

A Novel Approach for Question Pair Similarity Identification

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, West Bengal

formerly known as **West Bengal University of Technology**
In Pursuit of Knowledge and Excellence

Maulana Abul Kalam Azad University of Technology

BF 142, BF Block, Sector 1, Kolkata, West Bengal 700064

May 2020

A Novel Approach for Question Pair Similarity Identification



**INSTITUTE
OF ENGINEERING & MANAGEMENT**
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 “**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.(Dr.) 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

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

A Novel Approach for Question Pair Similarity Identification


INSTITUTE OF ENGINEERING & MANAGEMENT



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: 08/07/2020

A Novel Approach for Question Pair Similarity Identification

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.

A Novel Approach for Question Pair Similarity Identification

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.

A Novel Approach for Question Pair Similarity Identification

Contents

List of Figures.....	vii
List of Tables.....	viii
1. Introduction.....	1
2. Literature Survey.....	2
2.1 Related Works.....	2
2.2 Our Approach.....	3
3. Dataset.....	5
4. Splitting the data into train and test dataset.....	6
5. Feature Extraction.....	7
5.1 Basic Features.....	7
5.2 NLP and Fuzzy Features.....	8
5.3 Word Embedding.....	13
6. Fixing the Missing, Indefinite values	18
7. K-Fold Cross Validation with Log Loss and Accuracy.....	19
7.1 Logistic Regression with SGD.....	19
7.2 Linear SVM with SGD and Hinge Loss.....	22
7.3 Random Forest Classifier.....	25
7.4 Extra Tree Classifier.....	27
7.5 Gradient Boost Decision Tree	29
7.6 Stacking Classifier.....	30
7.7 Adaptive Boosting.....	31
7.8 Conclusion after Cross Validation.....	31
8. Fitting Different Machine Learning Models and Testing Accuracy.....	32
8.1 Random Forest Classifier (Bagging) with Log Loss & Accuracy.....	32
8.2 Gradient Boost Decision Tree (Boosting) with Log Loss & Accuracy...	35
8.3 Stacking Classifier with Log Loss & Accuracy	37
9. Testing with Real Life Example.....	39
9.1 Predicting with Logistic Regression and SGD.....	40
9.2 Predicting with Random Forest Classifier.....	41
9.3 Predicting with Gradient Boost Decision Tree.....	42
9.4 Predicting with Stacking Classifier.....	43
9.5 Conclusion after Testing.....	44
10. Conclusion and Recommendation.....	45
11. Bibliography.....	46

A Novel Approach for Question Pair Similarity Identification

List of Figures

1. Our Solution Approach in Diagram.....	4
2. word_share vs is_dupliacte violin plot and pdf.....	8
3. fuzz_ratio vs is_dupliacte violin plot and pdf.....	10
4. fuzz_partial_ratio vs is_duplicate violin plot and pdf.....	10
5. fuzz_token_sort_ratio vs is_duplicate violin plot and pdf.....	11
6. Pair plot of different basic and fuzzy features.....	12
7. word_mover_distance.....	14
8. word_mover_distance.....	14
9. word_mover_distance.....	15
10. word_mover_distance.....	15
11. Word to Vector Conversion Example.....	17
12. Random Forest Classifier Confusion Matrix Result.....	34
13. Gradient Boost Decision Tree Confusion Matrix Result.....	36
14. Stacking Classifier Confusion Matrix Result.....	38

A Novel Approach for Question Pair Similarity Identification

List of Tables

1	Word Mover Distance.....	16
2	Normalized Word Mover Distance	16
3	Logistic Regression Prediction Result.....	40
4	Random Forest Prediction Result.....	41
5	Gradient Boost Decision Tree Prediction Result.....	42
6	Stacking Classifier Prediction Result.....	43

A Novel Approach for Question Pair Similarity Identification

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>

A Novel Approach for Question Pair Similarity Identification

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].

A Novel Approach for Question Pair Similarity Identification

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.

After that, we split our dataset into train and test data in the ratio of 70:30.

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 metrics 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 train dataset and create a final dataset for cross validation and 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.

A Novel Approach for Question Pair Similarity Identification

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 metrics. 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 for cross validation to find out which algorithm(s) are performing well for this dataset according to our performance metrics.

Then, we apply the same preprocessing techniques for the test dataset and prepare final test dataset and then measure test accuracy score and test log loss against train accuracy score and train log loss and observe whether they are overfitting or underfitting or performing correctly.

Finally, we deploy the machine learning models which are working best among all and then take user input Question Pairs and predict duplicate(1) and non-duplicate(0) to see how are they classifying the Question Pairs in real life.

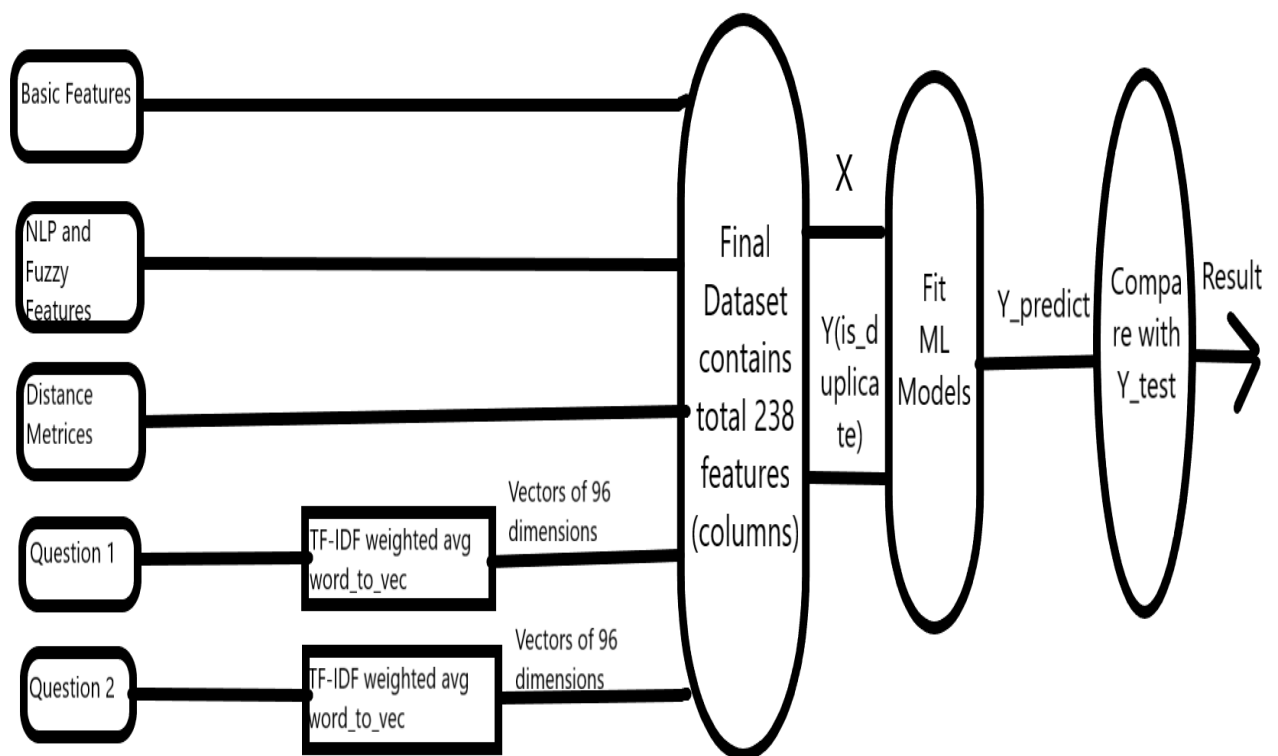


Figure 1 Our Solution Approach in Diagram

A Novel Approach for Question Pair Similarity Identification

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?

A Novel Approach for Question Pair Similarity Identification

4

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 considered 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, X_test, y_train, y_test = train_test_split(X, y, stratify=y, test_size=0.3, random_state=13)
```

A Novel Approach for Question Pair Similarity Identification

5

Feature Extraction

5.1 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

A Novel Approach for Question Pair Similarity Identification

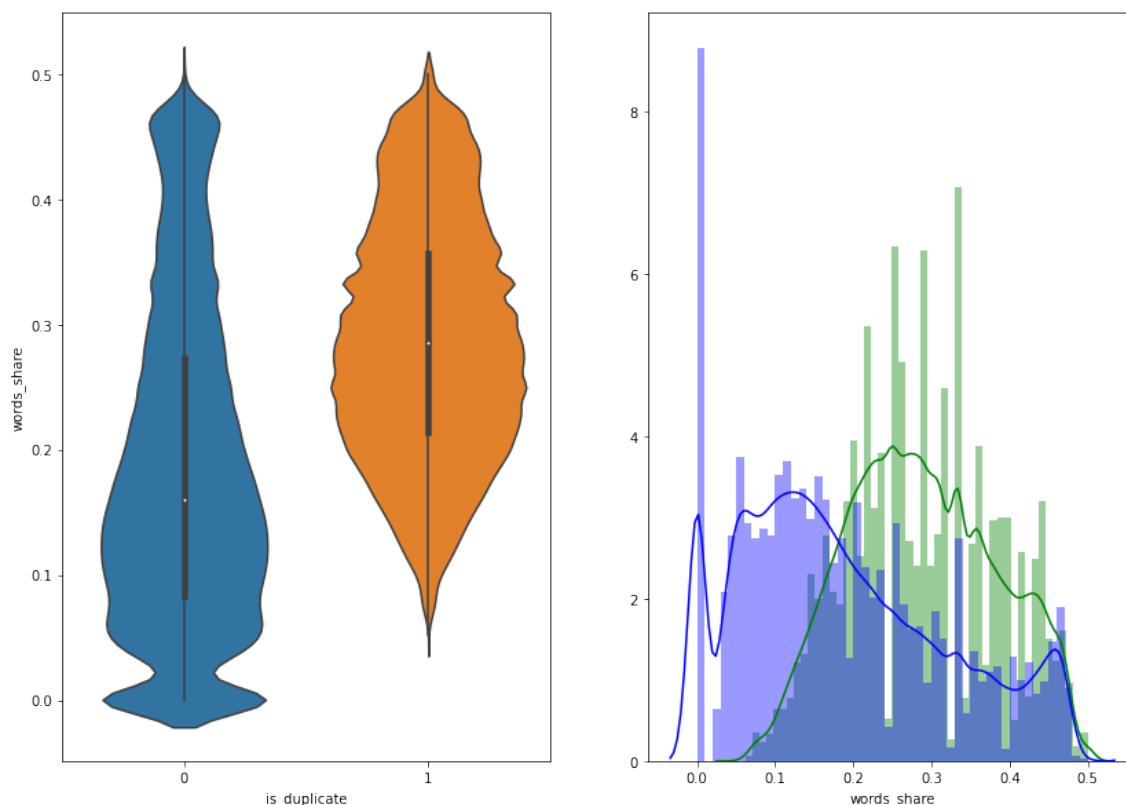


Figure 2 word_share vs is_duplicate violin plot and pdf

5.2 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
$$\text{cwc_min} = \text{common_word_count} / (\min(\text{len}(\text{q1_words}), \text{len}(\text{q2_words})))$$
- **Common-word-count-maximum:** Ratio of common_word_count to max length of word count of Q1 and Q2
$$\text{cwc_max} = \text{common_word_count} / (\max(\text{len}(\text{q1_words}), \text{len}(\text{q2_words})))$$
- **Common-stop-count-minimum:** Ratio of common_stop_count to min length of stop count of Q1 and Q2
$$\text{csc_min} = \text{common_stop_count} / (\min(\text{len}(\text{q1_stops}), \text{len}(\text{q2_stops})))$$

A Novel Approach for Question Pair Similarity Identification

- **Common-stop-count-maximum:** Ratio of common_stop_count to max length of stop count of Q1 and Q2
$$\text{csc_max} = \text{common_stop_count} / (\max(\text{len}(\text{q1_stops}), \text{len}(\text{q2_stops})))$$
- **Common-token-count-minimum:** Ratio of common_token_count to min length of token count of Q1 and Q2
$$\text{ctc_min} = \text{common_token_count} / (\min(\text{len}(\text{q1_tokens}), \text{len}(\text{q2_tokens})))$$
- **Common-token-count-maximum:** Ratio of common_token_count to max length of token count of Q1 and Q2
$$\text{ctc_max} = \text{common_token_count} / (\max(\text{len}(\text{q1_tokens}), \text{len}(\text{q2_tokens})))$$
- **Last-word-equivalent:** Check if Last word of both questions is equal or not. Returns Boolean value
$$\text{last_word_eq} = \text{int}(\text{q1_tokens}[-1] == \text{q2_tokens}[-1])$$
- **First-word-equivalent:** Check if First word of both questions is equal or not. Returns Boolean value
$$\text{first_word_eq} = \text{int}(\text{q1_tokens}[0] == \text{q2_tokens}[0])$$
- **absolute-length-difference:** Abs. length difference
$$\text{abs_len_diff} = \text{abs}(\text{length}(\text{q1_tokens}) - \text{length}(\text{q2_tokens}))$$
- **mean-length:** Average Token Length of both Questions
$$\text{mean_length} = (\text{length}(\text{q1_tokens}) + \text{length}(\text{q2_tokens})) / 2$$
- **longest-substring-ratio:** Ratio of length longest common substring to min length of token count of Q1 and Q2
$$\text{longest-substring-ratio} = \text{length}(\text{longest common substring}) / (\min(\text{length}(\text{q1_tokens}), \text{length}(\text{q2_tokens})))$$

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]

A Novel Approach for Question Pair Similarity Identification

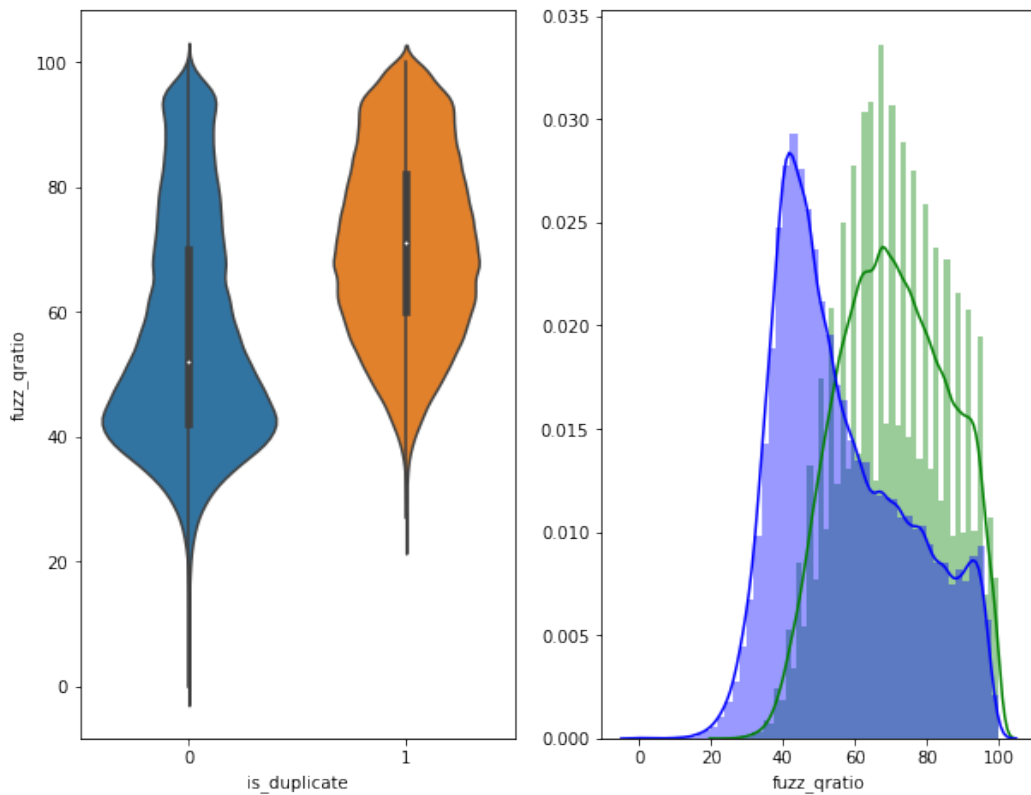


Figure 3 *fuzz_ratio* vs *is_duplicate* violin plot and pdf

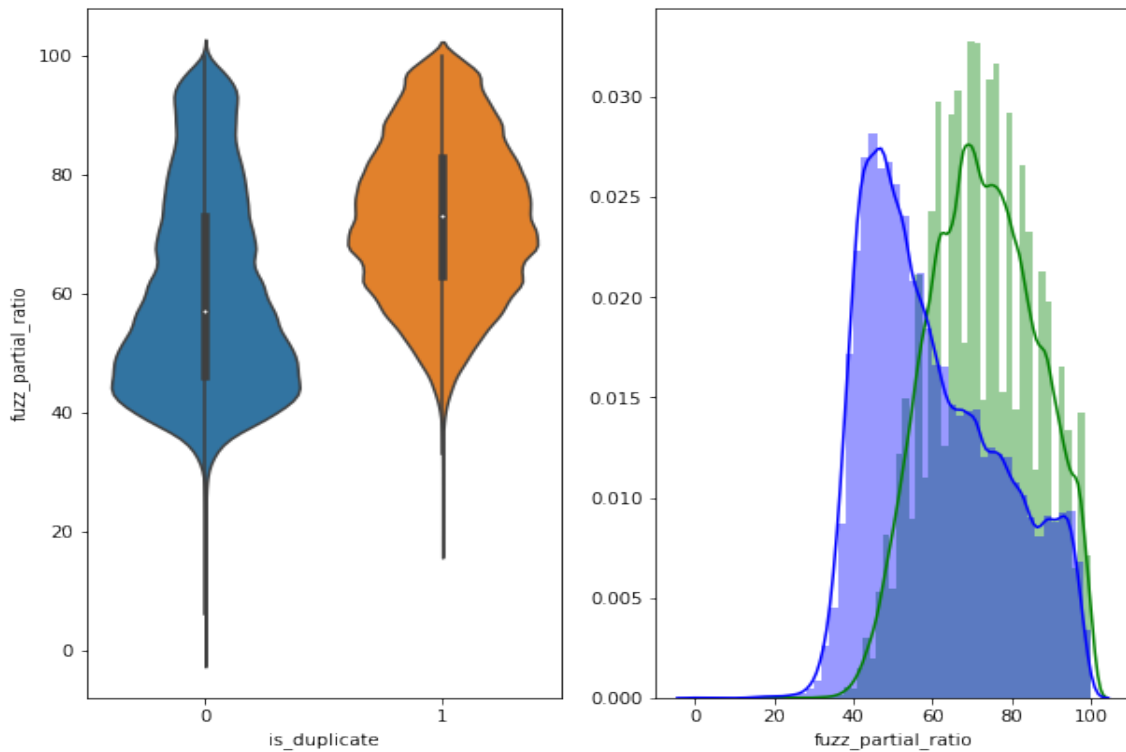


Figure 4 *fuzz_partial_ratio* vs *is_duplicate* violin plot and pdf

A Novel Approach for Question Pair Similarity Identification

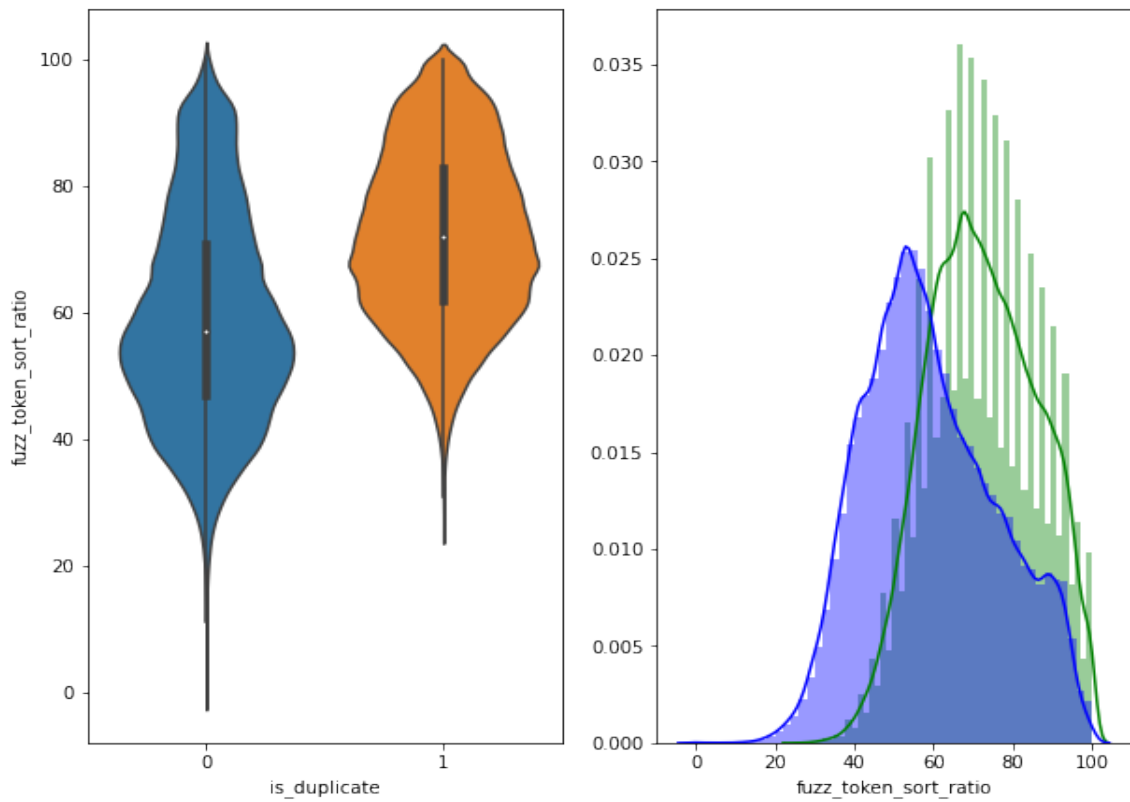


Figure 5 *fuzz_token_sort_ratio* vs *is_duplicate* violin plot and pdf

A Novel Approach for Question Pair Similarity Identification

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

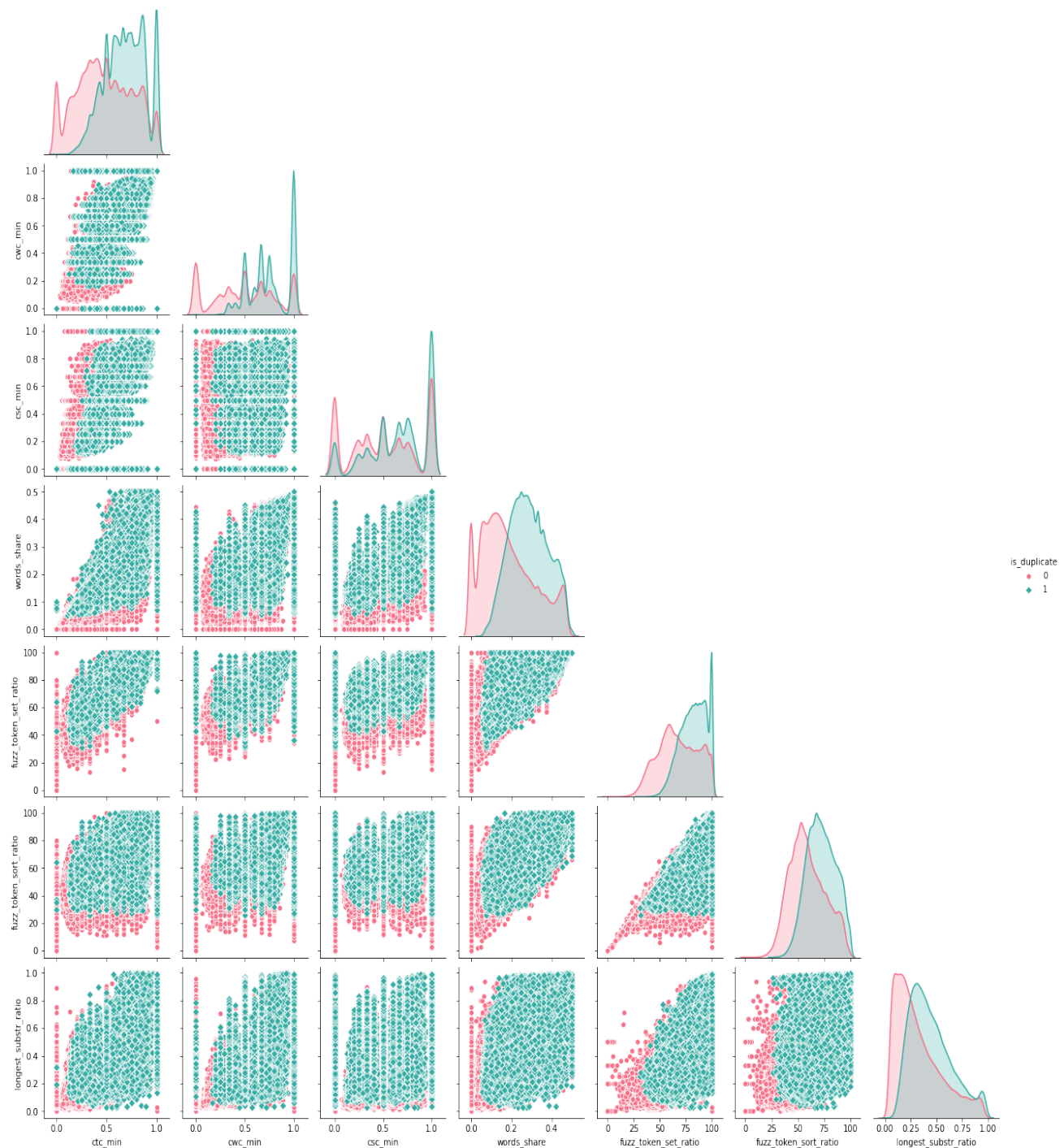


Figure 6 Pair plot of different basic and fuzzy features

A Novel Approach for Question Pair Similarity Identification

5.3 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-negative300.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)
```

A Novel Approach for Question Pair Similarity Identification

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

A Novel Approach for Question Pair Similarity Identification

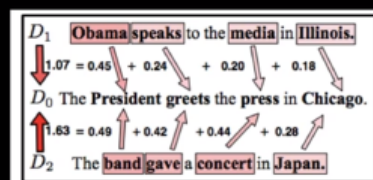
document distance = sum(word distance) = document similarity



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

Figure 9 word_mover_distance

document distance = sum(word distance) = document dissimilarity



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.

https://www.youtube.com/watch?v=nX1g_wPSYOI

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

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]
```

A Novel Approach for Question Pair Similarity Identification

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('GoogleNews-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.nan_to_num(q1_vectors), np.nan_to_num(q2_vectors))] [13]
```


A Novel Approach for Question Pair Similarity Identification

We 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)] [13]
```

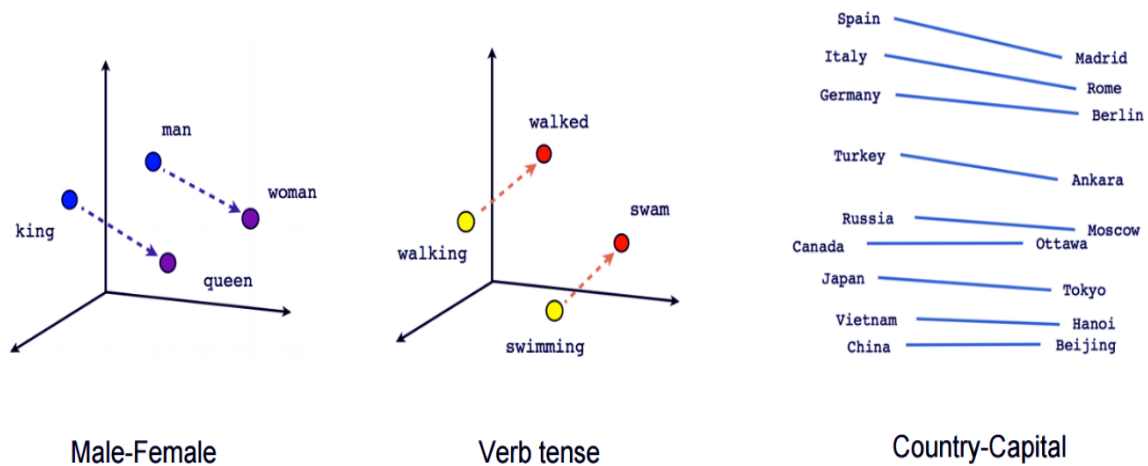


Figure 11 Word to Vector Conversion Example[14]

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

A Novel Approach for Question Pair Similarity Identification

6

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)]
```

A Novel Approach for Question Pair Similarity Identification

7

K-Fold Cross Validation with Log Loss and Accuracy

We run K-Fold Cross Validation for the following Machine Learning Models to test which models are going to perform well in this dataset.

```
from sklearn.model_selection import StratifiedKFold
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import cross_val_predict
from sklearn.model_selection import cross_validate
k_fold = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)
```

7.1 Cross Validation with Logistic Regression & SGD[15]

```
alpha = np.random.uniform(0.00005, 0.00035, 7)
alpha = np.round(alpha, 7)
alpha.sort()

for i in alpha:
    clf = SGDClassifier(alpha=i, penalty='l2', loss='log', random_state=42)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    scoring=('accuracy', 'neg_log_loss')
    scores = cross_validate(sig_clf, X_train_final, y=y_train_final, cv=k_fold, n_jobs=-1, scoring=scoring, return_train_score=True)
    #print(scores['train_accuracy'])
    #print(scores['test_accuracy'])
    print('For alpha = ', i, ', mean of 10-fold cross_validation TRAIN accuracy score is = ', (round(np.mean(scores['train_accuracy']), 5)), '\n')
    print('For alpha = ', i, ', mean of 10-fold cross_validation TEST accuracy score is = ', (round(np.mean(scores['test_accuracy']), 5)), '\n')
    print('\n')
    print('For alpha = ', i, ', mean of 10-fold cross_validation TRAIN log loss score is = ', abs(round(np.mean(scores['train_neg_log_loss']), 5)), '\n')
    print('For alpha = ', i, ', mean of 10-fold cross_validation TEST log loss score is = ', abs(round(np.mean(scores['test_neg_log_loss']), 5)), '\n')
```

A Novel Approach for Question Pair Similarity Identification

The result is as follows:

For alpha = 0.0001163 , mean of 10-fold cross_validation TRAIN accuracy score is = 0.79276

For alpha = 0.0001163 , mean of 10-fold cross_validation TEST accuracy score is = 0.79228

For alpha = 0.0001163 , mean of 10-fold cross_validation TRAIN log loss score is = 0.40855

For alpha = 0.0001163 , mean of 10-fold cross_validation TEST log loss score is = 0.40923

For alpha = 0.0001912 , mean of 10-fold cross_validation TRAIN accuracy score is = 0.79133

For alpha = 0.0001912 , mean of 10-fold cross_validation TEST accuracy score is = 0.79098

For alpha = 0.0001912 , mean of 10-fold cross_validation TRAIN log loss score is = 0.4101

For alpha = 0.0001912 , mean of 10-fold cross_validation TEST log loss score is = 0.41102

For alpha = 0.0001991 , mean of 10-fold cross_validation TRAIN accuracy score is = 0.79183

For alpha = 0.0001991 , mean of 10-fold cross_validation TEST accuracy score is = 0.79122

For alpha = 0.0001991 , mean of 10-fold cross_validation TRAIN log loss score is = 0.40943

For alpha = 0.0001991 , mean of 10-fold cross_validation TEST log loss score is = 0.41022

For alpha = 0.0002076 , mean of 10-fold cross_validation TRAIN accuracy score is = 0.79149

A Novel Approach for Question Pair Similarity Identification

For alpha = 0.0002076 , mean of 10-fold cross_validation TEST accuracy score is = 0.79071

For alpha = 0.0002076 , mean of 10-fold cross_validation TRAIN log loss score is = 0.40997

For alpha = 0.0002076 , mean of 10-fold cross_validation TEST log loss score is = 0.4107

For alpha = 0.0002503 , mean of 10-fold cross_validation TRAIN accuracy score is = 0.78967

For alpha = 0.0002503 , mean of 10-fold cross_validation TEST accuracy score is = 0.78896

For alpha = 0.0002503 , mean of 10-fold cross_validation TRAIN log loss score is = 0.41134

For alpha = 0.0002503 , mean of 10-fold cross_validation TEST log loss score is = 0.41213

For alpha = 0.000293 , mean of 10-fold cross_validation TRAIN accuracy score is = 0.78982

For alpha = 0.000293 , mean of 10-fold cross_validation TEST accuracy score is = 0.78907

For alpha = 0.000293 , mean of 10-fold cross_validation TRAIN log loss score is = 0.41088

For alpha = 0.000293 , mean of 10-fold cross_validation TEST log loss score is = 0.41171

A Novel Approach for Question Pair Similarity Identification

7.2 Cross Validation with Linear SVM & SGD[15]

```
alpha = np.random.uniform(0.00005,0.00035,7)
alpha = np.round(alpha,7)
alpha.sort()

for i in alpha:
    clf = SGDClassifier(alpha=i, penalty='l2', loss='hinge', random_state=42)
    sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
    scoring= ('accuracy', 'neg_log_loss')
    scores = cross_validate(sig_clf, X_train_final, y=y_train_final, cv=k_fold, n_jobs=-1, scoring=scoring, return_train_score=True)
    #print(scores['train_accuracy'])
    #print(scores['test_accuracy'])
    print('For alpha = ', i, ', mean of 10-fold cross_validation TRAIN accuracy score is = ', (round(np.mean(scores['train_accuracy']), 5)), '\n')
    print('For alpha = ', i, ', mean of 10-fold cross_validation TEST accuracy score is = ', (round(np.mean(scores['test_accuracy']), 5)), '\n')
    print('\n')
    print('For alpha = ', i, ', mean of 10-fold cross_validation TRAIN log loss score is = ', abs(round(np.mean(scores['train_neg_log_loss']), 5)), '\n')
    print('For alpha = ', i, ', mean of 10-fold cross_validation TEST log loss score is = ', abs(round(np.mean(scores['test_neg_log_loss']), 5)), '\n')
```

The result is as follows:

For alpha = 7.3e-05 , mean of 10-fold cross_validation TRAIN accuracy score is = 0.78967

For alpha = 7.3e-05 , mean of 10-fold cross_validation TEST accuracy score is = 0.78897

For alpha = 7.3e-05 , mean of 10-fold cross_validation TRAIN log loss score is = 0.41646

For alpha = 7.3e-05 , mean of 10-fold cross_validation TEST log loss score is = 0.41723

For alpha = 7.76e-05 , mean of 10-fold cross_validation TRAIN accuracy score is = 0.7907

For alpha = 7.76e-05 , mean of 10-fold cross_validation TEST accuracy score is = 0.79035

A Novel Approach for Question Pair Similarity Identification

For alpha = 7.76e-05 , mean of 10-fold cross_validation TRAIN log loss score is = 0.41408

For alpha = 7.76e-05 , mean of 10-fold cross_validation TEST log loss score is = 0.41492

For alpha = 0.0001373 , mean of 10-fold cross_validation TRAIN accuracy score is = 0.79344

For alpha = 0.0001373 , mean of 10-fold cross_validation TEST accuracy score is = 0.79288

For alpha = 0.0001373 , mean of 10-fold cross_validation TRAIN log loss score is = 0.41162

For alpha = 0.0001373 , mean of 10-fold cross_validation TEST log loss score is = 0.41226

For alpha = 0.0001894 , mean of 10-fold cross_validation TRAIN accuracy score is = 0.79459

For alpha = 0.0001894 , mean of 10-fold cross_validation TEST accuracy score is = 0.79422

For alpha = 0.0001894 , mean of 10-fold cross_validation TRAIN log loss score is = 0.41124

For alpha = 0.0001894 , mean of 10-fold cross_validation TEST log loss score is = 0.41201

For alpha = 0.0002552 , mean of 10-fold cross_validation TRAIN accuracy score is = 0.7947

For alpha = 0.0002552 , mean of 10-fold cross_validation TEST accuracy score is = 0.79401

For alpha = 0.0002552 , mean of 10-fold cross_validation TRAIN log loss score is = 0.41109

A Novel Approach for Question Pair Similarity Identification

For alpha = 0.0002552 , mean of 10-fold cross_validation TEST log loss score is = 0.41184

For alpha = 0.0002587 , mean of 10-fold cross_validation TRAIN accuracy score is = 0.79479

For alpha = 0.0002587 , mean of 10-fold cross_validation TEST accuracy score is = 0.79397

For alpha = 0.0002587 , mean of 10-fold cross_validation TRAIN log loss score is = 0.41129

For alpha = 0.0002587 , mean of 10-fold cross_validation TEST log loss score is = 0.41202

A Novel Approach for Question Pair Similarity Identification

7.3 Cross Validation with Random Forest Classifier[16]

```
from sklearn.ensemble import RandomForestClassifier as RFC
estimators = [75,100,150,200,300,400]

for i in estimators:
    clf = RFC(n_estimators=i,max_depth=11,n_jobs=-1)#low bias high variance model, as depth increases variance increases. while bagging the variance will come down automatically in fact very low. n_jobs=-1 to parallelize the task into cpu cores
    #class_weight={0: 1, 1: 1.75}
    scoring=('accuracy', 'neg_log_loss')
    scores = cross_validate(clf, X_train_final, y=y_train_final, cv=k_fold_five, n_jobs=-1, scoring=scoring, return_train_score=True)
    #print(scores['train_accuracy'])
    #print(scores['test_accuracy'])
    print('For alpha = ', i, ', mean of 5-fold cross_validation TRAIN accuracy score is = ', (round(np.mean(scores['train_accuracy']), 5)), '\n')
    print('For alpha = ', i, ', mean of 5-fold cross_validation TEST accuracy score is = ', (round(np.mean(scores['test_accuracy']), 5)), '\n')
    print('\n')
    print('For alpha = ', i, ', mean of 5-fold cross_validation TRAIN log loss score is = ', abs(round(np.mean(scores['train_neg_log_loss']), 5)), '\n')
    print('For alpha = ', i, ', mean of 5-fold cross_validation TEST log loss score is = ', abs(round(np.mean(scores['test_neg_log_loss']), 5)), '\n')
```

The result is as follows:

```
For alpha = 75 , mean of 5-fold cross_validation TRAIN accuracy score is = 0.83711
```

```
For alpha = 75 , mean of 5-fold cross_validation TEST accuracy score is = 0.8215
```

```
For alpha = 75 , mean of 5-fold cross_validation TRAIN log loss score is = 0.36127
```

```
For alpha = 75 , mean of 5-fold cross_validation TEST log loss score is = 0.38037
```

```
-----
-----
```

```
For alpha = 100 , mean of 5-fold cross_validation TRAIN accuracy score is = 0.83682
```

```
For alpha = 100 , mean of 5-fold cross_validation TEST accuracy score is = 0.82081
```

A Novel Approach for Question Pair Similarity Identification

For alpha = 100 , mean of 5-fold cross_validation TRAIN log loss score
is = 0.36304

For alpha = 100 , mean of 5-fold cross_validation TEST log loss score
is = 0.38212

For alpha = 150 , mean of 5-fold cross_validation TRAIN accuracy score
is = 0.83775

For alpha = 150 , mean of 5-fold cross_validation TEST accuracy score
is = 0.82201

For alpha = 150 , mean of 5-fold cross_validation TRAIN log loss score
is = 0.36098

For alpha = 150 , mean of 5-fold cross_validation TEST log loss score
is = 0.38003

For alpha = 200 , mean of 5-fold cross_validation TRAIN accuracy score
is = 0.83698

For alpha = 200 , mean of 5-fold cross_validation TEST accuracy score
is = 0.82102

For alpha = 200 , mean of 5-fold cross_validation TRAIN log loss score
is = 0.36181

For alpha = 200 , mean of 5-fold cross_validation TEST log loss score
is = 0.38086

A Novel Approach for Question Pair Similarity Identification

7.4 Cross Validation with Extra Tree Classifier[17]

```
from sklearn.ensemble import ExtraTreesClassifier as EXC
estimators = [75,100,150,200,300]

for i in estimators:
    clf = EXC(n_estimators=i,max_depth=11,n_jobs=-1
              scoring=('accuracy', 'neg_log_loss')
              scores = cross_validate(clf, X_train_final, y=y_train_final, cv=k_fold_five, n_jobs=-1, scoring=scoring, return_train_score=True)
    #print(scores['train_accuracy'])
    #print(scores['test_accuracy'])
    print('For estimator = ', i, ', mean of 5-fold cross_validation TRAIN accuracy score is = ', (round(np.mean(scores['train_accuracy']), 5)), '\n')
    print('For estimator = ', i, ', mean of 5-fold cross_validation TEST accuracy score is = ', (round(np.mean(scores['test_accuracy']), 5)), '\n')
    print('\n')
    print('For estimator = ', i, ', mean of 5-fold cross_validation TRAIN log loss score is = ', abs(round(np.mean(scores['train_neg_log_loss']), 5)), '\n')
    print('For estimator = ', i, ', mean of 5-fold cross_validation TEST log loss score is = ', abs(round(np.mean(scores['test_neg_log_loss']), 5)), '\n')
```

The result is as follows:

```
For estimator = 75 , mean of 5-fold cross_validation TRAIN accuracy score is = 0.79265
```

```
For estimator = 75 , mean of 5-fold cross_validation TEST accuracy score is = 0.78398
```

```
For estimator = 75 , mean of 5-fold cross_validation TRAIN log loss score is = 0.45233
```

```
For estimator = 75 , mean of 5-fold cross_validation TEST log loss score is = 0.45816
```

```
-----
-----
```

```
For estimator = 100 , mean of 5-fold cross_validation TRAIN accuracy score is = 0.79091
```

```
For estimator = 100 , mean of 5-fold cross_validation TEST accuracy score is = 0.78257
```

A Novel Approach for Question Pair Similarity Identification

For estimator = 100 , mean of 5-fold cross_validation TRAIN log loss score is = 0.45293

For estimator = 100 , mean of 5-fold cross_validation TEST log loss score is = 0.45863

For estimator = 150 , mean of 5-fold cross_validation TRAIN accuracy score is = 0.79045

For estimator = 150 , mean of 5-fold cross_validation TEST accuracy score is = 0.78197

For estimator = 150 , mean of 5-fold cross_validation TRAIN log loss score is = 0.45377

For estimator = 150 , mean of 5-fold cross_validation TEST log loss score is = 0.45949

For estimator = 200 , mean of 5-fold cross_validation TRAIN accuracy score is = 0.79207

For estimator = 200 , mean of 5-fold cross_validation TEST accuracy score is = 0.78288

For estimator = 200 , mean of 5-fold cross_validation TRAIN log loss score is = 0.45227

For estimator = 200 , mean of 5-fold cross_validation TEST log loss score is = 0.45801

For estimator = 300 , mean of 5-fold cross_validation TRAIN accuracy score is = 0.79193

For estimator = 300 , mean of 5-fold cross_validation TEST accuracy score is = 0.78327

For estimator = 300 , mean of 5-fold cross_validation TRAIN log loss score is = 0.45267

For estimator = 300 , mean of 5-fold cross_validation TEST log loss score is = 0.45843

A Novel Approach for Question Pair Similarity Identification

7.5 Cross Validation with Gradient Boost Decision Tree[18]

```
import xgboost as xgb
clf = xgb.XGBClassifier(max_depth=11, n_estimators=80, learning_rate=0.08,
    colsample_bytree=.7, gamma=0, reg_alpha=4, objective='binary:logistic',
    eta=0.3, silent=1, subsample=0.8)
scoring=('accuracy', 'neg_log_loss')
scores = cross_validate(clf, X_train_final, y=y_train_final, cv=k_fold_
    five, n_jobs=-1, scoring=scoring, return_train_score=True)
#print(scores['train_accuracy'])
#print(scores['test_accuracy'])
print('Mean of 5-fold cross_validation TRAIN accuracy score is = ', (ro
    und(np.mean(scores['train_accuracy']), 5)), '\n')
print('Mean of 5-fold cross_validation TEST accuracy score is = ', (rou
    nd(np.mean(scores['test_accuracy']), 5)), '\n')
print('\n')
print('Mean of 5-fold cross_validation TRAIN log loss score is = ', abs
    (round(np.mean(scores['train_neg_log_loss']), 5)), '\n')
print('Mean of 5-fold cross_validation TEST log loss score is = ', abs(
    round(np.mean(scores['test_neg_log_loss']), 5)), '\n')
```

The result is as follows:

Mean of 5-fold cross_validation TRAIN accuracy score is = 0.89268

Mean of 5-fold cross_validation TEST accuracy score is = 0.8463

Mean of 5-fold cross_validation TRAIN log loss score is = 0.25871

Mean of 5-fold cross_validation TEST log loss score is = 0.31634

A Novel Approach for Question Pair Similarity Identification

7.6 Cross Validation with Stacking Classifier[19][20]

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.ensemble import StackingClassifier
import xgboost as xgb

estimators = [('rf', RandomForestClassifier(n_estimators=200, max_depth=11, random_state=42)), ('sgc', SGDCClassifier(alpha=0.0001291, penalty='l2', loss='hinge', random_state=42)), ('sgdc', (SGDCClassifier(alpha=0.0002032, penalty='l2', loss='log', random_state=42)))]
clf = StackingClassifier(estimators=estimators, final_estimator=xgb.XGBClassifier(max_depth=11, learning_rate=0.08, n_estimators=400, n_jobs=-1, subsample=0.85, colsample_bytree=0.85))
scoring=('accuracy', 'neg_log_loss')
scores = cross_validate(clf, X_train_final, y=y_train_final, cv=k_fold_five, n_jobs=-1, scoring=scoring, return_train_score=True)
#print(scores['train_accuracy'])
#print(scores['test_accuracy'])
print('Mean of 5-fold cross_validation TRAIN accuracy score is = ', (round(np.mean(scores['train_accuracy']), 5)), '\n')
print('Mean of 5-fold cross_validation TEST accuracy score is = ', (round(np.mean(scores['test_accuracy']), 5)), '\n')
print('\n')
print('Mean of 5-fold cross_validation TRAIN log loss score is = ', abs(round(np.mean(scores['train_neg_log_loss']), 5)), '\n')
print('Mean of 5-fold cross_validation TEST log loss score is = ', abs(round(np.mean(scores['test_neg_log_loss']), 5)), '\n')
```

The result is as follows:

Mean of 5-fold cross_validation TRAIN accuracy score is = 0.83485

Mean of 5-fold cross_validation TEST accuracy score is = 0.82256

Mean of 5-fold cross_validation TRAIN log loss score is = 0.33037

Mean of 5-fold cross_validation TEST log loss score is = 0.35392

A Novel Approach for Question Pair Similarity Identification

7.7 Cross Validation with Adaptive Boosting[21]

```
clf = abc(n_estimators=75, learning_rate=0.08, algorithm='SAMME.R', random_state=42)
scoring= ('accuracy', 'neg_log_loss')
scores = cross_validate(clf, X_train_final, y=y_train_final, cv=k_fold_five, n_jobs=-1, scoring=scoring, return_train_score=True)
#print(scores['train_accuracy'])
#print(scores['test_accuracy'])
print('Mean of 5-fold cross_validation TRAIN accuracy score is = ', (round(np.mean(scores['train_accuracy']), 5)), '\n')
print('Mean of 5-fold cross_validation TEST accuracy score is = ', (round(np.mean(scores['test_accuracy']), 5)), '\n')
print('\n')
print('Mean of 5-fold cross_validation TRAIN log loss score is = ', abs(round(np.mean(scores['train_neg_log_loss']), 5)), '\n')
print('Mean of 5-fold cross_validation TEST log loss score is = ', abs(round(np.mean(scores['test_neg_log_loss']), 5)), '\n')
```

The result is as follows:

Mean of 5-fold cross_validation TRAIN accuracy score is = 0.77419

Mean of 5-fold cross_validation TEST accuracy score is = 0.77413

Mean of 5-fold cross_validation TRAIN log loss score is = 0.58505

Mean of 5-fold cross_validation TEST log loss score is = 0.58508

7.8 Conclusion after Cross Validation

We can see that the train log loss score and test log loss score are quite similar and so as train accuracy score and test accuracy score. So, we can say that none of the models are overfitting or underfitting. They are just working correctly.

Among all the Machine Learning models applied above, the Random Forest, Gradient Boost Decision Tree(XgBoost) and Stacking Classifier(mlxtend) are giving the best cross validation accuracy score and log loss score. So, in the next part, we'll train our full training data only with these three algorithms and then verify how test data is performing against training results.

A Novel Approach for Question Pair Similarity Identification

8

Fitting different Machine Learning Models and Testing Accuracy

8.1 Random Forest Classifier[16] (Bagging) with Log Loss & Accuracy

We apply Random Forest Classifier[16] with max_depth=11 and estimators = [75,100,150,200,300,400] and the performance metric(s) is Log Loss and Accuracy. We also plot confusion matrix to get True Positive, True Negative, False Positive, False Negative, Precision Matrix, Recall Matrix.

```
from sklearn.ensemble import RandomForestClassifier as RFC

estimators = [75,100,150,200,300,400]
test_scores = []
train_scores = []
for i in estimators:
    clf = RFC(n_estimators=i,max_depth=11,n_jobs=-1)
    clf.fit(X_train_final,y_train_final)
    predict_y_train_log_loss = clf.predict_proba(X_train_final)
    log_loss_train = log_loss(y_train_final, predict_y_train_log_loss,
eps=1e-15)
    train_scores.append(log_loss_train)

    predict_y_train_accuracy = clf.predict(X_train_final)
    accuracy_train = accuracy_score(y_train_final, predict_y_train_accu
racy, normalize=True, sample_weight=None)

    predict_y_test_log_loss = clf.predict_proba(X_test_final)
    log_loss_test = log_loss(y_test_final, predict_y_test_log_loss, eps
=1e-15)
    test_scores.append(log_loss_test)

    predict_y_test_accuracy = clf.predict(X_test_final)
    accuracy_test = accuracy_score(y_test_final, predict_y_test_accurac
y, normalize=True, sample_weight=None)
```


A Novel Approach for Question Pair Similarity Identification

```
print('estimators =',i,', Train Log Loss =',log_loss_train,', Test
Log Loss =',log_loss_test)
print('\n')
print('estimators =',i,', Train accuracy =',accuracy_train,', Test
accuracy =',accuracy_test)
print('\n-----
-----\n')
plt.plot(estimators,train_scores,label='Train Log Loss')
plt.plot(estimators,test_scores,label='Test Log Loss')
plt.xlabel('estimators')
plt.ylabel('Log Loss')
predicted_y =np.argmax(predict_y,axis=1)
print("Total number of data points in test data:", len(predicted_y))
print('\n')
print("Number of correctly classified points in test data = ", accuracy
_score(y_test_final, predict_y_test_accuracy, normalize=False))
print('\n-----
-----\n')
plot_confusion_matrix(y_test_final, predicted_y)
```

The Result came out as:

```
estimators = 75 , Train Log Loss = 0.36405628952583796 , Test Log Loss
= 0.4482741565083662 estimators = 75 , Train accuracy = 0.8333150300587
003 , Test accuracy = 0.7602264743563252 -----
-----
estimators = 100 , Train Log Loss = 0.364081158622155 , Test Log Loss =
0.4512752424670658 estimators = 100 , Train accuracy = 0.83471434492580
09 , Test accuracy = 0.7565648634128198 -----
-----
estimators = 150 , Train Log Loss = 0.3642655651906377 , Test Log Loss
= 0.44863101931269006 estimators = 150 , Train accuracy = 0.83404125676
18791 , Test accuracy = 0.7572178369219325 -----
-----
-- estimators = 200 , Train Log Loss = 0.36457575379920254 , Test Log L
oss = 0.4498438404386922 estimators = 200 , Train accuracy = 0.83448762
04916377 , Test accuracy = 0.758135306029673 -----
-----
---- estimators = 300 , Train Log Loss = 0.36431555931886833 , Test Log
Loss = 0.44712159018228653 estimators = 300 , Train accuracy = 0.835118
197824154 , Test accuracy = 0.7593668636607844 -----
-----
----- estimators = 400 , Train Log Loss = 0.36443428900082925 , Test L
og Loss = 0.4490843781341466 estimators = 400 , Train accuracy = 0.8351
359106705729 , Test accuracy = 0.7577716245815597 -----
-----
Total number of data points in test data: 120985 Number of correctly cl
assified points in test data = 91679
```

A Novel Approach for Question Pair Similarity Identification

Now, since the train log loss and test log loss are almost similar in addition to similarity of train accuracy score and test accuracy score, we can conclude that our Random Forest Classifier Model[16] is not overfitting or underfitting. It is working correctly.

The result of confusion matrix is as follows:

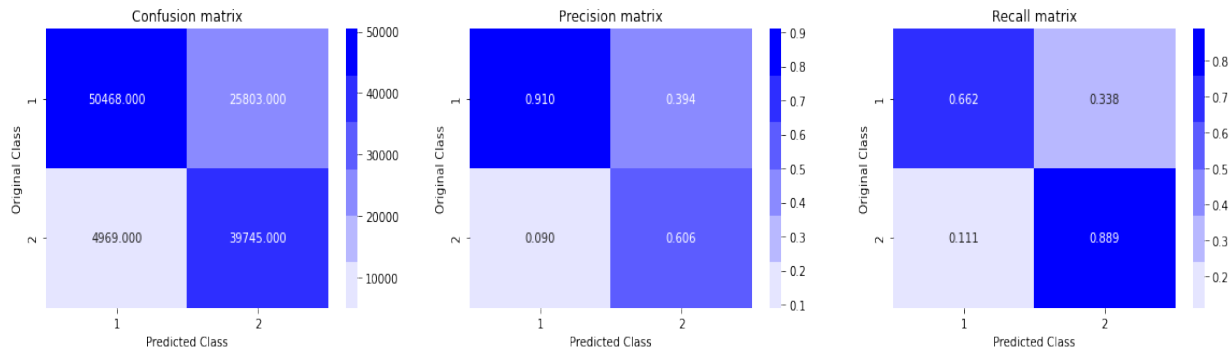


Figure 12 Random Forest Classifier Confusion Matrix Result

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

A Novel Approach for Question Pair Similarity Identification

8.2 Gradient Boost Decision Tree[18] (Boosting) with Log Loss & Accuracy

We apply Gradient Boost Decision Tree of XgBoost[18] Library with max_depth=8 and n_estimators = 400 and learning rate is 0.02 and the performance metric(s) is Log Loss and Accuracy. We also plot confusion matrix to get True Positive, True Negative, False Positive, False Negative, Precision Matrix, Recall Matrix.

```
import xgboost as xgb
clf = xgb.XGBClassifier(max_depth=8, n_estimators=400, learning_rate=0.02,
    colsample_bytree=.9, gamma=0, reg_alpha=4, objective='binary:logistic',
    eta=0.3, silent=1, subsample=0.9)
clf.fit(X_train_final, y_train_final)

predict_y_train_log_loss = clf.predict_proba(X_train_final)
print("The train log loss is:", log_loss(y_train_final, predict_y_train_log_loss,
    eps=1e-15))
print('\n')
predict_y_test_log_loss = clf.predict_proba(X_test_final)
print("The test log loss is:", log_loss(y_test_final, predict_y_test_log_loss,
    eps=1e-15))
print('\n-----\n')
predict_y_train_accuracy = clf.predict(X_train_final)
print("The train accuracy is:", accuracy_score(y_train_final, predict_y_train_accuracy,
    normalize=True, sample_weight=None))
print('\n')
predict_y_test_accuracy = clf.predict(X_test_final)
print("The test accuracy is:", accuracy_score(y_test_final, predict_y_test_accuracy,
    normalize=True, sample_weight=None))
print('\n-----\n')
predicted_y = np.argmax(predict_y, axis=1)
print("Total number of data points in test data:", len(predicted_y))
print('\n')
print("Number of correctly classified points in test data = ", accuracy_score(y_test_final,
    predict_y_test_accuracy, normalize=False))
print('\n-----\n')
plot_confusion_matrix(y_test_final, predicted_y)
```

A Novel Approach for Question Pair Similarity Identification

The Result came out as:

The train log loss is: 0.3067093284342253

The test log loss is: 0.41839509859716916

The train accuracy is: 0.8560370694449857

The test accuracy is: 0.7842811092284167

Total number of data points in test data: 120985

Number of correctly classified points in test data = 88837

Now, since the train log loss and test log loss are almost similar in addition to similarity of train accuracy score and test accuracy score, we can conclude that our Gradient Boost Decision Tree Model[18] is not overfitting or underfitting. It is working correctly.

The result of confusion matrix is as follows:

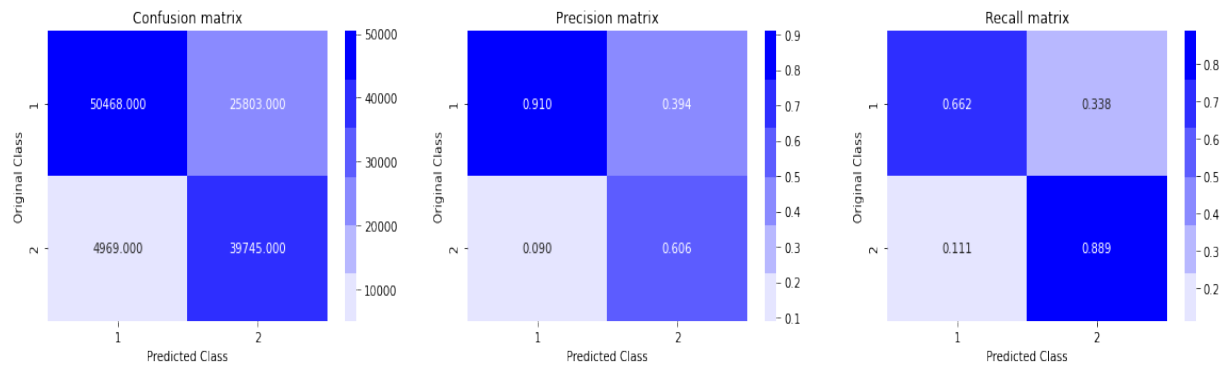


Figure 13 Gradient Boost Decision Tree Confusion Matrix Result

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

A Novel Approach for Question Pair Similarity Identification

8.3 Stacking Classifier[19][20] with Log Loss & Accuracy

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=11. As a final estimator we keep Gradient Boost Decision Tree with max_depth=10, subsample=0.85.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.ensemble import StackingClassifier
import xgboost as xgb

estimators = [('rf', RandomForestClassifier(n_estimators=70, max_depth=
11, 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_train_log_loss = clf.predict_proba(X_train_final)
print("The train log loss is:", log_loss(y_train_final, predict_y_train_
log_loss, eps=1e-15))
print('\n')
predict_y_test_log_loss = clf.predict_proba(X_test_final)
print("The test log loss is:", log_loss(y_test_final, predict_y_test_log
_loss, eps=1e-15))
print('\n-----\n')
predict_y_train_accuracy = clf.predict(X_train_final)
print("The train accuracy is:", accuracy_score(y_train_final, predict_y_
train_accuracy, normalize=True, sample_weight=None))
print('\n')
predict_y_test_accuracy = clf.predict(X_test_final)
print("The test accuracy is:", accuracy_score(y_test_final, predict_y_te
st_accuracy, normalize=True, sample_weight=None))
print('\n-----\n')
predicted_y = np.argmax(predict_y, axis=1)
print("Total number of data points in test data:", len(predicted_y))
print('\n')
print("Number of correctly classified points in test data = ", accuracy
_score(y_test_final, predict_y_test_accuracy, normalize=False))
print('\n-----\n')
plot_confusion_matrix(y_test_final, predicted_y)
```

A Novel Approach for Question Pair Similarity Identification

The result came out as follows:

The train log loss is: 0.3337574162829578

The test log loss is: 0.42296628046467

The train accuracy is: 0.8320751308093708

The test accuracy is: 0.7895156424350126

Total number of data points in test data: 120985

Number of correctly classified points in test data = 91890

Now, since the train log loss and test log loss are almost similar in addition to similarity of train accuracy score and test accuracy score, we can conclude that our Stacking Classifier Model[19][20] is not overfitting or underfitting. It is working correctly. It takes a long time to run. So, we could not test much changing different parameters. But we are hopeful that if we can choose the parameters optimally this model will give very good result.

The result of confusion matrix is as follows:

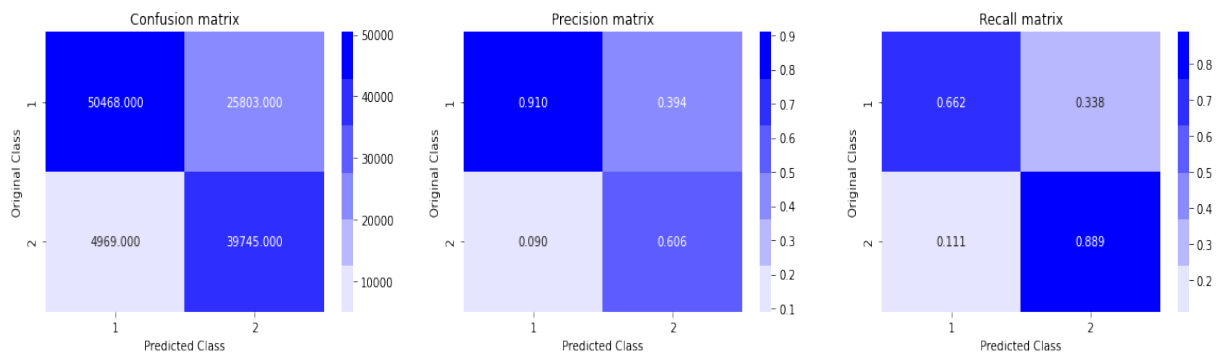


Figure 14 Stacking Classifier Confusion Matrix Result

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

A Novel Approach for Question Pair Similarity Identification

9

Testing with Real Life Example

Now that we have deployed three best machine learning models for this dataset, it is time to observe how good they are performing when they are given real life dataset. We have taken three question pairs for this test. They are as follows:

- 1. How can milk be stored at room temperature?
Which flavor of corn flakes goes best with milk?**
- 2. What are the benefits of 8GB RAM over 4GB RAM?
Which is better for gaming: i3 with 4GB or 8GB RAM?**
- 3. How do I get an internship in ISRO?
How do I apply for a summer internship in ISRO?**

We applied the same preprocessing techniques to prepare final dataset containing these three questions.

A Novel Approach for Question Pair Similarity Identification

9.1 Predicting with Logistic Regression and SGD[15]

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import RandomizedSearchCV

clf = SGDClassifier(alpha=0.0003335, penalty='l2', loss='log', random_state=42)
sig_clf = CalibratedClassifierCV(clf, method="sigmoid")
sig_clf.fit(X_train_final, y_train_final)

predict_y_exp_accuracy = sig_clf.predict(X_exp_final)

res_logistic = pd.DataFrame(data=predict_y_exp_accuracy, columns=["res_logistic"])
res_logistic['id'] = df_hold['id']
res = df_hold.merge(res_logistic, on='id', how='left')
```

The result is as follows:

	i d	Qid 1	Qid 2	Question1	Question2	Res_Logistic Regression
	0	1	2	How can milk be stored at room temperature?	Which flavor of corn flakes goes best with milk?	1
	1	3	4	What are the benefits of 8GB RAM over 4GB RAM?	Which is better for gaming: i3 with 4GB or 8GB RAM?	1
	2	5	6	How do I get an internship in ISRO?	How do I apply for a summer internship in ISRO?	1

Table 3 Logistic Regression Prediction Result

Here duplicate=1 and not-duplicate=0

The first question pair is actually not-duplicate. But Logistic Regression classified it as duplicate. So, it is clear that Logistic Regression is not performing well in real life also. Let's check out how the other three models are performing since they were performing way better than Logistic Regression in cross validation and test data analysis.

A Novel Approach for Question Pair Similarity Identification

9.2 Predicting with Random Forest Classifier[16]

```
from sklearn.ensemble import RandomForestClassifier as RFC
clf = RFC(n_estimators=75,max_depth=11,n_jobs=-1)#low bias high variance model, as depth increases variance increases. while bagging the variance will come down automatically in fact very low. n_jobs=-1 to parallelize the task into cpu cores
        #class_weight={0: 1, 1: 1.75}
clf.fit(X_train_final,y_train_final)

predict_y_exp_accuracy = clf.predict(X_exp_final)

res_RandomForest = pd.DataFrame(data=predict_y_exp_accuracy, columns=["res_RF"])
res_RandomForest['id'] = res['id']
res = res.merge(res_RandomForest, on='id',how='left')
res.head()
```

The result is as follows:

	id	Qid1	Qid2	Question1	Question2	Res_Random Forest
	0	1	2	How can milk be stored at room temperature?	Which flavor of corn flakes goes best with milk?	0
	1	3	4	What are the benefits of 8GB RAM over 4GB RAM?	Which is better for gaming: i3 with 4GB or 8GB RAM?	1
	2	5	6	How do I get an internship in ISRO?	How do I apply for a summer internship in ISRO?	1

Table 4 Random Forest Prediction Result

Here duplicate=1 and not-duplicate=0.

We can observe that the first question pair which is classified as duplicate by Logistic Regression is actually not-duplicate. So, here Random Forest classified it correctly as not-duplicate and hence we can see the improvement. Also, it supports the cross validation and test data analysis scores.

A Novel Approach for Question Pair Similarity Identification

9.3 Predicting with Gradient Boost Decision Tree[18]

```
import xgboost as xgb
clf = xgb.XGBClassifier(max_depth=8, n_estimators=400, learning_rate=0.02,
    colsample_bytree=.9, gamma=0, reg_alpha=4, objective='binary:logistic',
    eta=0.3, silent=1, subsample=0.9)
clf.fit(X_train_final, y_train_final)

predict_y_exp_accuracy = clf.predict(X_exp_final)

res_XgBoost = pd.DataFrame(data=predict_y_exp_accuracy, columns=["res_GBDT_XgBoost"])
res_XgBoost['id'] = res['id']
res = res.merge(res_XgBoost, on='id', how='left')
res.head()
```

The result is as follows:

	id	Qid1	Qid2	Question1	Question2	Res_Gradient_Boost_Decision_Tree
	0	1	2	How can milk be stored at room temperature?	Which flavor of corn flakes goes best with milk?	0
	1	3	4	What are the benefits of 8GB RAM over 4GB RAM?	Which is better for gaming: i3 with 4GB or 8GB RAM?	1
	2	5	6	How do I get an internship in ISRO?	How do I apply for a summer internship in ISRO?	1

Table 5 Gradient Boost Decision Tree Prediction Result

Here duplicate=1 and not-duplicate=0.

We can observe that the first question pair which is classified as duplicate by Logistic Regression is actually not-duplicate. So, here Gradient Boost Decision Tree classified it correctly as not-duplicate and hence we can see the improvement. Also, it supports the cross validation and test data analysis scores.

A Novel Approach for Question Pair Similarity Identification

9.4 Predicting with Stacking Classifier[19][20]

```
from sklearn.svm import LinearSVC
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
from sklearn.ensemble import StackingClassifier
import xgboost as xgb

estimators = [('rf', RandomForestClassifier(n_estimators=70, max_depth=
11, 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_exp_accuracy = clf.predict(X_exp_final)
predict_y_exp_prob = clf.predict_proba(X_exp_final)

import xgboost as xgb
clf = xgb.XGBClassifier(max_depth=8, n_estimators=400, learning_rate=0.
02, colsample_bytree=.9, gamma=0, reg_alpha=4, objective='binary:logist
ic', eta=0.3, silent=1, subsample=0.9)
clf.fit(X_train_final, y_train_final)

predict_y_exp_accuracy = clf.predict(X_exp_final)

res_StackingCl = pd.DataFrame(data=predict_y_exp_accuracy, columns=["re
s_StackingCl"])
res_StackingCl['id'] = res['id']
res = res.merge(res_StackingCl, on='id', how='left')
res.head()
```

The result is as follows:

	id	Qid1	Qid2	Question1	Question2	Res_Stacking Classifier
	0	1	2	How can milk be stored at room temperature?	Which flavor of corn flakes goes best with milk?	0
	1	3	4	What are the benefits of 8GB RAM over 4GB RAM?	Which is better for gaming: i3 with 4GB or 8GB RAM?	1
	2	5	6	How do I get an internship in ISRO?	How do I apply for a summer internship in ISRO?	1

Table 6 Stacking Classifier Prediction Result

A Novel Approach for Question Pair Similarity Identification

Here duplicate=1 and not-duplicate=0.

We can observe that the first question pair which is classified as duplicate by Logistic Regression is actually not-duplicate. So, here Stacking Classifier classified it correctly as not-duplicate and hence we can see the improvement. Also, it supports the cross validation and test data analysis scores.

9.5 Conclusion after testing

We can observe that how Random Forest, Gradient Boost Decision Tree, Stacking Classifier are outperforming Logistic Regression when we are taking user input Question Pairs also. This surely supports the cross validation and test data analysis Scores.

Hence, we can conclude that Random Forest, Gradient Boost Decision Tree(XgBoost), Stacking Classifier(MLXtend) are working pretty well in classifying the Q pairs. The 2nd pair is actually not duplicate, but they are very close since they have almost 80% words in common. Our aim will be improving our model such that it can detect this kind of pairs also.

A Novel Approach for Question Pair Similarity Identification

Conclusion and Recommendation

After going through all the results that different Machine Learning Models are providing, we can conclude that:

- Random Forest is working fine with test log loss 0.44 and test accuracy 0.76.
- But the best result is given by Gradient Boost Decision Tree with test log loss 0.41 and test accuracy 0.78 and Stacking Classifier with test log loss 0.42 and test accuracy 0.78.
- 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.
- When we are testing with Question Pairs given by users, there also Random Forest, Gradient Boost Decision Tree and Stacking Classifier are working quite well.
- So, we can conclude that the best Machine Learning Models for this dataset are Gradient Boost Decision Tree and Stacking Classifier according to the results and statistics we have got so far.
- 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.

A Novel Approach for Question Pair Similarity Identification

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/abhishekkkrthakur/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

A Novel Approach for Question Pair Similarity Identification

- [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.

A Novel Approach for Question Pair Similarity Identification

- [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 classification,” in *Proc. Conf. Empirical 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.