Capstone Project 1

Duplicate detection in the Quora question pairs dataset

Zahiduzzaman Biswas

Abstract

Semantic similarity analysis is a well-known research area in the domain of Natural Language processing (NLP). Using Traditional NLP techniques and machine learning tools, two texts can be predicted with remarkable accuracy whether they are similar or not. In this paper, such techniques are implanted using simple Naïve Bayes classifier to predict whether each pair of questions are duplicate or not.

1. Introduction

With the increasing access to the fast internet all over the world, internet has become one of the most popular sources of knowledge. The number of the users in the question-answer forums like Quora, Reddit, stackoverflow etc. is growing rapidly. Since millions of users, from different technical backgrounds and experiences, ask questions, similar or same questions, many a time, repeat multiple times. It eventually creates a maze of threads, where finding the best answer become a challenge for the new users. In order to provide the best and the most efficient user-experience, therefore, a reliable online knowledge base without any duplicate questions is essential. In the field of natural language processing, long standing researches are going on to build models that can automatically detect duplicate questions and combine them in one thread in order to help the users find the best possible answers to their questions. Recently, Quora hosted a competition on Kaggle.com, to find a better solution to the problem. Currently, they use a Random Forest model to identify duplicate questions[1]. Inspired by the competition, we made an attempt to build a machine learning model that can efficiently detect the duplicate questions in the Quora question pair dataset.

In the competition, a training dataset with 404,290 labeled rows and a testing dataset of 3563475 question-pair rows without labels. Training dataset contains 2 text columns for the question pairs (‘question1’, ‘quuestion2’), 2 numeric columns of ids for each question (‘qid1’, ‘qid2’), 1 numeric column for id of each question pair and 1 label column to indicate whether the pair is duplicate or not(‘is_duplicate’). The training question-pairs are labeled manually and contains errors. In the test dataset, there is 1 column of test id for each question pair and 2 text columns for the question pairs. [1]

The goal of this competition is to predict which of the provided pairs of questions contain two questions with the same meaning. A duplicate pair does not necessarily contain the same words. For example, “What is the step by step guide to invest in share market?” and “What is the step by step guide to invest in share market in India?” have almost same wording, but the intents of them are different; therefore, the are not duplicate. On the other hand, “How is the new Harry Potter book ‘Harry Potter and the Cursed Child’?” and “How bad is the new book by J. K Rowling?” do not have any word in common, but they are duplicate, for they have the

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

same answer. Therefore, the challenge is whether the model can predict the questions in each pair are semantically equivalent and can be answered with the same words.

2. Background/Related Work

Symantec similarity is a major field of research in the NLP domain. Companies working on machine translation, search engine, text summarization and so on directly or indirectly uses Symantec similarity approaches. Budanitsky and Hirst (2006) [2] provided an overview of many of the knowledge-based measures to find Symantec relatedness derived from the rich resources of WordNet. Prior works are also done on semantic relatedness of sentences. These works mostly focused on logical inference and entailment through based on the Stanford Natural Language Inference Corpus. One of the notable work by Zhigou et al. [3] proposed two neural-based frameworks—"Siamese" neural network approach and "compare-aggregate" approach. One the first of works focusing on attention methods using LSTMs was Rocktaschel et al. (2015) [5] which introduced word-by-word attention methods with the hypothesis attending on the premise. Some famous papers were published by Tomas Mikolov [6] where he used deep learning ideas such as word2vec and doc2vec. Some of our approaches are inspired by these papers. Besides, many contestants in the quora question pair published their works, which are mostly based on neural network approaches.

3. Approach:

3.1 Data Exploration :

The provided training dataset is comparably clean, well-structured, and fully labeled. It has only two rows containing null values, which is less than 0.0005% of the overall training data. Therefore, these two were removed. Although many questions were repeated in each column but as a pair all the remaining 404,288 rows are unique. Finally, the training dataset has 149263 question pairs labeled as duplicate, and 255025 pairs are not duplicate.

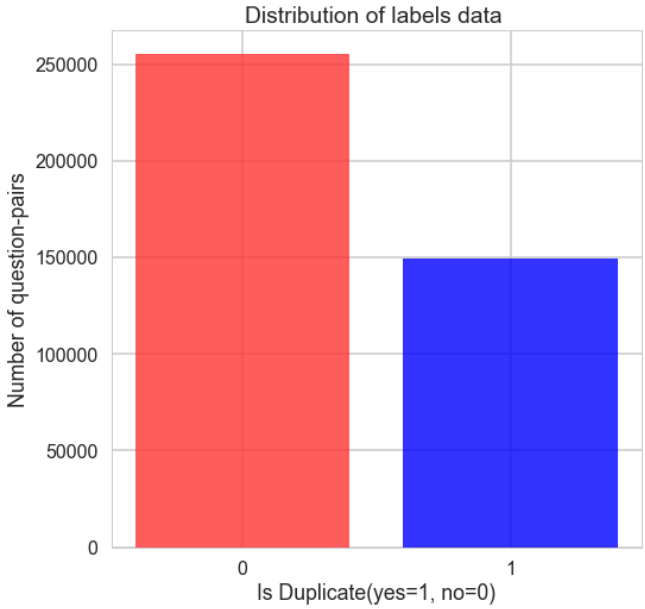

Figure 1: Bar plot of the labels in training dataset

On the other hand, the test dataset is preprocessed before it is used for model evaluation. The provided dataset contains 3563475 rows of testing data, out of which about 262144 rows contains invalid test ids. Besides, the dataset also contains over a million of duplicate data. After the preprocessing, the test dataset contains 2345796 unique rows of testing data.

In the training dataset, over 99% of the questions in question1 columns are consisted of less than 300 characters. Almost all the questions contain less than 80 words. (Firgure 2).

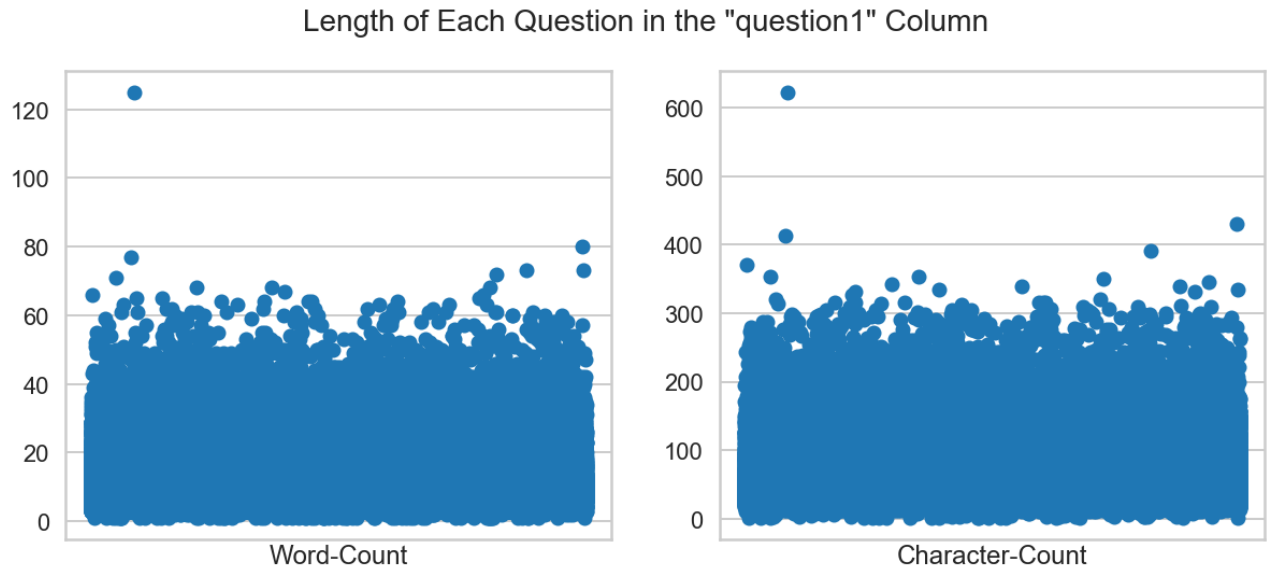

Figure 2: Plots of number of characters in each text in 'question1'

In the question2 column, over 99% of the questions are consisted of less than 400 characters. Almost all the questions contain less than 80 words. (Firgure 3).

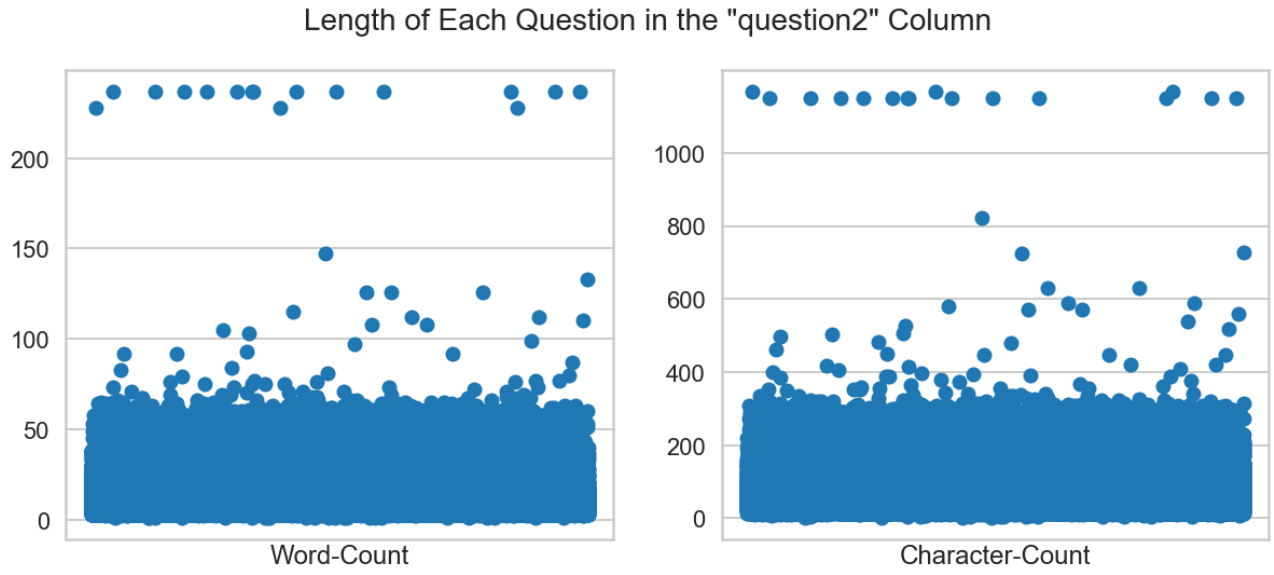

Figure 3: Plots of number of characters in each text in 'question1'

From the exploration of the cleaned data, one of the most common words is found to be ‘india’, which indicates a large number of the questions were related to India. A few other most frequent words are ‘donald trump’, ‘hilary clinton’ etc. A good share of the questions, therefore, are asked about the USA presidential election in 2016. Besides those, some notable words related to contemporary trends such as ‘weight loss’, ‘programming language’, ‘make money’ etc. are clearly visible in the figure 2. In summary, questions from different fields and perspectives are available in the training data, which adds complexity to the task of finding the common intents.

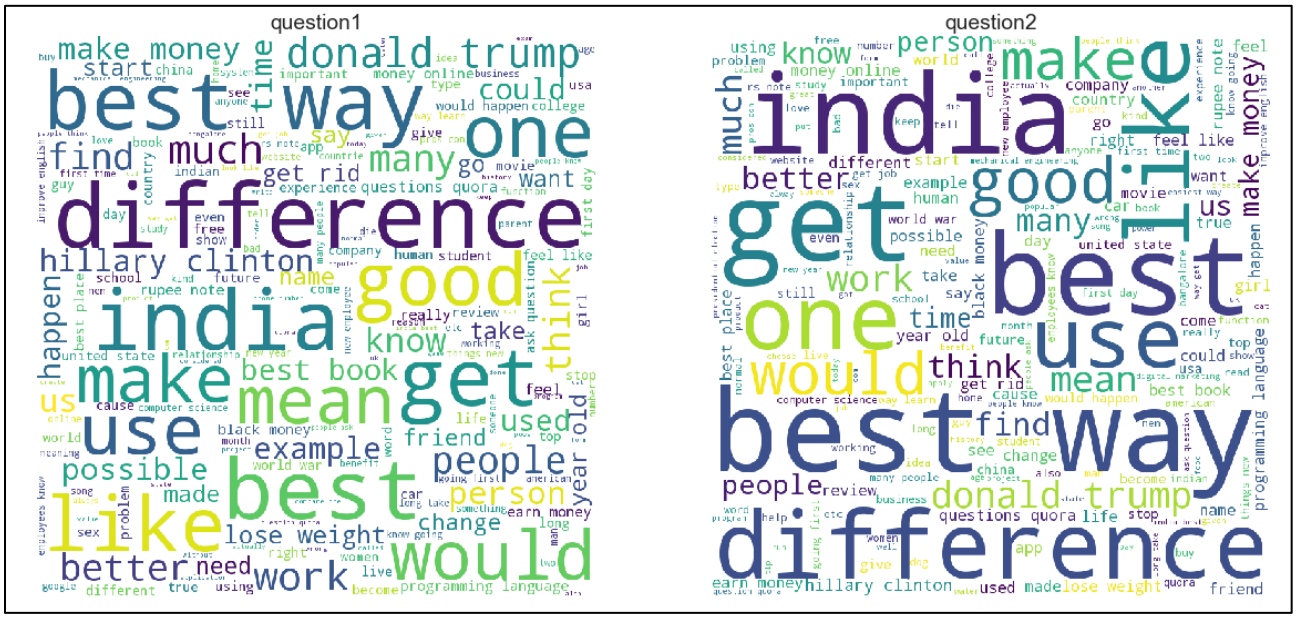

Figure 4. Word cloud for most frequent words in the training dataset

3.2 Text Preprocessing

The purpose of the text preprocessing is to reduce each question text to a minimal form for efficient embedding and vectorization. A noisy text with redundant characters or words could hamper the accuracy of the model. Therefore, the following steps are performed at this stage:

- Both question1 and question2 columns contain some questions with less than 3 words. Most of these questions do not express any meaningful intent. For example, ‘Big Data?’, ‘Uri Attack?’ ‘…… …?’ etc. For a classification, any question text with 2 words or less would be counted for noise. That is why, we remove the corresponding rows.
- The question texts are preprocessed with the preprocessing functions of the genism library. The texts contain numbers, letters of different cases, special characters, punctuations, and foreign characters. Each row of the question text is at first converted to lower case. All the non-alphanumeric characters such as punctuations, special characters, and foreign characters are then filtered out, leaving only the English letters and numbers along with the spaces in each text. Commonly used words, for instance, ‘articles and prepositions as well as redundant whitespaces and single character words contribute very little to the meaning of the texts. For the efficiency those are also removed from the question-pairs. In the remaining text, many words exist in different morphological form. These words are

stemmed to the common base form. For example, ‘costs’ and ‘cost’ are both reduced to ‘cost’ by stemming. Consequently, it would reduce the dimensionality of word vectors.

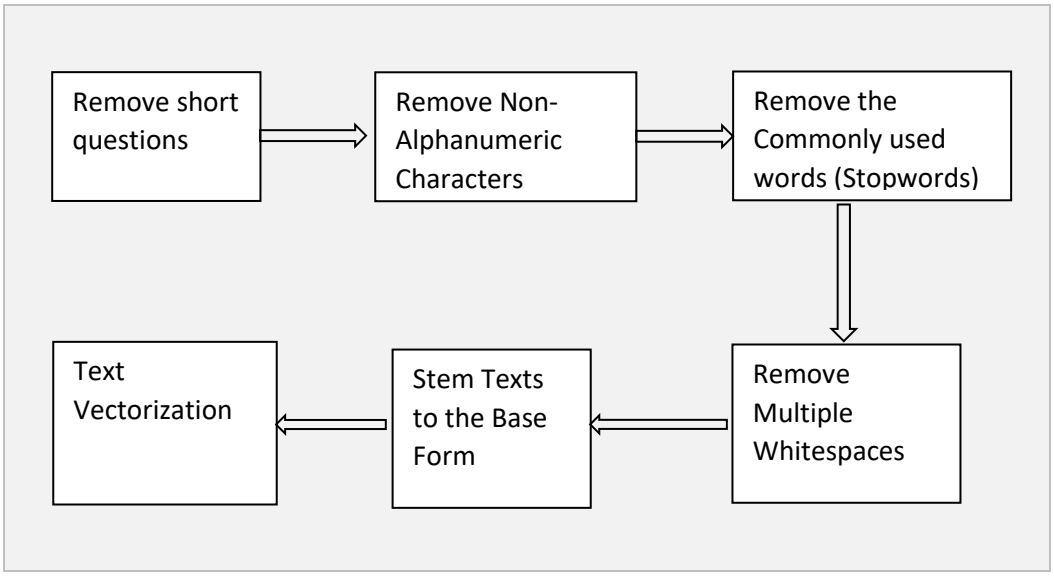

Figure 5: Flow chart of the overall steps to the text data processing

3.3 Text Vectorization:

Preprocessed text is vectorized as a numerical feature to input the question texts to the machine learning algorithm. The question-pairs are converted to the Bag-of-word representation. That is, the entire question-pair corpus is tokenized and assigned an id for each token. Each of these tokens is turned in to a numeric feature based on the frequency. Since the highly frequent may influence the model more than its actual weight, all the tokens are reweighted with Tf-idf term weighing. Tf stands for term frequency while idf means inverse document frequency. In Tf-idf term weighing frequency or number of occurrences of each term in a document is multiplied by its idf component:

$$\text{tf-idf(t,d)} = \text{tf(t,d)} \times \text{idf(t)}$$

$$\text{idf}(t) = log\frac{1+n_d}{1+\text{df}(d,t)} + 1$$

Here t represents each term, df(d,t) is the frequency of the term in a document (d), and idf(t) is idf component of the term. The following cumulative plot shows the document frequency in the training data. At the document count of 1 the graph starts to become steep. Then at x = 20, it hits the plateau.

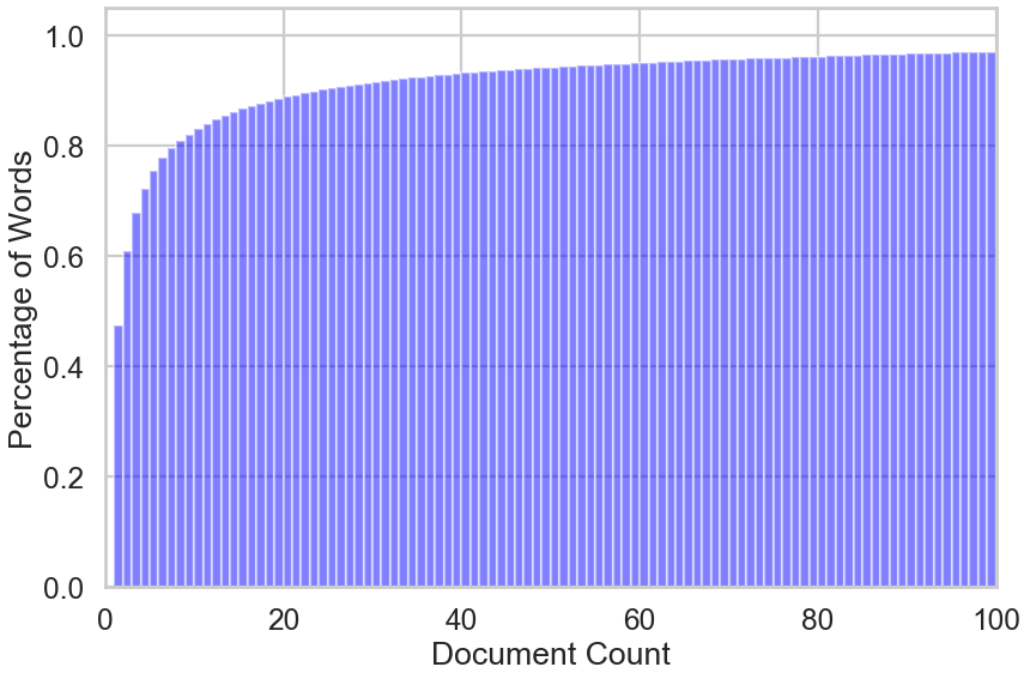

Figure 6: Cumulative plot of the document frequency in the Training data

3.4 Doc2Vec Embeddings and Distance Measure

Each question is also embedded with Doc2vec and represented by a unique numeric vector. Doc2Vec is a way to project a document or sentences in a high dimensional vector space. It projects each word in a question document in the vector space using the window of words around it and concatenates the word vectors (W) together. At the same time, it concatenates a unique document vector (D) for each question to finalize the vector representation.

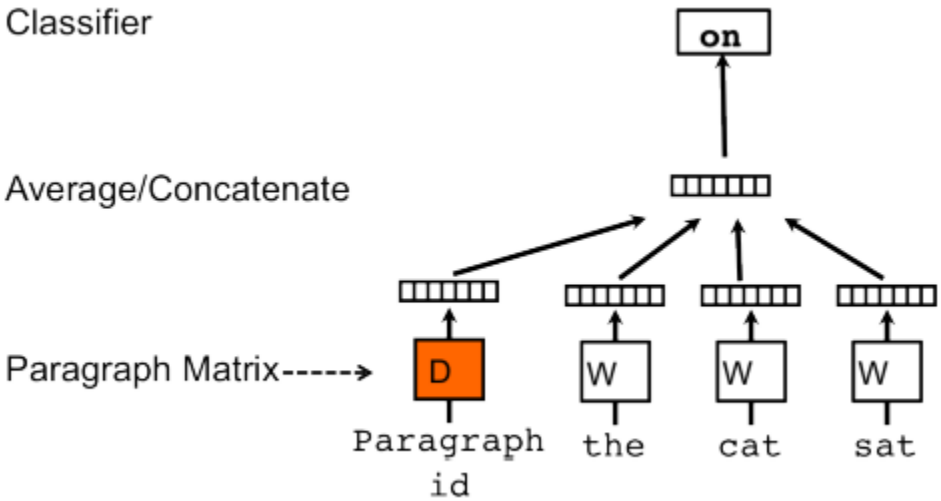

Figure 5. Framework of Doc2vec in learning a paragraph/sentence/document [6]

Since the words are vectorized based on the context, distances of these vectors from each other in the vector space represents how semantically similar they are. Different distance algorithms are used to compare the similarity and differences between the documents. Two most common metrics, Cosine distance and Euclidean distance, are used for extracting two numerical features for question-pair texts. Cosine distance measures the cosine of the angle between two vectors. It is calculated from the dot product of the two vectors. Cosine distance between x and y vectors are calculated using the following equation:

$$\cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{|\vec{x}| \cdot |\vec{y}|}$$

Euclidean distance is calculated using the following equation:

$$|\vec{x} - \vec{y}| = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

3.5 Naïve Bayes classifier:

Finally, a Multinomial Naïve Bayes classifier is used to train the duplicate question pair detection models. The Naïve Bayes classifier takes all the input features and classify the data based on the Bayes theorem:

$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)}$$

However Multinomial Naïve Bayes is a variant of this traditional Naïve Bayes classifier. After the Laplace smoothing, in the Multinomial Naïve Bayes Classifier, the likelihood of a feature in a class change as below:

$$P(f_i|c) = \frac{N_{ic} + \alpha}{N_c + \alpha N_i}$$

where N_{ic} is the number of times feature f_i was seen in class c; N_c is the number of times class c was seen; N_i is the number of times feature f_i was seen globally; and α is a regularization parameter used for Laplace smoothing.

3.6 Evaluation Metrics

The provided training dataset contains some erroneous labeling due to human error. As a result of this discrepancy achieving 100% accuracy is never possible. That's why the model is trained to predict probability of labels, instead of the exact label. Therefore, log loss metric is chosen for the evaluation. For the true class labels y = {0,1}, if the probability of y = 1 is p, the log loss is the negative log likelihood of the classifier:

$$-\log \Pr(y|p) = -(y \log(p) + (1 - y) \log(1 - p))$$

4. Experiments & Results:

4.1 Base Model: Multinomial Naïve Bayes with Tf-idf vectorizer without hyperparameter tuning:

The base model is trained on five features: character-lengths of each question in the pair, word-count of them, tf-idf vectorized representation of the text of each combined question-pair. A simple pipeline model is built with first step to combine all the numeric features with the tf-idf vectorized feature, and the second step is to classify them with the multinomial Naïve Bayes classifier. Some of the key parameters of vectorizer such as in how many documents a word appeared (min_df), combine the common phrases into features (ngram) and the Laplace smoothing parameter (α) are set to the default values. Log loss result of test data in the Kaggle evaluation system, 0.438 for public set and 0.440 for private set.

Hyper-parameters	Values
min_df	1
ngram	1
α	1

Table 1: Values of the hyper-parameter in the base model

4.2 Updated Base Model with Tuned Hyper-parameters:

At this step, the texts are vectorized using bigram (ngram =(1,2)) in the tf-idf vectorizer. Rest of the parameters are tuned with grid search and cross validation. The updated model with tuned hyper-parameters scored 0.403 on the public set and 0.404 on the private test set in Kaggle evaluation system.

Hyper-parameters	Values
min_df	1
ngram	(1, 2)
α	0.1

Table 2: Values of the hyper-parameter in the update model

4.3 Updated Model with Doc2vec trained on question corpus:

At this step, each question was vectorized in a higher dimensional vector space with Doc2vec. Then the consine distance and Euclidean distance between the questions in the pairs are measured in the vector space. The previous updated model is further improved by adding this distance metrics as features. This model scored 0.398 on the public set and 0.399 on the private test set in Kaggle evaluation system.

Hyper-parameters	Values
min_df ngram	0.0002 (1, 2)
α	0.1
Window epochs min_count vector_size	1 1 1 5

Table 3: Values of the hyper-parameter in the model

4.4 Improved model with updated Doc2vec training parameters:

At this step, the model is further improved by increasing window size and epoch as well as by updating other training parameters.

Hyper-parameters	Values
min_df ngram	0.0002 (1, 2)
α	0.1
Window epochs min_count vector_size	7 10 2 100

Table 4: Values of the hyper-parameter in the model

Finally, the model is experimented using a pretrained doc2vec model trained on Wikipedia. Below is the summary of the experiment.

S. No.	Models	Hyper-parameters	Public Score (log_loss)	Private Score (log_loss)
1	Base Model: Multinomial Naïve Bayes with, Tf-idf vectorizer	MultinomilaNB : α = 1.0 TfidfVectorizer: min_df = 1	0.438	0.440
2	Update model with bi-gram and Tuned parameters, Multinomial Naïve Bayes, Tf-idf vectorizer	MultinomilaNB : α = 0.15 TfidfVectorizer: min_df = 1 ngram = (1,2)	0.404	0.403

Table 5: Model experiments without Doc2Vec and the corresponding log loss scores from Kaggle system

A few experiments was conducted with the Doc2Vec models. We vectorized the texts using Doc2Vec to find the cosine and Euclidean distances between each question pair. Result of the corresponding experiments are presented below:

S. no.	Model Description	Parameters	Public Score (log-loss)	Private Score (log-loss)
3	Updated Model With	MultinomilaNB : α = 0.15	0.398	0.399

	Improved Multinomial Naïve Bayes Classifier, Improved Tf-idf vectorizer, Doc2Vec trained on question-paris corpus with default parameters	TfidfVectorizer: min_df = 1 ngram = (1,2) Doc2Vec: Window = 1 epochs = 1 min_count = 1 vector_size = 5 trained on question-pairs corpus		
4	Improved model with Improved Multinomial Naïve Bayes, Improved Tf-idf vectorizer, Doc2Vec with updated parameters trained on the Training question pairs	MultinomilaNB : α = 0.15 TfidfVectorizer: min_df = 1 ngram = (1,2) Doc2Vec: Window = 7 epochs = 10 workers = 8 min_count = 2 vector_size = 100 trained on question-paris corpus	0.390	0.398
5	Model updated with Improved Multinomial Naïve Bayes, Improved Tf-idf vectorizer, Pretrained Doc2vec model on wikipedia corpus	MultinomilaNB : α = 0.15 TfidfVectorizer: min_df = 1 ngram = (1,2)	0.404	0.405

Table 6: Model experiments without Doc2Vec and the corresponding log loss scores from Kaggle system

Finally, we tried a few different classifiers on Doc2Vec vectorized texts to find to observe the performances of them. Below is the result of them:

Classifiers	Training time	Public Score	Private Score
MultinomialNB	2 hours 13 minutes	0.560	0.561
Logistic Regression Classifier	Approximately 30 minutes	0.542	0.542

SVM	Over 8 hours (On 50% of training data)	0.550	0.552

Table 7: Model performance with different classifiers

5. Conclusion and Recommendations

This model is built on the basic Naive Base classifier with very minimal parameter tuning due to the time constraints. Further parameter tuning with cross validation of different folding of the training data set would improve the model. Besides, this model only uses 6 features such as the text of the questions, word counts, and character counts. It also ignores the case of the characters, foreign characters, and special characters, which may cause a potential information loss. For future experiment some recommended steps are following:

- Correct misspelled words, replace abbreviation with the complete words
- Add special characters count, word share ratio, capitalization count as features
- Besides cosine and Euclidean distances, other distance features can be added
- Fuzzy ratio features can be added
- Deep learning implementation such word2vec, artificial neural networks

References:

[1] Lili Jiang, Shuo Chang, and Nikhil Dandekar. Engineering at Quora: Semantic Question Matching with Deep Learning. https://engineering.quora.com/Semantic-Question-Matching-with-Deep-Learning. 2017

[2] Budanitsky and G. Hirst. 2006. Evaluating wordnet-based measures of lexical semantic relatedness. Computational Linguistics, 32(1):13–47.

[3] Wang, Zhiguo, Wael Hamza, and Radu Florian. "Bilateral Multi-Perspective Matching for Natural Language Sentences." arXiv preprint arXiv:1702.03814 (2017).

[4] Tomas Mikolov, Stefan Kombrink, Lukas Burget, Jan Cernocky, and Sanjeev Khudanpur. Extensions of recurrent neural network language model. In Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on, pages 5528–5531. IEEE, 2011.

[5] Tim Rocktaschel, Edward Grefenstette, Karl Moritz Hermann, Tomas Kocisky, Phil Blunsom. Reasoning about entailment with neural attention. In ICLR 2016.

[6] Tomas Mikolov, Iliya Sutskever, Kai Chen, Greg Corrado, and Jeffery Dan. Distributed Representation of Words and Phrases and . In Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on, pages 5528–5531. IEEE, 2011.