num(q2_vectors))] [13]
```

```
data['minkowski_distance'] = [minkowski(x, y, 3) for (x, y) in zip(np.n
an_to_num(q1_vectors), np.nan_to_num(q2_vectors))] [13]
```

I also calculate **skewness**[23] **and kurtosis**[23] (Pearson) of the dataset.

```
data['skew_q1vec'] = [skew(x) for x in np.nan_to_num(q1_vectors)] [13]
```

```
data['kur_q1vec'] = [kurtosis(x) for x in np.nan_to_num(q1_vectors)] [1
3]
```

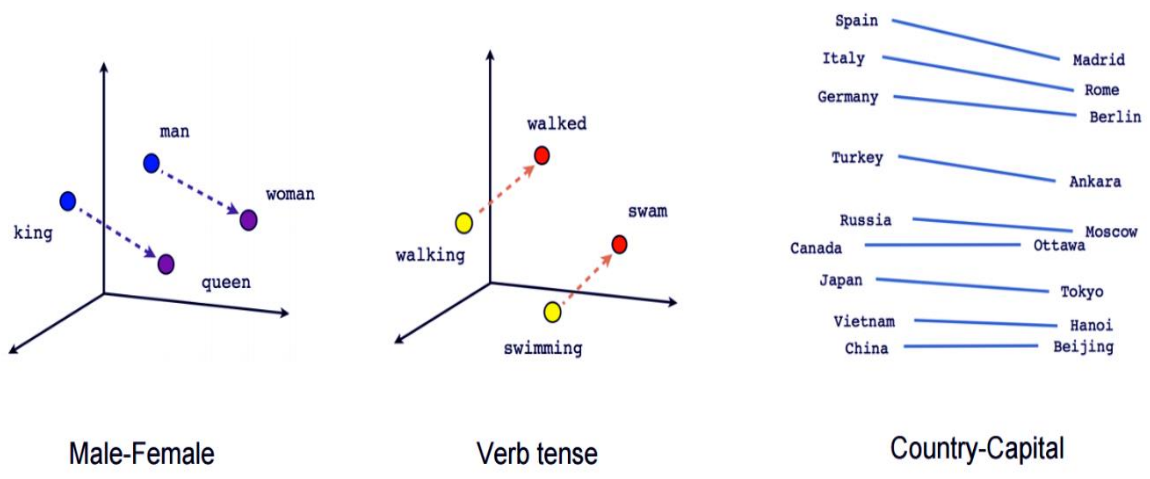


Figure 11 Word to Vector Conversion Example[14]

Finally, I save all these basic features, fuzzy[1] features, TF-IDF[3] weighted average word to vector for all sentences, distance metrices into .csv file for the future use.

5

Preparing the data for fitting Models

5.1 Fixing the Missing, Indefinite values

We run the below command to print the rows with null values:

```
nan_rows = final_data[final_data.isnull().any(1)]
print (nan_rows)
```

We run the below command to print the columns with null values:

```
nan_values = final_data.isna()
nan_columns = nan_values.any()

columns_with_nan = final_data.columns[nan_columns].tolist()
print(columns_with_nan)
```

Now, we run the below command to check whether there is any value in the dataset which is greater than float64 which is considered as infinity:

```
np.where(final_data.values >= np.finfo(np.float64).max)
```

Now, we fix the Nan, positive infinity and negative infinity values in dataset:

```
final_data = final_data[~final_data.isin([np.nan, np.inf, -np.inf]).any(1)]
```

5.2 Splitting the data into train and test dataset

Here, it could be perfect if timestamp was given along with the questions in the dataset. Then we could have split the dataset according to time series. The past 70 percent data could be considered as train dataset and the present 30 percent data could be considerd as test dataset. But, since here no timestamp is provided, we have to go for random train test splitting. Here we are going for 70:30 train test splitting.

```
X_train_final,X_test_final, y_train_final, y_test_final = train_test_sp
lit(final_data, y_true, stratify=y_true, test_size=0.3,random_state=13)
```

6

Fitting different Machine Learning Models

6.1 Random Model with worst case Log Loss

We find, Log loss on Test Data using Random Model 0.8836919522842575
Hence the worst log loss that can happen is 0.8836919522842575 This is the tightest b
ound of log loss. Any model should have a log loss lesser than this.

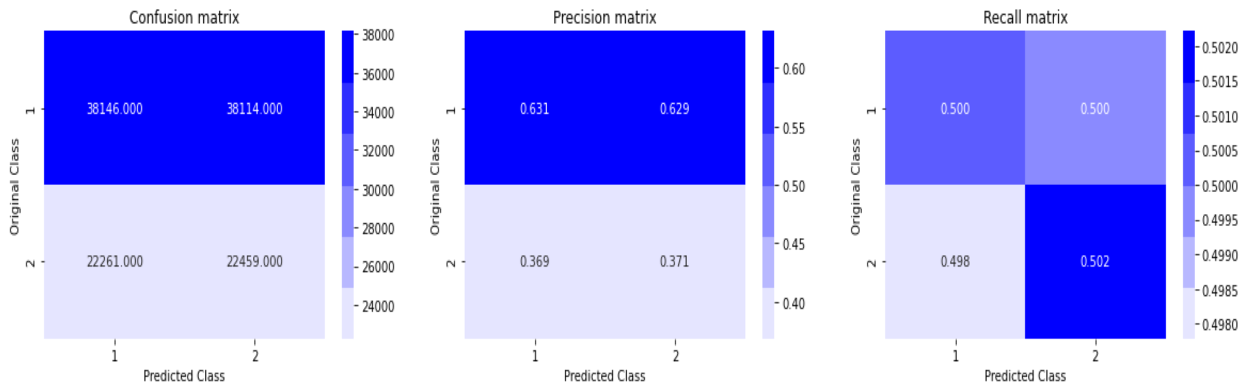


Figure 12 Random Model Confusion Matrix Result

Hence, we can conclude that True Positive Rate, True Negative Rate is not quite high and False Positive Rate and False Negative Rate is not quite low. So, we can say, our model is a dumb and not at all working correctly. Any model should work better than this. This is for setting a benchmark for the worst case.

6.2 Logistic Regression with SGD[15] and Log Loss

We apply Logistic Regression with Stochastic Gradient Descent[15] and L2 regularization and the loss function is Log Loss. We also plot confusion matrix to get True Positive, True Negative, False Positive, False Negative, Precision Matrix, Recall Matrix.

```
alpha = np.random.uniform(0.000025,0.00035,14)
alpha = np.round(alpha,7)
alpha.sort()
```

```
clf = SGDClassifier(alpha=i, penalty='l2', loss='log', random_state=42)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(X_train_final, y_train_final)
```

```
predict_y_train = sig_clf.predict_proba(X_train_final)
print('For best alpha = ', alpha[best_alpha], "The train log loss is:",
log_loss(y_train_final, predict_y_train,eps=1e-15))
predict_y_test = sig_clf.predict_proba(X_test_final)
print('For best alpha = ', alpha[best_alpha], "The test log loss is:",l
og_loss(y_test_final, predict_y_test,eps=1e-15))
predicted_y =np.argmax(predict_y_test,axis=1) # from the whole column o
f predicted_y picking the highest value
```

The Result came out as:

```
For best alpha = 0.0002032 The train log loss is: 0.3975058281421835
For best alpha = 0.0002032 The test log loss is: 0.400262401191614
```

Now, since the train log loss and test log loss is almost similar, we can conclude that o ur Logistic Regression Model is not overfitting or underfitting. It is working correctly.

The result of confusion matrix is as follows:

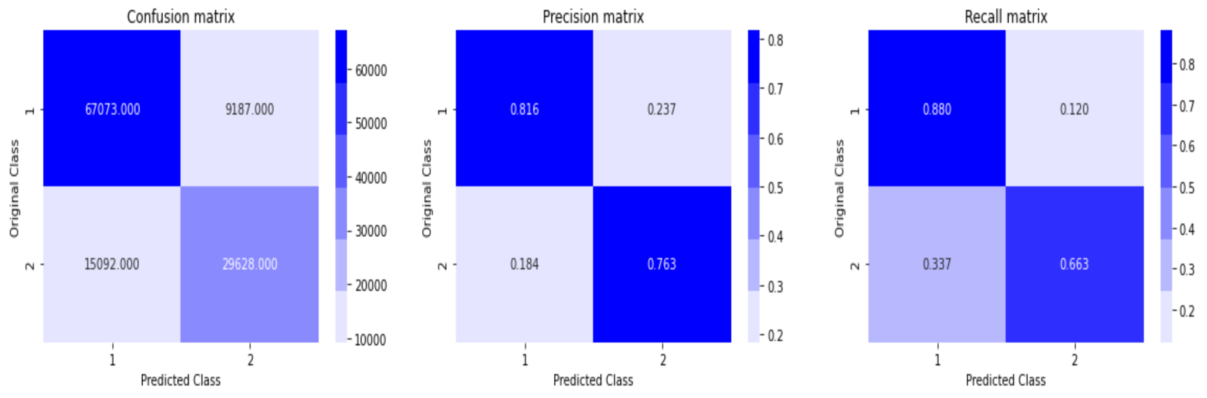


Figure 13 Logistic Regression Confusion Matrix Result

Hence, we can conclude that True Positive Rate, True Negative Rate is quite high and False Positive Rate and False Negative Rate is quite low. As a result, Precision and Recall is also quite high. So, we can say, our model is working correctly.

6.3 Linear SVM with SGD[15] and Hinge Loss

We apply Support Vector Machine with Stochastic Gradient Descent[15], Calibrated Classifier[22] and L2 regularization and the loss function is Log Loss. We also plot confusion matrix to get True Positive, True Negative, False Positive, False Negative, Precision Matrix, Recall Matrix.

```
alpha = np.random.uniform(0.000020,0.00032,14)
alpha = np.round(alpha,7)
alpha.sort()
```

```
clf = SGDClassifier(alpha=i, penalty='l2', loss='hinge', random_state=1
#applying hinge loss to apply svm
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(X_train_final, y_train_final)
```

```
predict_y = sig_clf.predict_proba(X_train_final)
print('For best alpha = ', alpha[best_alpha], "The train log loss is:",
log_loss(y_train_final, predict_y,eps=1e-15))
predict_y = sig_clf.predict_proba(X_test_final)
print('For best alpha = ', alpha[best_alpha], "The test log loss is:",l
og_loss(y_test_final, predict_y,eps=1e-15))
predicted_y =np.argmax(predict_y,axis=1)
```

The Result came out as:
```
For best alpha = 0.0001291 The train log loss is: 0.40061898857796839
For best alpha = 0.0001291 The test log loss is: 0.40329099571822147
```

Now, since the train log loss and test log loss is almost similar, we can conclude that o ur Linear Support Vector Machine Model is not overfitting or underfitting. It is worki ng correctly.

The result of confusion matrix is as follows:

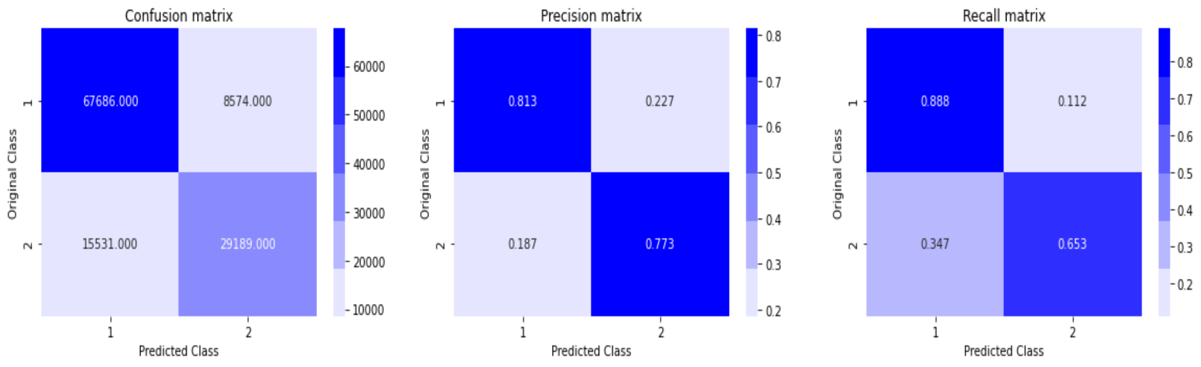


Figure 14 SVM Confusion Matrix Result

Hence, we can conclude that True Positive Rate, True Negative Rate is quite high and False Positive Rate and False Negative Rate is quite low. As a result, Precision and Recall is also quite high. So, we can say, our model is working correctly.

6.4 Random Forest Classifier[16] (Bagging) with Log Loss

We apply Random Forest Classifier[16] with max_depth=12 and estimators = [75,100,150,200,300,400,600] and the loss function is Log Loss. We also plot confusion matrix to get True Positive, True Negative, False Positive, False Negative, Precision Matrix, Recall Matrix.

```
estimators = [75,100,150,200,300,400,600]
```

```
clf = RFC(n_estimators=i,max_depth=12,n_jobs=-1)#low bias high variance
model, as depth increases variance increases. while bagging the varianc
e will come down automatically in fact very low.
```

```
clf.fit(X_train_final,y_train_final)
predict_y = clf.predict_proba(X_train_final)
log_loss_train = log_loss(y_train_final, predict_y, eps=1e-15)
predict_y = clf.predict_proba(X_test_final)
log_loss_test = log_loss(y_test_final, predict_y, eps=1e-15)
```

The Result came out as:

```
estimators = 75 Train Log Loss 0.34381762823424866 Test Log Loss 0.3759
4589804755857
estimators = 100 Train Log Loss 0.34321467175912385 Test Log Loss 0.376
0628832327492
estimators = 150 Train Log Loss 0.3426310615865547 Test Log Loss 0.3754
429488109406
estimators = 200 Train Log Loss 0.3415642687851075 Test Log Loss 0.3740
0762711869606
estimators = 300 Train Log Loss 0.342455260049411 Test Log Loss 0.37531
94256730679
estimators = 400 Train Log Loss 0.342692041877361 Test Log Loss 0.37513
82081822963
estimators = 600 Train Log Loss 0.3425744387941161 Test Log Loss 0.3747
2982375945785
```

Now, since the train log loss and test log loss are almost similar, we can conclude that our Random Forest Classifier Model[16] is not overfitting or underfitting. It is workin g correctly.

The result of confusion matrix is as follows:

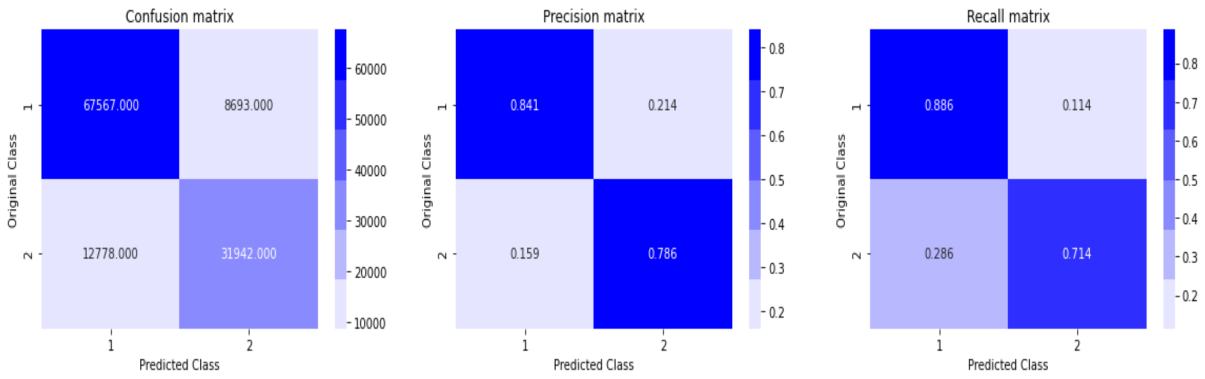


Figure 15 Random Forest Classifier Confusion Matrix Result

Hence, we can conclude that True Positive Rate, True Negative Rate is quite high and False Positive Rate and False Negative Rate is quite low. As a result, Precision and Recall is also quite high. So, we can say, our model is working correctly.

6.5 Gradient Boost Decision Tree[18] (Boosting) with Log Loss

We apply Gradient Boost Decision Tree of XgBoost[18] Library with max_depth=12 and n_estimators = 80 and learning rate is 0.08. We also plot confusion matrix to get True Positive, True Negative, False Positive, False Negative, Precision Matrix, Recall Matrix.

```
clf = xgb.XGBClassifier(max_depth=12, n_estimators=80, learning_rate=0.
08, colsample_bytree=.7, gamma=0, reg_alpha=4, objective='binary:logist
ic', eta=0.3, silent=1, subsample=0.8)
```

```
clf.fit(X_train_final,y_train_final)
predict_y = clf.predict_proba(X_train_final)
print("The train log loss is:",log_loss(y_train_final, predict_y, eps=1
e-15))
predict_y = clf.predict_proba(X_test_final)
print("The test log loss is:",log_loss(y_test_final, predict_y, eps=1e-
15))
predicted_y =np.argmax(predict_y,axis=1)
```

The Result came out as:

```
The train log loss is: 0.23062255281875668
The test log loss is: 0.31667970755105135
```

Now, since the train log loss and test log loss is almost similar, we can conclude that o ur Gradient Boost Decision Tree Model[18] is not overfitting or underfitting. It is wor king correctly.

The result of confusion matrix is as follows:

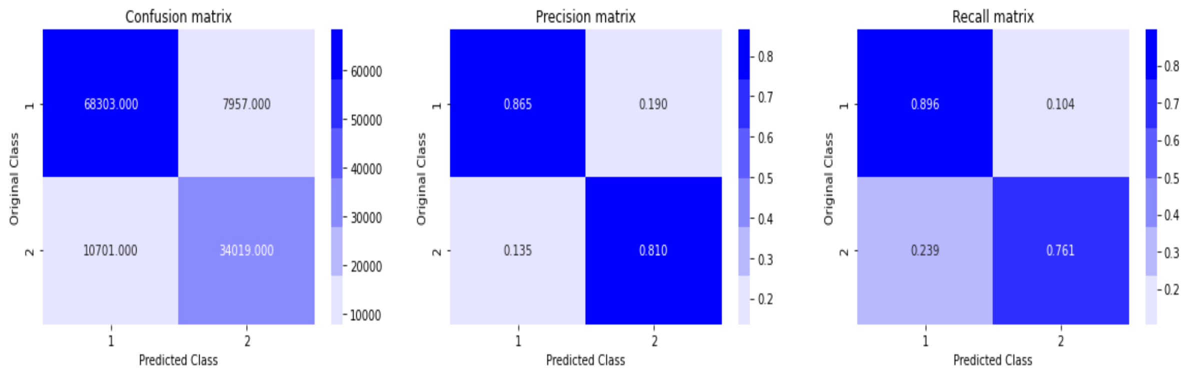


Figure 16 Gradient Boost Decision Tree Confusion Matrix Result

Hence, we can conclude that True Positive Rate, True Negative Rate is quite high and False Positive Rate and False Negative Rate is quite low. As a result, Precision and Recall is also quite high. So, we can say, our model is working correctly.

6.6 Extra Tree Classifier[17] with Log Loss

We apply Extra Tree Classifier[17] with max_depth=11 and estimators = [75,100,150,200,300,400,600] and the loss function is Log Loss. We also plot confusion matrix to get True Positive, True Negative, False Positive, False Negative, Precision Matrix, Recall Matrix.

```
estimators = [75,100,150,200,300,400,600]
```

```
exc_clf = EXC(n_estimators=i,max_depth=11,n_jobs=-1)
```

```
exc_clf.fit(X_train_final,y_train_final)
predict_y = exc_clf.predict_proba(X_train_final)
log_loss_train = log_loss(y_train_final, predict_y, eps=1e-15)
predict_y = exc_clf.predict_proba(X_test_final)
log_loss_test = log_loss(y_test_final, predict_y, eps=1e-15)
```

The Result came out as:

```
estimators = 75 Train Log Loss 0.450102611906145 Test Log Loss 0.457482
4435144607
estimators = 100 Train Log Loss 0.45098598666278017 Test Log Loss 0.457
9522099853415
estimators = 150 Train Log Loss 0.4525042349159359 Test Log Loss 0.4595
575202374123
estimators = 200 Train Log Loss 0.45120648194586865 Test Log Loss 0.458
38589870297236
estimators = 300 Train Log Loss 0.450591165516195 Test Log Loss 0.45787
7510118349
estimators = 400 Train Log Loss 0.45174390149749843 Test Log Loss 0.458
81073819637014
estimators = 600 Train Log Loss 0.4521951937339841 Test Log Loss 0.4592
1927085099207
```

Now, since the train log loss and test log loss is almost similar, we can conclude that o ur Extra Tree Classifier Model[17] is not overfitting or underfitting. It is working corr ectly.

The result of confusion matrix is as follows:

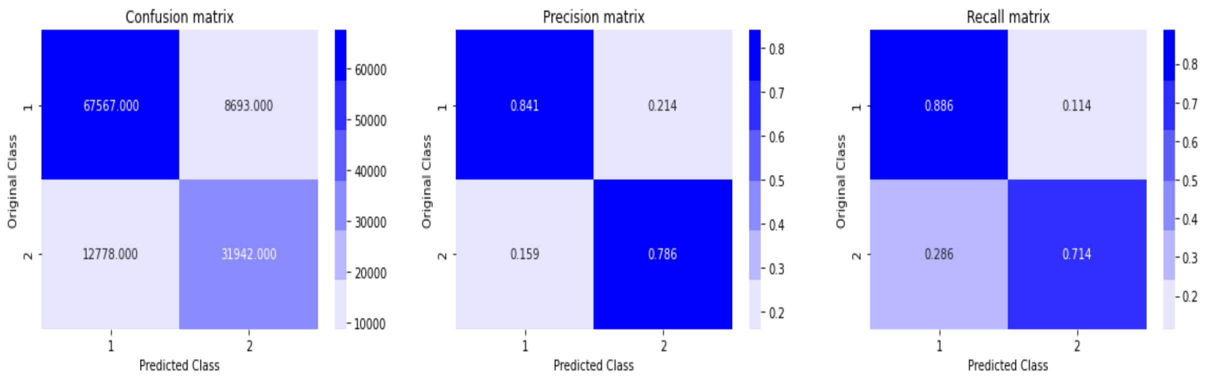


Figure 17 Extra Tree Classifier Confusion Matrix Result

Hence, we can conclude that True Positive Rate, True Negative Rate is quite high and False Positive Rate and False Negative Rate is quite low. As a result, Precision and Recall is also quite high. So, we can say, our model is working correctly.

6.7 Stacking Classifier[19][20] with Log Loss

We apply Stacking Classifier[19][20] with initial estimator as 1. Logistic Regression with log loss and L2 regularization 2. Linear SVM with hinge loss and L2 regularization 3. Random Forest Classifier with n_estimators=70 and max_depth=12. As a final estimator we keep Gradient Boost Decision Tree with max_depth=10, subsample=0.85.

```
estimators = [('rf', RandomForestClassifier(n_estimators=70, max_depth=
12, random_state=42)), ('sgc', SGDClassifier(alpha=10**(-5), penalty='l
2', loss='hinge', random_state=42)), ('sgdc', (SGDClassifier(alpha=10**
(-5), penalty='l2', loss='log', random_state=42)))]
clf = StackingClassifier(estimators=estimators, final_estimator=xgb.XGB
Classifier(max_depth=10,learning_rate=0.02,n_estimators=400,n_jobs=-1,
subsample=0.85, colsample_bytree=0.85))
```

```
clf.fit(X_train_final, y_train_final)
predict_y = clf.predict_proba(X_train_final)
print("The train log loss is:",log_loss(y_train_final, predict_y, eps=1
e-15))
predict_y = clf.predict_proba(X_test_final)
print("The test log loss is:",log_loss(y_test_final, predict_y, eps=1e-
15))
```

The result came out as follows:

```
The train log loss is: 0.30921514588491233
The test log loss is: 0.34918981438080887
```

Now, since the train log loss and test log loss are almost similar, we can conclude that our Stacking Classifier Model[19][20] is not overfitting or underfitting. It is working c orrectly. It takes long time to run. So, we could not test much changing different param eters. But we are hopeful that if we can choose the parameters optimally this model wi ll give very good result.

The result of confusion matrix is as follows:

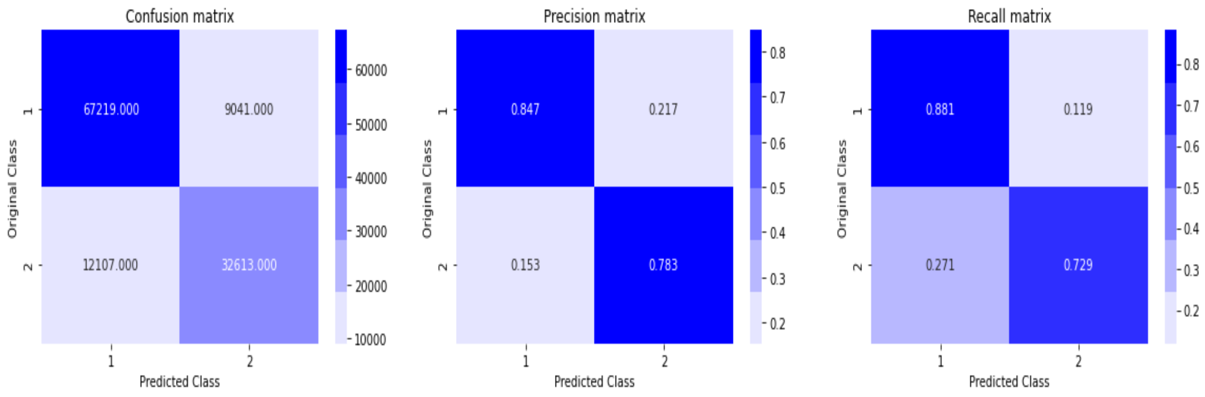


Figure 18 Stacking Classifier Confusion Matrix Result

Hence, we can conclude that True Positive Rate, True Negative Rate is quite high and False Positive Rate and False Negative Rate is quite low. As a result, Precision and Recall is also quite high. So, we can say, our model is working correctly.

6.8 Adaptive Boosting[21] with Log Loss

We apply Adaptive Boosting[21] with n_estimators = 75 and algorithm = SAMME.R

```
abc_clf = abc(n_estimators=75, learning_rate=0.02, algorithm='SAMME.R',
random_state=42)
```

```
abc_clf.fit(X_train_final,y_train_final)
predict_y = clf.predict_proba(X_train_final)
print("The train log loss is:",log_loss(y_train_final, predict_y, eps=1
e-15))
predict_y = abc_clf.predict_proba(X_test_final)
print("The test log loss is:",log_loss(y_test_final, predict_y, eps=1e-
15))
```

The result came out as follows:

```
The train log loss is: 0.30921514588491233
The test log loss is: 0.5325878474916613
```

Now, since the train log loss and test log loss are almost similar, we can conclude that our Adaptive Boosting[21] Model is not overfitting or underfitting. It is working corre ctly. But the result this is providing is the worst among all the models. So, we don't co nsider it as a good model.

The result of confusion matrix is as follows:

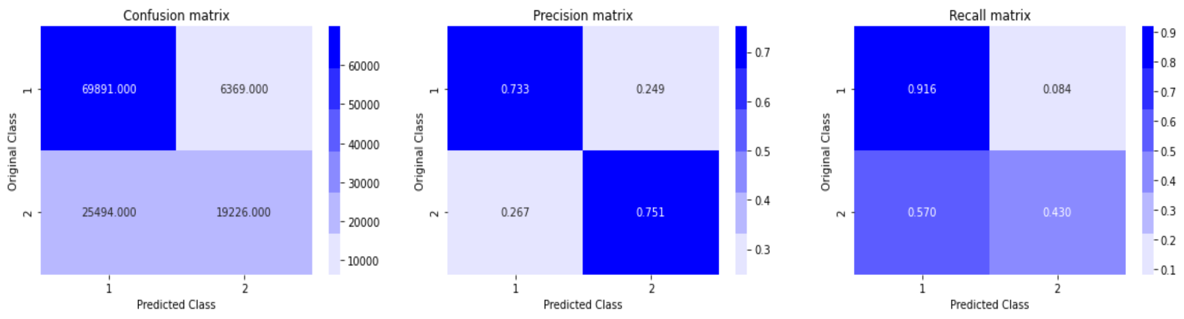


Figure 19 Adaptive Boosting Confusion Matrix Result

Hence, we can conclude that True Positive Rate, True Negative Rate is quite high and False Positive Rate and False Negative Rate is quite low. As a result, Precision and Recall is also quite high. So we can say, model is working correctly.

Conclusion and Recommendation

After going through all the results that different Machine Learning Models are providing, we came into conclusion that:

- Logistic Regression with Stochastic Gradient Descent and SVM with Stochastic Gradient Descent is working fine with test log loss 0.40.

- But the best result is given by Gradient Boost Decision Tree with test log loss 0.31 and Stacking Classifier with test log loss 0.34.

- Stacking Classifier took very long time to run. So, we could not test much with hyperparameter tuning. But, if we can choose parameters optimally, this model should perform the best among all.

- We can still improve the accuracy of the model by correcting the mis-spellings in the text.

- We can improve the accuracy if we can find more relevant features.

- We can also apply Deep Learning techniques like LSTM, Bi-LSTM for better results than traditional Machine Learning models.

Bibliography

[1] https://github.com/seatgeek/fuzzywuzzy

[2] https://code.google.com/archive/p/word2vec/

[3] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

[4] https://markroxor.github.io/gensim/static/notebooks/WMD_tutorial.html

[5] https://en.wikipedia.org/wiki/Cosine_similarity

[6] https://docs.tibco.com/pub/spotfire/6.5.2/doc/html/hc/hc_city_block_distance.htm

[7] https://en.wikipedia.org/wiki/Jaccard_index

[8] https://en.wikipedia.org/wiki/Canberra_distance

[9] https://en.wikipedia.org/wiki/Euclidean_distance

[10] https://en.wikipedia.org/wiki/Minkowski_distance

[11] https://en.wikipedia.org/wiki/Stop_words

[12] http://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/

[13]https://github.com/abhishekkrthakur/is_that_a_duplicate_quora_question/blob/master/feature_engineering.py

[14] https://developers.google.com/machine-learning/crash-course/embeddings/translating-to-a-lower-dimensional-space

[15] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.SGDClassifier.html

[16] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[17] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html

[18] https://xgboost.readthedocs.io/en/latest/python/python_api.html

[19] http://rasbt.github.io/mlxtend/user_guide/classifier/StackingCVClassifier/

[20] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.StackingClassifier.html

[21] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

[22] https://scikit-learn.org/stable/modules/generated/sklearn.calibration.CalibratedClassifierCV.html

[23] https://www.itl.nist.gov/div898/handbook/eda/section3/eda35b.htm

[24] https://github.com/tqdm/tqdm

[25] https://www.quora.com/How-many-questions-have-been-asked-on-Quora-1

[26] https://www.kaggle.com/c/quora-question-pairs

[27] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a 'siamese' time delay neural network," in *Proc. 6th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, San Francisco, CA, USA, 1993, pp. 737_744.

[28] B. N. Patro, V. K. Kurmi, S. Kumar, and V. P. Namboodiri, "Learning semantic sentence embeddings using sequential pair-wise discriminator," *CoRR*, vol. abs/1806.00807, pp. 1_15, Mar. 2018.

[29] J. Mueller and A. Thyagarajan, "Siamese recurrent architectures for learning sentence similarity," in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 2786_2792.

[30] M. Tsubaki, K. Duh, M. Shimbo, and Y. Matsumoto, "Non-linear similarity learning for compositionality," in *Proc. 13th AAAI Conf. Artif. Intell.*, 2016, pp. 2828_2834.

[31] B. Rychalska, K. Pakulska, K. Chodorowska, W. Walczak, and P. Andruszkiewicz, "Samsung Poland NLP team at SemEval-2016
task 1: Necessity for diversity; combining recursive autoencoders, WordNet and ensemble methods to measure semantic similarity," in *Proc.*

10th Int. Workshop Semantic Eval. (SemEval), San Diego, CA, USA, 2016, pp. 602_608.

[32] R. Johnson and T. Zhang, "Supervised and semi-supervised text categorization using LSTM for region embeddings," in *Proc. ICML*, 2016, pp. 1_9.

[33] D. Tang, B. Qin, and T. Liu, "Document modeling with gated recurrent neural network for sentiment classi_cation," in *Proc. Conf. Empir-ical Methods Natural Lang. Process.*, Lisbon, Portugal, Sep. 2015, pp. 1422_1432.

[34] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," in *Proc. 53rd Annu. Meeting Assoc. Comput. Linguistics 7th Int. Joint Conf. Natural Lang. Process.*, Beijing, China, vol. 1, Jul. 2015, pp. 1556_1566.

[35] R. Kiros, Y. Zhu, R. Salakhutdinov, R. S. Zemel, A. Torralba, R. Urtasun, and S. Fidler, "Skip-thought vectors," *CoRR*, vol. abs/1506.06726, pp. 1_11, Jun. 2015.