KANTIPUR ENGINEERING COLLEGE

(Affiliated to Tribhuvan University)

Dhapakhel, Lalitpur

[Subject Code: CT755]

A MAJOR PROJECT FINAL REPORT ON

SEMANTIC QUESTION PAIR MATCHING WITH DEEP LEARNING

Submitted by:

Ankit Kharel [2072/BCT/95]

Anshul Bikram Rana [2072/BCT/96]

Kritish Dhaubanjar [2072/BCT/102]

Pradyumna Subedi [2072/BCT/110]

A MAJOR PROJECT FINAL REPORT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR IN COMPUTER ENGINEERING

Submitted to:

Department of Computer and Electronics Engineering

August, 2019

SEMANTIC QUESTION PAIR MATCHING WITH DEEP LEARNING

Submitted by:

Ankit Kharel [2072/BCT/95]

Anshul Bikram Rana [2072/BCT/96]

Kritish Dhaubanjar [2072/BCT/102]

Pradyumna Subedi [2072/BCT/110]

Supervised by:

Ajay Mani Paudel

Executive Director

Natural Language Processing Hub

A MAJOR PROJECT FINAL REPORT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE DEGREE OF BACHELOR IN COMPUTER ENGINEERING

Submitted to:

Department of Computer and Electronics Engineering

Kantipur Engineering College

Dhapakhel, Lalitpur

August, 2019

ACKNOWLEDGMENT

First and foremost, we would like to express our sincere gratitude towards the Department of Computer and Electronics Engineering, Kantipur Engineering College, for providing us the opportunity to undertake this project which has helped us learn and implement our skills and knowledge. We are grateful to the Head of Department, **Er. Rabindra Khati**, for his precious encouragement, support and guidance throughout the project.

We would like to acknowledge, **Er. Ajay Mani Paudel**, for his constant supervision, unwavering support, invaluable suggestions and consultation throughout the project, which has propelled us to come a long way accomplishing the project with the finest result.

We would like to express our utmost appreciation to **Er. Shiva Gautam**, for helping us with machine learning at various stages of this project. We are also highly indebted to **Er. Bishal Thapa** for co-operating with us and helping us carry out this project smoothly.

Our thanks and appreciations also go to the authors of various papers that helped us gain the in-depth knowledge of various aspects. Finally, we are thankful to all our teachers and all other people who have contributed in completion this project and preparation of this report. We have tried to make this report error free. Any error in this report is our own responsibility. Suggestions and comments from readers are highly appreciable.

Sincerely,

Ankit Kharel	[2072/BCT/95]
Anshul Bikram Rana	[2072/BCT/96]
Kritish Dhaubanjar	[2072/BCT/102]
Pradyumna Subedi	[2072/BCT/110]

ABSTRACT

Q&A forums like Quora, Stack-overflow, Reddit, etc are highly susceptible to question pair duplication. Two questions asking the same thing could be too different in terms of vocabulary and syntactic structure, which makes identifying their semantic equivalence challenging. In this report, we explore methods of determining semantic equivalence between pairs of questions using a dataset released by Quora of more than 400,000 questions pairs through Machine Learning with Natural Language Processing.

Our machine learning approach is based upon Levenshtein distance between two sentences and the sentence-vector encoding using Word2Vec models to experiment with a variety of distance metrics and predict their semantic equivalence. We compare the standard supervised classification methods such as Logistic Regression, KNN and Random Forest with our artificial neural network (ANN). Our experimental results show that the artificial neural network with word embeddings achieves high performance, achieved F1-score of 0.6529 with 0.7236 accuracy on the test set.

Keywords– semantic analysis, duplicate questions, natural language processing, machine learning, word embeddings

TABLE OF CONTENTS

Acknowledgment i

Abstract ii

List of Figures v

List of Tables v

List of Abbreviations vi

1 Introduction 1

1.1 Background 1

1.2 Problem Statement 2

1.3 Objectives 2

1.4 Applications 2

1.5 Features 3

1.6 Feasibility Analysis 3

1.6.1 Economic Feasibility 3

1.6.2 Technical Feasibility 3

1.6.3 Operational Feasibility 3

1.7 System Requirements 4

1.7.1 Software Requirements 4

1.7.2 Hardware Requirements 4

2 Literature Review 5

3 Methodology 9

3.1 System Architecture and Overview 9

3.2 Data Acquisition and Dataset Description 10

3.3 Data Pre-processing 11

3.3.1 Word Tokenization 11

3.3.2 Stopwords 11

3.3.3 Stemming and Lemmatization 12

3.4 Feature Extraction 13

3.4.1 Traditional Bag of Words 13

3.4.2 Term Frequency Inverse-Document Frequency 14

3.4.3 Word Embedding: word2vec 15

3.4.4 Google News vectors 17
3.5 Feature Engineering 18
3.5.1 NLTK Library Groundworks 18
3.5.2 Fuzzy wuzzy Features 18
3.5.3 Word Mover Distance 19
3.5.4 Vector distances 19
3.6 Analysis and Visualization 22
3.7 Supervised Machine Learning Models 24
3.8 Neural Network Design 24
3.8.1 Framework of Artificial Neural Network: 25
3.8.2 Adam optimization 28
3.8.3 Training 28
3.9 Software Development Model 30
3.9.1 Sequence based on Test-Driven Development by Example: . . . 30
4 Output **32**
5 Results and Analysis **34**
5.1 Confusion Matrix 34
5.2 Accuracy and Loss Functions 35
5.3 Receiver Operating Characteristic Curve 37
5.4 Work Schedule 37
6 Conclusion **38**
6.1 Conclusion 38
6.2 Future Enhancement 38
References **39**

LIST OF FIGURES

1.1 Few sample lines of the quora dataset 1
2.1 Siamese neural network architecture 7
3.1 Overall System Architecture. 9
3.2 Distribution of the total question pairs. 10
3.3 Few questions after tokenization 11
3.4 A question after stopword removal 11
3.5 A question after stopword removal and stemming 12
3.6 A question after stopword removal and lemmatization 13
3.7 Bag of words model for a small corpus. 14
3.8 TF-IDF of words 'what' and 'lorem' 14
3.9 Word2vec model showing similar words of an unknown vector with semantic context 16
3.10 CBOW model 17
3.11 Simple feature extraction. 18
3.12 Fuzzy feature extraction. 19
3.13 Some vector distances between questions in pair 20
3.14 Correlation Heatmap of features. 21
3.15 Distribution of lengths of question pairs 22
3.16 Distribution of common words of question pairs 22
3.17 Scatter plot of common words and cosine distance of question pairs . . 23
3.18 Scatter plot of common words and fuzz partial ratio of question pairs . . 23
3.19 Architecture of Neural Network 25
3.20 ReLU activation function 26
3.21 Sigmoid Activation Function 27
3.22 Training of ANN 29
3.23 Test Driven Development Model 30
4.1 Home page to evaluate a pair of questions or lookup similar words. . . . 32
4.2 Evaluating a pair of questions. 32
4.3 Lookup similar words from word2vec model. 33
5.1 Training Loss of ANN 36

5.2 Training Accuracy of ANN 36
5.3 ROC Curve 37
5.4 Gantt Chart 37

LIST OF TABLES

3.1 Fuzzy Wuzzy sample features 19
3.2 Comparison of Supervised models for duplication analysis 24
5.1 Corresponding Confusion Matrix of ANN Model 34
5.2 Summary of Confusion Matrix 35
5.3 Training ANN at different learning rate. 35

LIST OF ABBREVIATIONS

ANN Artificial Neural Network

CBOW Continuous Bag of Words

CNN Convolutional Neural Network

KNN K Nearest Neighbour

LSTM Long Shot Term Memory

MLP Multi Layer Perceptron

NER Named Entity Recognition

NLTK Natural Language Toolkit

POS Part of Speech

ReLU Rectified Linear Unit

ROC Receiver Operating Characteristic

SVM Support Vector Machine

TF-IDF Term FrequencyInverse Document Frequency

WMD Word Mover Distance

CHAPTER 1
INTRODUCTION

1.1 Background

Quora has a monthly visitor of approximately 300 million users and more than 38 million questions have been asked till date[1]. It comes in no surprise that many users ask semantically equivalent questions. In this study we define two questions as semantically equivalent if they can be adequately answered by the exact same answer. In order to build a high-quality knowledge base, it is important to ensure each unique question exists on Q&A forums only once.

Understanding semantic relatedness of questions would allow understanding of much of the user-generated content on the Q&A forums. Thus, detecting duplicate questions could greatly benefit the community. Quora, infact realized the severe issue and hence published its dataset for the first time in 27 Feb, 2017 [2], which could be used for the machine learning for data analysis.

id	qid1	qid2	question1	question2	is_duplicate
447	895	896	What are natural numbers?	What is a least natural number?	0
1518	3037	3038	Which pizzas are the most popularly ordered pizzas on Domino's menu?	How many calories does a Dominos pizza have?	0
3272	6542	6543	How do you start a bakery?	How can one start a bakery business?	1
3362	6722	6723	Should I learn python or Java first?	If I had to choose between learning Java and Python, what should I choose to learn first?	1

Figure 1.1: Few sample lines of the quora dataset

Source: https://data.quora.com/First-Quora-Dataset-Release-Question-Pairs

In figure 1.1 though the questions with qid1 and qid2 of 6542 and 6543 respectively mean the same, the phrase are completely different and hence are the duplicate questions. The existence of duplicate questions can be analyzed using the machine learning and training them using the datasets.

Unlike the first one, the questions with qid1 and qid2 of 895 and 896 respecively shown in figure 1.1 look similar in terms of phrases but are not the duplicates. We need to test for exact semantic coincidence between questions and analyze if it is the question with

same intent.

1.2 Problem Statement

All the forums are based on the principle that ensures, each unique question exists on the forum only once and that there should be a single question section for each logically distinct question. For example, we'd consider questions like "What are the best ways to lose weight?", "How can a person reduce weight?" and "What are effective weight loss plans?" to be duplicate questions because they all have the same intent and should not exist separately because the intent behind both is identical.

The duplication of questions in Q&A Forums results in the lack of sensible search for the user base. A dull redundancy in the questions and answers still result in inadequacy of responses to the questioners. Inappropriate segragation of the valid information stream and unwanted distribution of the user base on varying level of activity and intellect of answers while polluting the user's feed.

1.3 Objectives

1. To develop a module to analyze and identify the questions having same semantic meaning so as to prevent the duplication for Q&A forums like Quora .
2. To develop a module that would assist mitigatation of redundancy in answers to support high-quality knowledge base.

1.4 Applications

Ideally, using this approach, the indentified duplicate questions could be merged together into a single canonical question, as doing so would provide a number of usage.

1. Avoid frequently repeated questions from polluting highly engaged user's feed on Q&A forums caused by slight variations on the same questions.
2. Improve search and discovery based on knowlegde of alternative phrasings of the same questions.

3. High quality Q&A knowledge bases having more value to users and researchers when there is a single cannonical question and colections of answers, instead of having knowledge fragmented and spread throughout the site.

1.5 Features

1. Deep Learning to classify whether question pairs are duplicates or not.
2. Selected highly dominant features from the questions and implementation of minimal cost machine architectural model.

1.6 Feasibility Analysis

1.6.1 Economic Feasibility

The economic feasibility determined from the nominal research shows positive economic benefits to the related institution that the system will provide. It involves cost/benefits analysis and it is the most frequently used method for evaluating the effectiveness of a new proposed system. Our project has requirements of software and hardware at medium level on implementation, that means it does not have a high economic requirement and is economically feasible to us, even at a nominal expense.

1.6.2 Technical Feasibility

The technical feasibility studied the technical requirements that are to be taken under consideration in order to know if the proposed system is feasible or not. The technical area related to hardware and software requirements, requires not much of software nor hardware, and ultimately, it can be considered technically feasible.

1.6.3 Operational Feasibility

Operational feasibility that accounts for our developed system has no demand for expert human resources. Considering the simplicity of our program, and the machine model obtained from it, we are assured that our project is operationally feasible to any area of

application.

1.7 System Requirements

1.7.1 Software Requirements

Python is the choice for analyzing large volumes of data, processing it and implementing it for machine learning, with a large community for support and simplicity.

- **Development platform:** Python 3.7 or higher
- **Operating System:** Windows 7, Ubuntu 16.04
- **Integrated Development Environment:** Jupyter Notebook, Spyder
- **Python Libraries:** keras, matplotlib, fuzzywuzzy, scikit-learn, nltk, numpy, pickle, pandas, flask

1.7.2 Hardware Requirements

- **RAM:** 8 GB RAM (Minimum)
- **Processor:** Intel Core i5-7th Generation
- **Storage:** 10 GB

CHAPTER 2
LITERATURE REVIEW

In the paper 'A Paraphrase and Semantic Similarity Detection System for User Generated Short-Text Content on Microblogs', a set of features was proposed that, although well-known in the NLP literature for solving other problems, had not been explored for detecting paraphrase or semantic similarity, on noisy user-generated short-text data such as Twitter. Support vector machine (SVM) based learning was applied and used the benchmark Twitter paraphrase data, released as a part of SemEval 2015, for experiments. The system delivered a paraphrase detection F1-score of 0.717 and semantic similarity detection F1-score of 0.741, thereby significantly outperforming the then-existing systems, that deliver F1-scores of 0.696 and 0.724 for the two problems respectively.[3] The main contributions of the work are the following:

- It provided a machine learning based model, and a set of lexical, syntactic, semantic and pragmatic features, for detecting paraphrases on user-generated noisy Twitter short-text content data.
- Empirically showed the goodness of the proposed features, on a benchmark Twitter paraphrase corpus. The system outperformed all the systems that existed in the state of the art.
- Further demonstrated the effectiveness of the feature set on benchmark clean-corpus text data, namely Microsoft Paraphrase dataset, and obtain a rank within the top 10 existing systems, by training the system (the features) on their dataset, and testing on their dataset as well.

Features Selection

- **Lexical Features :** Word-level gram features, Stemming, Stopword removal etc.
- **Syntactic Features:** Part-of-Speech (POS) agreement features, Named entity (NE) features
- **Semantic Features:** Word overlap features (Best word, Adjectives), Phrase overlap features
- **Pragmatic Features:** Subjectivity/objectivity agreement feature

Choice of Approach: Why not Deep Learning?

This approach used the traditional methodology of identifying features, and a traditional SVM classifier. The choice of modeling in the given manner was, in the known state of the art, traditional feature-based systems, had so far outperformed deep learning systems, such as the recurrent neural network based approaches for paraphrase detection. A deeper subsequent study also models deep neural learning systems, and observed a similar shortcoming of such systems. A likely cause of this anomaly is the simple absence of a sufficiently large paraphrase dataset in the domain of user-generated noisy texts on microblogs such as Twitter.

In the paper 'Duplicate Quora Questions Detection' [4] the main idea was to first vectorize questions and extract features, train and predict using machine learning techniques based on question vectors and features previously built. They implemented two approaches to detect if two questions are duplicate. Different vectorization and feature extracting methods were used in the two approaches, one based on the Word2Vec model and TF-IDF score, the other one is a Neural Network method based on term frequency. Different classification methods were also used, for example, KNN, SVM and Random Forest. This reached accuracies of nearly 80% in both two approaches.

Using a dataset of 400,000 labeled question pairs provided by Q&A forum Quora, researcher at Department of Computer Science, Stanford University explored a series of deep learning methodologies for detecting duplicate question pairs: convolutional neural networks (CNNs), long short-term memory networks (LSTMs), and a hybrid model[5]. All three models are built atop a siamese network architecture and multilayer perceptron concatenation for the final inference. Empirical results show that LSTMs outperform CNNs in terms of accuracy, and that combining the two techniques provides no additional inference improvements. Moreover, all three deep learning techniques significantly outperform traditional NLP methods and simple multilayer perceptron baselines.

Each question was passed into a separate tower of the network with identical parameters and weights, producing a siamese network. The raw questions, represented as single-dimensional vectors of vocabulary indexes, are then converted into pre-trained word in

the embedding layer. The embedding matrix for each question is then passed through the encoding layer, that converts the word matrix into a single-dimensional feature vector. The feature vectors from each of the two independent networks are concatenated together. Finally, the concatenated feature vectors are passed through a Multi Layer Perceptron that produces the final output.

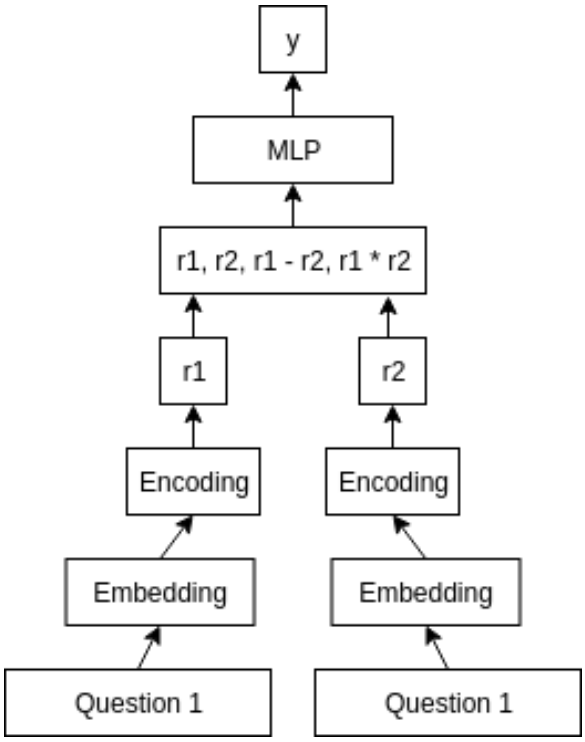

Figure 2.1: Siamese neural network architecture

Three different encoding strategies are used in the study in the encoding layer indicated in the figure above. Common to all three encoding methods is the final multilayer perceptron that combines the siamese network feature vectors together. The input to this network, when introduced, provided a 2% performance gain over a simple concatenation of the two feature vectors. Specifically, input used:

(r1, r2, r1 - r2, r1 .* r2)

Where r1 and r2 are the output feature vectors from the first question and the second question, respectively. The third concatenated vector is the difference between the first and the second vectors (which will ultimately be symmetric given that each tower in the

network uses the same model weights). Finally, the third concatenation is the element-wise product between the first and second vectors.

ReLU activation was used for nonlinearity, and the final output layer consists of a single (1 x 2) vector, with the two elements corresponding to each of the two classes: non-duplicate and duplicate.

In running experiments, a sample batch size of 64 was used and trained for a total of 20 epochs. Adam optimization was used with a learning rate of 0.001. Softmax cross entropy with logits was chosen for the loss. All of the implementation was written in TensorFlow, with separate run scripts for training and evaluation. Across all three of the models studied, a similar pattern of convergence was found. The model accuracy would sharply increase over the first training epoch, then quickly flatline after about 2 to 3 epochs.A similar pattern in the loss was observed, where it decreases for the first few epochs, then begins to increase consistently as the magnitude of the weights increases.

Overall, it was found that the three deep learning models significantly out-performed the baselines across all metrics, but none of the three deep learning models stood out as being noticeably more expressive than the others. The LSTM was notable for having slightly better overall performance than the CNN, but being significantly faster to train and perform inference. Most surprising was that the hybrid model did not provide any gain in performance over the LSTM whatsoever, and in some ways may have actually underperformed it. Its possible that the encoding of the LSTM captured some amount of information that was otherwise lost during CNN filtering process.

Evaluation for extracting various features carried out which includes the concept of fuzzywuzzy and vector distances of the texts was hugely inspired from the presentation in Pydata, Berlin, Germany by Abhishek Thakur, regarding the Deep learning model[6] used in Question duplication analysis.

CHAPTER 3
METHODOLOGY

3.1 System Architecture and Overview

The overall architecture of our system is show below. For each question in the pairs, the system undergoes a series of data preprocessing, feature extraction and engineering steps. The features from the question pairs are fed into a feed forward neural network for training and testing the accuracies. Similarly, the resulting trained model is used to evaluate a new pair of questions, following the same set of procedures of preprocessing, feature extraction and engineering to test for duplication.

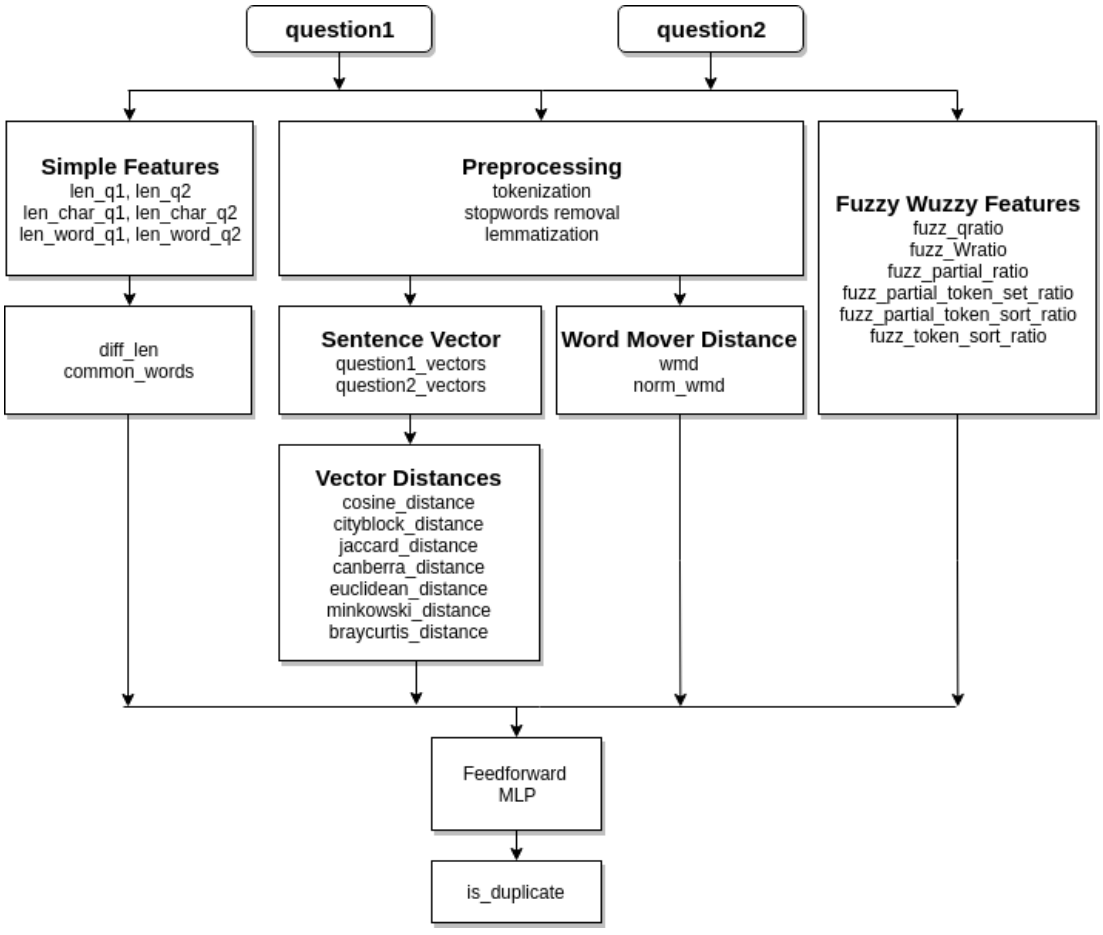

Figure 3.1: Overall System Architecture.

3.2 Data Acquisition and Dataset Description

Quora has given an (almost) real-world dataset[7] consising of a total of 404,290 question pairs in a tab-separated format; 255045 negative samples (non-duplicates) and 149306 positive samples (duplicates) with the label of *is_duplicate* along with every question pair. Positive samples comprises 40% of the total question pairs.

Target datasets can be downloaded from Quora. The training dataset contains valid question pairs , with labels:

- **id:** unique identifier for the question pair
- **qid1:** unique identifier for the first question
- **qid2:** unique identifier for the second question
- **question1:** full unicode text of the first question
- **quesition2:** full unicode text of the second question
- **is_duplicate:** label 1 if questions are duplicates, 0 otherwise

As a comparison, the test dataset contains 20% of the total question pairs, but without any "is_duplicate" label as a dataset for conducting further predictions on. The dataset was analyzed and the distribution was visualized in a count plot.

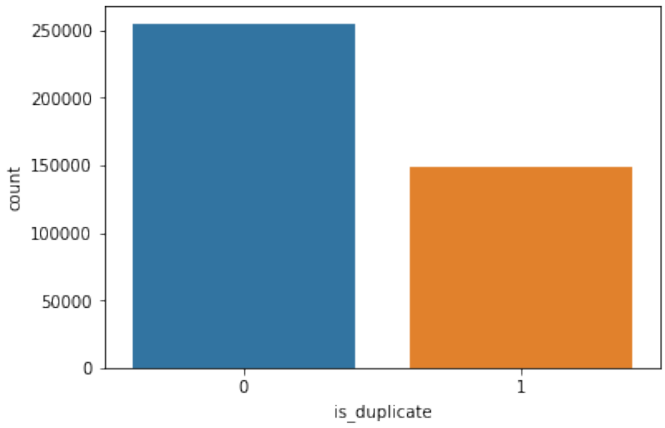

Figure 3.2: Distribution of the total question pairs.

3.3 Data Pre-processing

Insignificant columns; ‘id’, ‘qid1’, ‘qid2’ are dropped from the dataset since these have no contibution in the learning the semantics of the questions. **Tokenization** and **lemmatization** are performed for each question on the dataset. Similarly, question mark (?) and all the **stops words** in the dataset are removed.

3.3.1 Word Tokenization

Word tokenization is the process of splitting a large sample of text into words. This is a requirement in natural language processing tasks where each word needs to be captured and subjected to further analysis like classifying and counting them for a particular sentiment etc.

```
['what', 'is', 'the', 'step', 'by', 'step', 'guide', 'to', 'invest', 'in', 'share', 'market', 'i
n', 'india']
['what', 'is', 'the', 'story', 'of', 'kohinoor', 'koh', 'i', 'noor', 'diamond']
['how', 'can', 'i', 'increase', 'the', 'speed', 'of', 'my', 'internet', 'connection', 'while', 'u
sing', 'a', 'vpn']
```

Figure 3.3: Few questions after tokenization

3.3.2 Stopwords

A stop word is a commonly used word (such as the, a, an, in) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query. We would not want these words taking up space in our database, or taking up valuable processing time. For this, we can remove them easily, by storing a list of words that you consider to be stop words. NLTK in python has a list of stopwords stored in 16 different languages.

```
sentence = "What is the step by step guide to invest in share market in india?"
sentence = re.sub("[^a-zA-z]", " ", sentence.lower())
words = sentence.split()
sentence = [word for word in words if not word in stopwords.words("english")]
```

```
sentence
```

```
['step', 'step', 'guide', 'invest', 'share', 'market', 'india']
```

Figure 3.4: A question after stopword removal

3.3.3 Stemming and Lemmatization

For grammatical reasons, documents are going to use different forms of a word, such as organize, organizes, and organizing. Additionally, there are families of derivationally related words with similar meanings, such as democracy, democratic, and democratization. In many situations, it seems as if it would be useful for a search for one of these words to return documents that contain another word in the set.

The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. For instance:

```
ps = PorterStemmer()
sentence = """
How do I register institute providing training in various programming languages
for vocational courses as it will exempt it from the service tax?
"""
sentence = re.sub("[^a-zA-z]", " ", sentence.lower())
words = sentence.split()
sentence = [ps.stem(word) for word in words if not word in stopwords.words("english")]
```

```
sentence
```

```
['regist',
 'institut',
 'provid',
 'train',
 'variou',
 'program',
 'languag',
 'vocat',
 'cours',
 'exempt',
 'servic',
 'tax']
```

Figure 3.5: A question after stopword removal and stemming

```
wl = WordNetLemmatizer()
sentence = """
How do I register institute providing training in various programming languages
for vocational courses as it will exempt it from the service tax?
"""
sentence = re.sub("[^a-zA-z]", " ", sentence.lower())
words = sentence.split()
sentence = [wl.lemmatize(word) for word in words if not word in stopwords.words("english")]
```

```
sentence
```

```
['register',
 'institute',
 'providing',
 'training',
 'various',
 'programming',
 'language',
 'vocational',
 'course',
 'exempt',
 'service',
 'tax']
```

Figure 3.6: A question after stopword removal and lemmatization

3.4 Feature Extraction

The Machine Learning models as anticipated, is going to take in a lot of numerical values and vectors. For example, the number of times a word appears in a piece of text can be considered a feature. Several features are derived from the input text, and aggreggated on a per-text basis into feature vectors through Vector Space Modelling. Vector Space Modelling can largely be implemented via following techniques:

3.4.1 Traditional Bag of Words

A bag-of-words model is a way of extracting features from the text for the use in modeling. It is called a bag of words, because information about the order or structure of words in the document is discarded. This is only concerned with whether the known words occur in the document, not where in the document. Here, the length of document vector is equal to number of known words (vocabulary).

```
corpus = [
     'This is the first document.',
     'This document is the second document.',
     'And this is the third one.',
     'Is this the first document?',
]


['and', 'document', 'first', 'is', 'one', 'second', 'the', 'third', 'this']


[[0 1 1 1 0 0 1 0 1]
 [0 2 0 1 0 1 1 0 1]
 [1 0 0 1 1 0 1 1 1]
 [0 1 1 1 0 0 1 0 1]]
```

Figure 3.7: Bag of words model for a small corpus.

3.4.2 Term Frequency Inverse-Document Frequency

TF-IDF [8] is a reflect how important a word is to a document in a collection or corpus. One approach is to rescale the frequency of words by how often they appear in all the documents. Frequent words like is, a, the, etc are penalized. This approach to scoring is called Term FrequencyInverse Document Frequency, where:

$$W_{i,j} = tf_{i,j} * \log \frac{N}{df_i}$$

Where:

- $tf_{i,j}$: number of occurance of i in j
- df_i: is the number of documents containing i
- N: is the total number of documents

```
tfidf['what']
```

```
1.9399247964366455
```

```
tfidf['lorem']
```

```
12.99359822607572
```

Figure 3.8: TF-IDF of words 'what' and 'lorem'

- Term Frequency: It is the scoring of the frequency of the word in the document.
- Inverse Document Frequency: It is the scoring of how rare the words are across the current documents.

3.4.3 Word Embedding: word2vec

Word embedding is a collective name for a set of feature learning techniques and language modeling in Natural Language Processing. Here words and phrases are mapped into vectors of real numbers. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2vec [9] takes input a very large corpus of text and produces a vector space, which consist of several hundred dimensions, with each unique word in the corpus assigned equivalent vector in the space. Word vectors are positioned in the corresponding vector space such that words that share common contexts in the body are located in near to one another in the space.

Continuous Bag of Word Model

CBOW is learning to predict the word by the context. A context may be single word or multiple word for a given target words. Lets see this by an example The cat jumped over the puddle. So one approach is to treat The, cat, over, the, puddle as a context and from these words, be able to predict or generate the center word jumped. This type of model we call a Continuous Bag of Words Model.

Skip-gram Model

Another approach is to create a model such that given the center word jumped, the model will be able to predict or generate the surrounding words The, cat, over, the, puddle. Here we call the word jumped the context. We call this type of model a Skip-Gram model.

```
king = model.wv['king']
```

```
man = model.wv['man']
```

```
woman = model.wv['woman']
```

```
print(king)
print(man)
print(woman)
```

```
[-2.5829005   1.9179722   1.146964    0.7492415  -0.6000157  -2.8341634
 -2.8278897  -0.731253   -0.02157602  0.6726361  -0.16747534  1.263601
  1.2125832  -2.1578298  -0.5957854   1.279929   -1.4611832   0.14727864
 -0.40092555  2.397424   -2.579857    0.7057467  -1.0831238  -1.5788301
  0.47483215 -3.8856914  -0.3877771  -0.49705458 -3.5843647  -1.1941504
  0.18059061  0.07606576]
[-4.222011   -0.17043875  3.9976332   4.443278   -0.01917488  0.2033211
  0.5469626  -2.0805604  -1.2902623  -2.6193213  -1.5731192  -1.591861
  2.0518582   0.4749743  -0.5929095   0.57914406 -2.8458517   1.8716403
 -0.87846935  1.5925093  -2.500882   -2.345689   -3.243591   -0.17216216
 -1.4763216  -1.0502725  -1.0638894   3.3106458  -4.143433   -2.6550996
  5.3808413   1.4558139 ]
[-3.237001    0.84142005  2.3160563   5.4998603  -1.4409931   0.3867548
  0.42073995 -2.1927316  -0.7462781  -3.2965071  -2.287409   -2.848653
  1.235986    1.3379674  -1.140569   -0.66565263 -4.312319    3.8630867
 -0.26279888  0.58208823 -2.5978756  -1.7554814  -4.4215083   1.1114552
 -1.7188497   0.0101186  -2.9609072   4.4982452  -4.3487043  -2.5484457
  5.4881473   0.5893974 ]
```

```
unknown = king - man + woman
print(unknown)
```

```
[-1.5978904   2.929831   -0.53461313  1.8058238  -2.021834   -2.65073
 -2.9541123  -0.8434242   0.5224082  -0.00454974 -0.88176525  0.006809
  0.396711   -1.2948368  -1.1434448   0.03513235 -2.9276505   2.138725
  0.21474493  1.387003   -2.6768508   1.2959543  -2.2610412  -0.29521275
  0.2323041  -2.8253005  -2.2847948   0.69054484 -3.789636   -1.0874965
  0.28789663 -0.79035074]
```

```
model.similar_by_vector(unknown)
```

```
[('prince', 0.8524366021156311),
 ('king', 0.8297630548477173),
 ('lee', 0.8151587247848511),
 ('elizabeth', 0.7999888062477112),
 ('queen', 0.7999544143676758),
 ('abraham', 0.7891779541969299),
 ('george', 0.7809760570526123),
 ('jr', 0.7792613506317139),
 ('bruce', 0.7725250124931335),
 ('mary', 0.7633032202720642)]
```

Figure 3.9: Word2vec model showing similar words of an unknown vector with semantic context

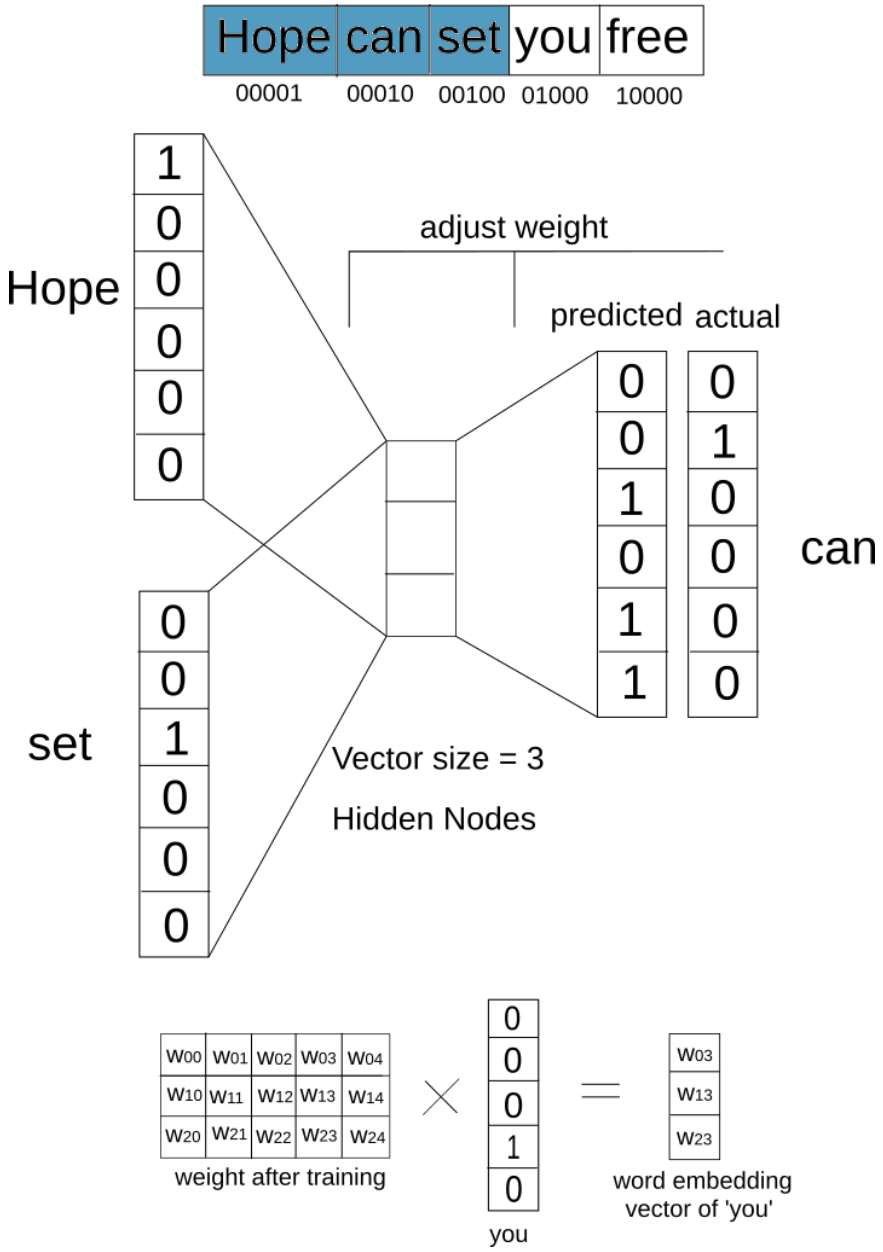

Figure 3.10: CBOW model

3.4.4 Google News vectors

The repository provided by the Google consist of word2vec pre-trained Google News corpus word vector model which consists of 3 million 300 dimension English word vectors. Google News Vector includes word vectors for a vocabulary of 3 million words and phrases that are trained on around 100 billion words from a Google News dataset. We require a high amount of memory to process this because gensim allocates a big matrix to hold all of the word vectors, and if we do the math, this is a huge matrix.

Mathematically, 3 million words * 300 features * 4bytes/feature = 3.35GB

3.5 Feature Engineering

Feature engineering involves extracting of information from the given dataset. Features are divided into four categories. This involves the working of basic *NLTK mathematics*, *fuzywuzzy parameters*, *Word Mover Distance* and *Vector distance*.

3.5.1 NLTK Library Groundworks

The simple features: *len_q1, len_q2, diff_len, len_char_q1, len_char_q2, len_word_q1, len_word_q2 and common_words* are derived from the dataset.

	question1	question2	is_duplicate	len_q1	len_q2	diff_len	len_char_q1	len_char_q2	len_word_q1	len_word_q2
0	What is the step by step guide to invest in sh...	What is the step by step guide to invest in sh...	0	66	57	9	20	20	14	12
1	What is the story of Kohinoor (Koh-i-Noor) Dia...	What would happen if the Indian government sto...	0	51	88	-37	21	29	8	13

Figure 3.11: Simple feature extraction.

3.5.2 Fuzzy wuzzy Features

Fuzzy String Matching is sometimes known as approximate string matching. It is the process of finding strings that approximately match a given pattern. It uses Levenshtein Distance to calculate the differences between sequences in a simple-to-use package. **The closeness of a match is measured in terms of edit distance, which is the number of primitive operations (insertion, deletion and substitution) necessary to convert the string into an exact match.** *Fuzzy QRatio, Fuzzy WRatio, Fuzzy partial ratio, Fuzzy partial token set ratio, Fuzzy partial token sort ratio, Fuzzy token set ratio and Fuzzy token sort ratio* are extracted from the questions pairs.

Simple Ratio	*"this is a test", "this is a test!"*	97
Partial Ratio	*"this is a test", "this is a test!"*	100
Token Sort Ratio	*"fuzzy wuzzy was a bear", "wuzzy fuzzy was a bear"*	100
Token Set Ratio	*"fuzzy was a bear", "fuzzy fuzzy was a bear"*	100

Table 3.1: Fuzzy Wuzzy sample features

fuzz_qratio	fuzz_WRatio	fuzz_partial_ratio	fuzz_partial_token_set_ratio	fuzz_partial_token_sort_ratio	fuzz_token_set_ratio	fuzz_token_sort_ratio
93	95	98	100	89	100	93
66	86	73	100	75	86	63

Figure 3.12: Fuzzy feature extraction.

3.5.3 Word Mover Distance

WMD is a method that allows us to assess the distance between two texts, even when they have no words in common. It uses *word2vec* vector embeddings of words. Normalized WMD, on the other hand, uses the Euclidean distance for calculation. To use WMD, we need some word embeddings. Gensim is very helpful for this purpose.

3.5.4 Vector distances

Various vector distances: *Cosine, Cityblock, Canberra, Euclidean, Minkowski and Braycurti distances* are measured for all the question pairs. These distances are plotted against common words which gives the similar plot.

Some Distance Metrics

- **Cosine Distance:** Cosine similarity is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them.

$$similarity = cos(\theta) = \frac{A.B}{|A||B|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

- **Euclidean Distance:** Euclidean distance is the 'ordinary' straight-line distance between two points in Euclidean space.

$$d(p,q) = d(q,p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + ... + (q_n - p_n)^2}$$

$$\sqrt{\sum_{i=1}^{n}(q_n - p_n)^2}$$

- **Minkowski Distance:** The Minkowski distance is a metric in a normed vector space which can be considered as a generalization of both the Euclidean distance and the Manhattan distance. The Minkowski distance of order p between two points:

$$D(X,Y) = (\sum_{i=1}^{n} |x_i - y_i|^p)^{1/p}$$

question1	question2	is_duplicate	wmd	norm_wmd	cosine_distance	cityblock_distance	euclidean_distance
What is the step by step guide to invest in sh...	What is the step by step guide to invest in sh...	0	0.564615	0.217555	0.037908	3.774843	0.275348
What is the story of Kohinoor (Koh-i-Noor) Dia...	What would happen if the Indian government sto...	0	3.772346	1.368796	0.574596	15.130415	1.072004

Figure 3.13: Some vector distances between questions in pair

All the features were combined into a single space, and then, the mathematical implication was realized.

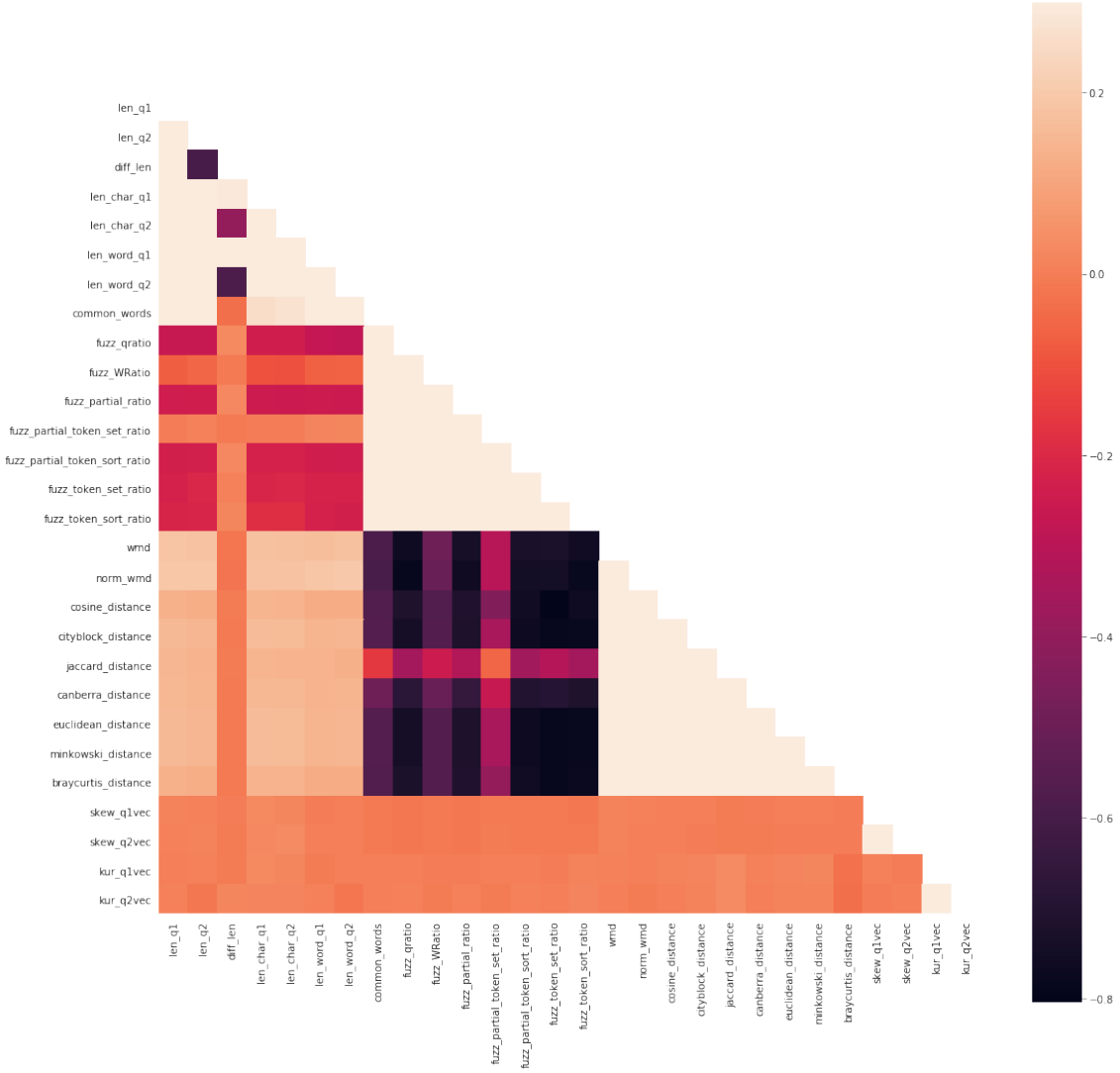

Figure 3.14: Correlation Heatmap of features.

3.6 Analysis and Visualization

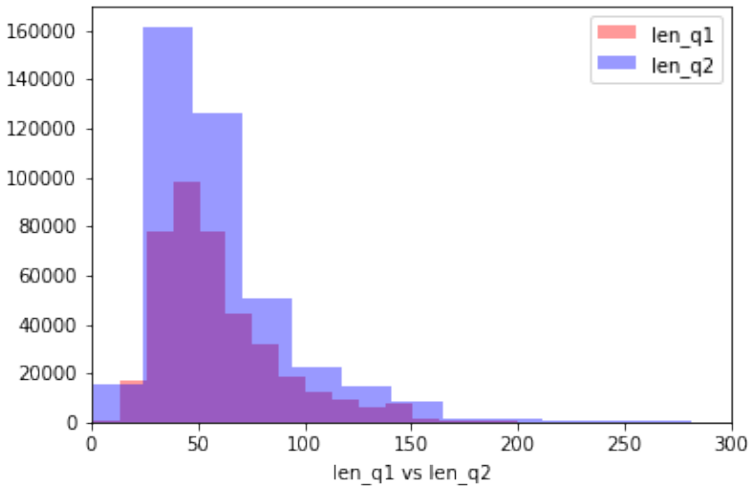

Figure 3.15: Distribution of lengths of question pairs

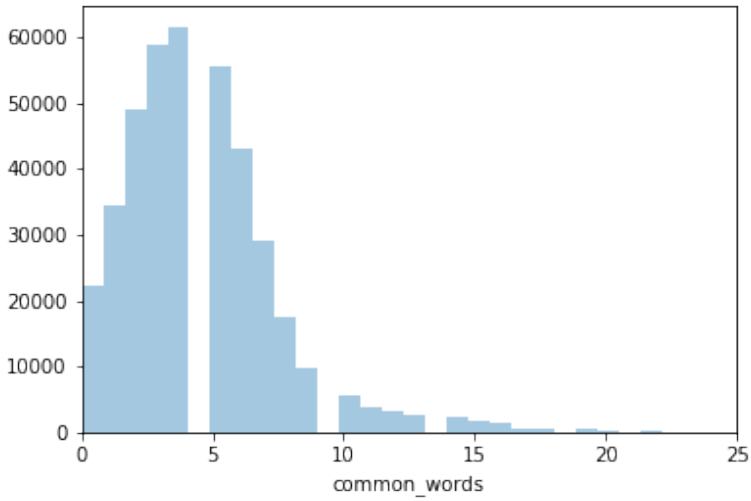

Figure 3.16: Distribution of common words of question pairs

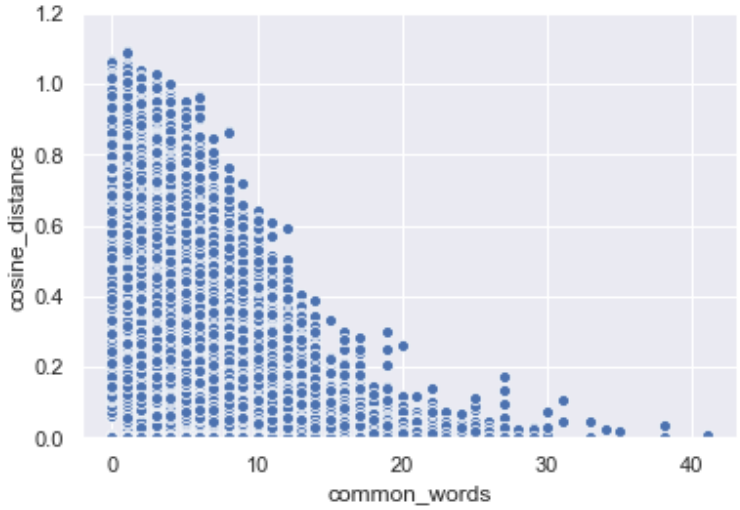

Figure 3.17: Scatter plot of common words and cosine distance of question pairs

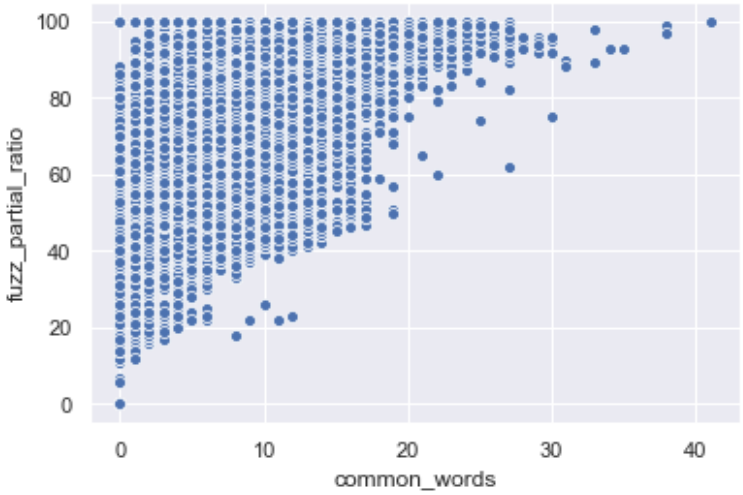

Figure 3.18: Scatter plot of common words and fuzz partial ratio of question pairs

3.7 Supervised Machine Learning Models

For the validation of feature-engineered parameters, application of different supervised machine learning algorithms is possible along with the study of their accuracy and results for few parameters. The supervised machine learning algorithms under considerations via *scikit-learn* library are:

S.N.	Supervised Machine Learning	Accuracy
1	Random Forest	0.7235
2	K Nearest Neighbors	0.7104
3	Logistic Regression	0.6680

Table 3.2: Comparison of Supervised models for duplication analysis

3.8 Neural Network Design

Artificial neural networks (ANN) or connectionist systems are computing systems vaguely inspired by the biological neural networks and astrocytes that constitute animal brains. The neural network itself is not an algorithm, but rather a framework for many different machine learning algorithms to work together and process complex data inputs. Such systems ”learn” to perform tasks by considering examples, generally without being programmed with any task-specific rules.

An ANN is based on a collection of connected units or nodes called artificial neurons, which loosely model the neurons in a biological brain. Each connection, like the synapses in a biological brain, can transmit a signal from one artificial neuron to another. An artificial neuron that receives a signal can process it and then signal additional artificial neurons connected to it.

In the brain, neurons can be connected to many other neurons nearby. In a typical neural network, however, information only flows one way. These neurons are spread across three layers:

1. The input layer consists of the neurons that do nothing more than receive the data and pass it on. The number of neurons in the input layer should be equal to the

number of features in your data set.

2. The output layer consists of a number of nodes depending on the type of model youre building. In a classification system, there will be one node for each type of label you might be applying, while in a regression system there will just be a single node that puts out a value.
3. In between these two layers is where things get interesting. Here, we have whats called the hidden layer, which also consists of a number of neurons (how many will depend on the number of neurons in your input and output layers, but dont worry about that). The nodes in the hidden layer apply transformations to the inputs before passing them on. As the network is trained, those nodes that are found to be more predictive of the outcome are weighted more heavily.

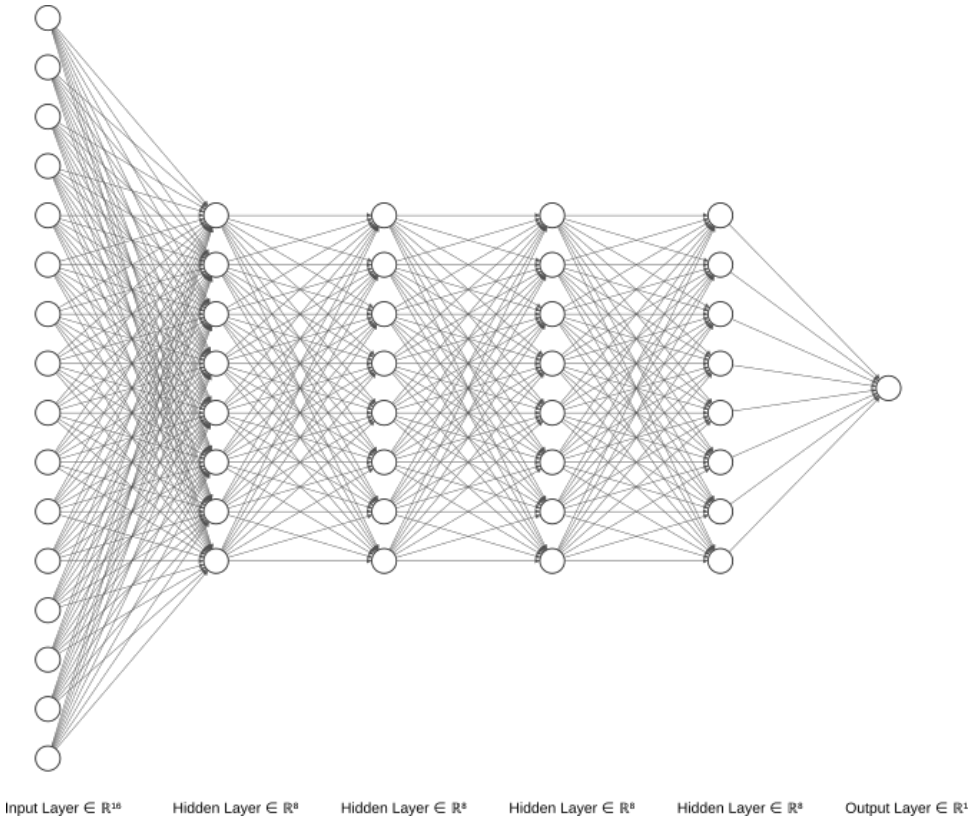

Figure 3.19: Architecture of Neural Network

3.8.1 Framework of Artificial Neural Network:

1. Assign random weights to all linkages to start the algorithm.
2. Using inputs and the (input - hidden node) linkages, find the activation rate of the hidden nodes.

3. Using activation rate of the hidden nodes and linkages to output, find the activation rate of output nodes.
4. Find the error rate at the output node and recalibrate all the linkages between hidden nodes and output nodes.
5. Using weights and error found at the output node, cascade down the error to hidden nodes.
6. Recalibrate the weights between hidden node and the input nodes.
7. Repeate the process till the convergence criterion is met.
8. Using the final weights, score the activation rate of the output node.

After completion of feature engineering, the parameters extracted are now tested to evaluate how these parameters affect the performance of our neural network. 16 best parameters are chosen through the analysis of correlation heat-map as show in figure 3.14. Artificial Neural Network is designed consisting of 5 hidden layers. Input layer consists of 16 nodes that send information to the hidden layer. The exact artificial neural network of our model, consist of an input layer with 16 input nodes, 5 hidden layers with 8 nodes each for precise result calculation and a single output node that results 1 or 0 i.e. duplicate or not_duplicate.

The nodes in the input layer and the hidden layers have ReLU as the activation function, while the output node has sigmoid function as the activation fucntion.

Mathematically, ReLU it is defined as y = max(0, x). Visually, it looks like the following:

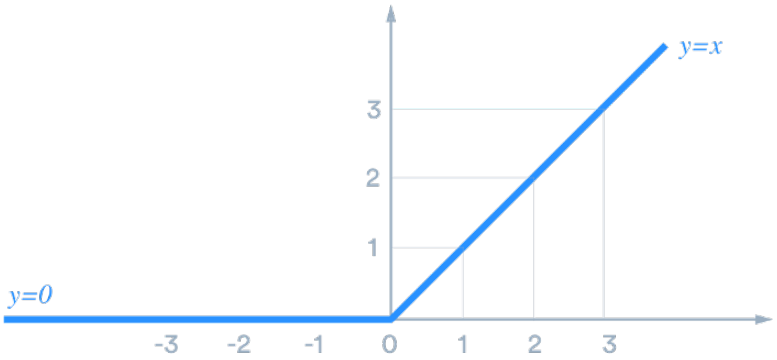

Figure 3.20: ReLU activation function

ReLU is linear (identity) for all positive values, and zero for all negative values. This means that:

1. Its cheap to compute as there is no complicated math. The model can therefore take less time to train or run.
2. It converges faster. Linearity means that the slope doesnt plateau, or saturate, when x gets large. It doesnt have the vanishing gradient problem suffered by other activation functions like sigmoid or tanh.
3. Its sparsely activated. Since ReLU is zero for all negative inputs, its likely for any given unit to not activate at all.

Many natural processes, such as those of complex system learning curves, exhibit a progression from small beginnings that accelerates and approaches a climax over time. When a specific mathematical model is lacking, a sigmoid function is often used.Often, sigmoid function refers to the special case of the logistic function shown in the figure 3.21 and defined by the formula:

$$S(X) = \frac{1}{1 + e^{-x}}$$

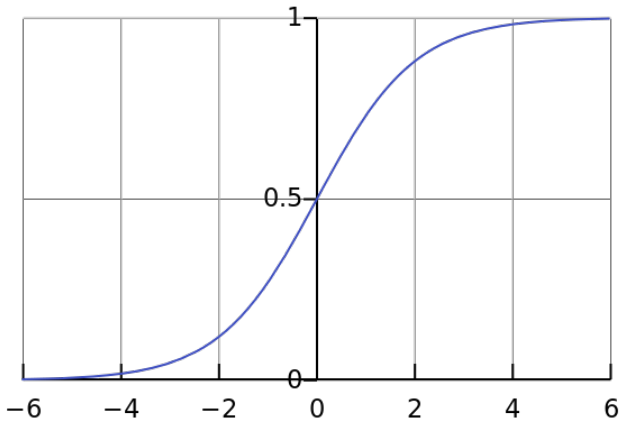

Figure 3.21: Sigmoid Activation Function

3.8.2 Adam optimization

Adam is an optimization algorithm that can used instead of the classical stochastic gradient descent procedure to update network weights iterative based in training data. Stochastic gradient descent maintains a single learning rate (termed alpha) for all weight updates and the learning rate does not change during training. The method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients. Adam can be looked at as a combination of Root Mean Square Propagation and Stochastic Gradient Descent with momentum. It uses the squared gradients to scale the learning rate like Root Mean Square Propagation and it takes advantage of momentum by using moving average of the gradient instead of gradient itself like Stochastic Gradient Descent with momentum.

Adam Configuration Parameters

- alpha. Also referred to as the learning rate or step size. The proportion that weights are updated (e.g. 0.001). Larger values (e.g. 0.3) results in faster initial learning before the rate is updated. Smaller values (e.g. 1.0E-5) slow learning right down during training
- beta1. The exponential decay rate for the first moment estimates (e.g. 0.9).
- beta2. The exponential decay rate for the second-moment estimates (e.g. 0.999). This value should be set close to 1.0 on problems with a sparse gradient (e.g. NLP and computer vision problems).
- epsilon. Is a very small number to prevent any division by zero in the implementation (e.g. 10E-8).

3.8.3 Training

We used 20% of dataset as testing dataset and remaining 80% as training dataset. Input parameter are: diff_len, common_words, fuzz_qratio ,fuzz_WRatio, fuzz_partial_ratio, fuzz_partial_token_set_ratio, fuzz_partial_token_sort_ratio, fuzz_token_set_ratio, fuzz_token_sort_ratio, wmd, cosine_distance, cityblock_distance, canberra_distance, eu-

clidean_distance, minkowski_distance and braycurtis_distance. The batch size is 10 and 30 epochs are used for training of our model. Batch size defines the number of samples that are going to be propagated each time through the network.

```
Epoch 1/30
314651/314651 [==============================] - 40s 127us/step - loss: 0.5243 - acc: 0.6863
Epoch 2/30
314651/314651 [==============================] - 40s 128us/step - loss: 0.5204 - acc: 0.6894
Epoch 3/30
314651/314651 [==============================] - 42s 133us/step - loss: 0.5185 - acc: 0.6916
Epoch 4/30
314651/314651 [==============================] - 40s 128us/step - loss: 0.5151 - acc: 0.6940
Epoch 5/30
314651/314651 [==============================] - 41s 129us/step - loss: 0.5131 - acc: 0.6953
Epoch 6/30
314651/314651 [==============================] - 40s 127us/step - loss: 0.5121 - acc: 0.6956
Epoch 7/30
314651/314651 [==============================] - 40s 129us/step - loss: 0.5116 - acc: 0.6963
Epoch 8/30
314651/314651 [==============================] - 41s 129us/step - loss: 0.5110 - acc: 0.6978
Epoch 9/30
314651/314651 [==============================] - 40s 129us/step - loss: 0.5105 - acc: 0.6988
Epoch 10/30
314651/314651 [==============================] - 40s 128us/step - loss: 0.5101 - acc: 0.6995
Epoch 11/30
314651/314651 [==============================] - 41s 131us/step - loss: 0.5058 - acc: 0.7100
Epoch 12/30
314651/314651 [==============================] - 40s 128us/step - loss: 0.5002 - acc: 0.7186
Epoch 13/30
314651/314651 [==============================] - 40s 127us/step - loss: 0.4994 - acc: 0.7201
Epoch 14/30
314651/314651 [==============================] - 40s 127us/step - loss: 0.4990 - acc: 0.7207
Epoch 15/30
314651/314651 [==============================] - 40s 126us/step - loss: 0.4985 - acc: 0.7209
Epoch 16/30
314651/314651 [==============================] - 40s 126us/step - loss: 0.4974 - acc: 0.7226
Epoch 17/30
314651/314651 [==============================] - 40s 127us/step - loss: 0.4964 - acc: 0.7241
Epoch 18/30
314651/314651 [==============================] - 40s 128us/step - loss: 0.4962 - acc: 0.7243
Epoch 19/30
314651/314651 [==============================] - 41s 131us/step - loss: 0.4959 - acc: 0.7248
Epoch 20/30
314651/314651 [==============================] - 40s 128us/step - loss: 0.4957 - acc: 0.7248
Epoch 21/30
314651/314651 [==============================] - 40s 127us/step - loss: 0.4956 - acc: 0.7251
Epoch 22/30
314651/314651 [==============================] - 40s 127us/step - loss: 0.4954 - acc: 0.7247
Epoch 23/30
314651/314651 [==============================] - 40s 128us/step - loss: 0.4952 - acc: 0.7248
Epoch 24/30
314651/314651 [==============================] - 48s 154us/step - loss: 0.4952 - acc: 0.7247
Epoch 25/30
314651/314651 [==============================] - 41s 131us/step - loss: 0.4951 - acc: 0.7251
Epoch 26/30
314651/314651 [==============================] - 42s 133us/step - loss: 0.4950 - acc: 0.7247
Epoch 27/30
314651/314651 [==============================] - 43s 137us/step - loss: 0.4948 - acc: 0.7250
Epoch 28/30
314651/314651 [==============================] - 41s 130us/step - loss: 0.4947 - acc: 0.7251
Epoch 29/30
314651/314651 [==============================] - 41s 130us/step - loss: 0.4947 - acc: 0.7252
Epoch 30/30
314651/314651 [==============================] - 41s 131us/step - loss: 0.4945 - acc: 0.7256
```

Figure 3.22: Training of ANN

3.9 Software Development Model

Our development approach is based upon **'Test Driven Development'**, that relies on the repetition of a very short development cycle: our requirements such as feature extraction, feature engineering, model development are turned into very specific test cases, then the program is improved to pass the new tests of required results. *This is opposed to software development that allows software to be added that is not proven to meet requirements.*

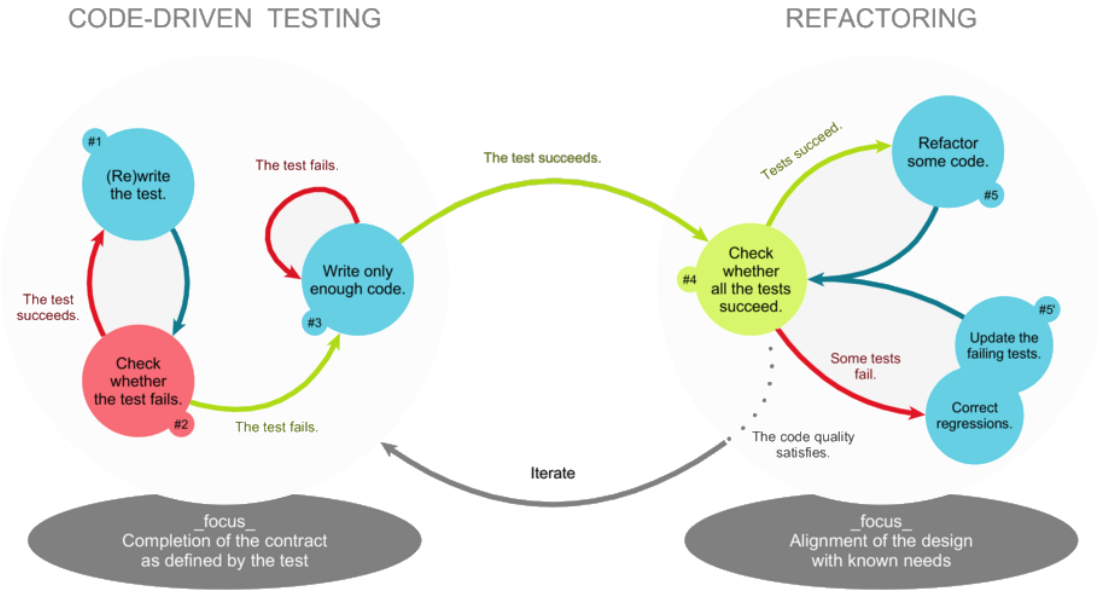

Figure 3.23: Test Driven Development Model

Source: https://en.wikipedia.org/wiki/Test-driven_development

3.9.1 Sequence based on Test-Driven Development by Example:

1. **Add a test:**

   In test-driven development, each new feature begins with writing a test. Write a test that defines a function or improvements of a function, which should be very succinct. To write a test, we as a developer must clearly understand the feature's specification and requirements.

2. **Run all tests and see if the new test fails:**

   This validates that the test harness is working correctly, shows that the new test

does not pass without requiring new code because the required behavior already exists, and it rules out the possibility that the new test is flawed and will always pass. The new test should fail for the expected reason. This step increases the our confidence in the new test.

3. **Write the code:**

   The next step is to write some code that causes the test to pass. The new code written at this stage is not perfect and may pass the test in an inelegant way. That is acceptable because it will be improved and honed in Step 5.

4. **Run tests:**

   If all test cases now pass, we can be confident that the new code meets the test requirements, and does not break or degrade any existing features. If they do not, the new code must be adjusted until they do.

5. **Refactor code:**

   The growing code base must be cleaned up regularly during test-driven development. New code can be moved from where it was convenient for passing a test to where it more logically belongs. Duplication must be removed. Object, class, module, variable and method names should clearly represent their current purpose and use, as extra functionality is added.

6. **Repeat**

CHAPTER 4

OUTPUT

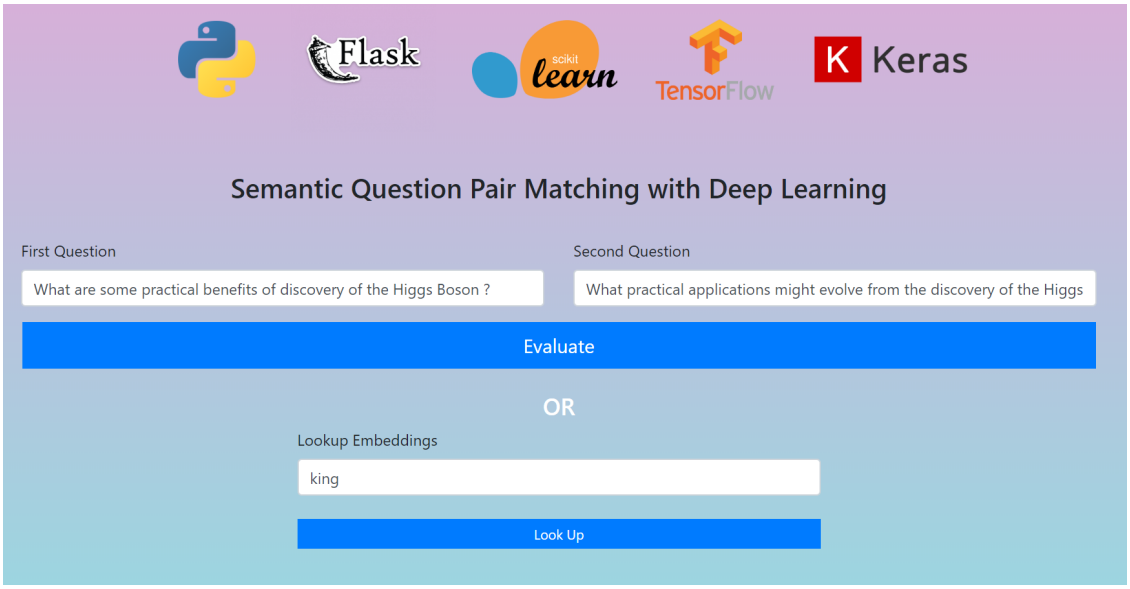

Figure 4.1: Home page to evaluate a pair of questions or lookup similar words.

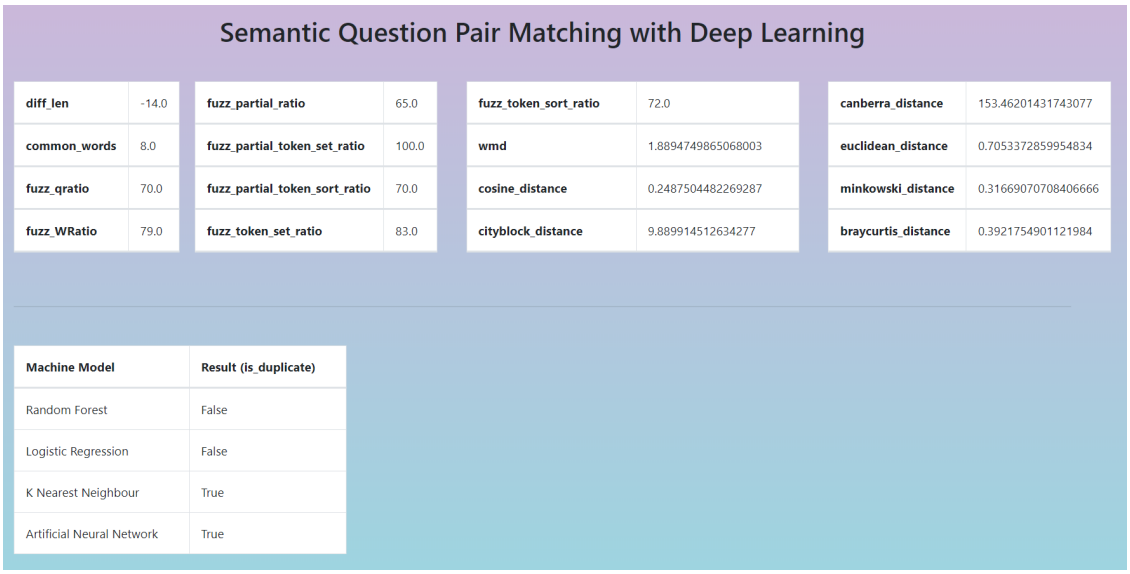

Figure 4.2: Evaluating a pair of questions.

Words related to king:

#	Words	Cosine Similarity	Vector
1	jackson	0.9170856475830078	[-1.7164836 1.8294125 0.32592058 -0.40480655 -0.08339158 -1.6826427 -1.7304238 -0.49248657 -0.1676851 -0.06072393 0.11962742 1.3668276 0.90763265 -1.0506763 -0.5218164 0.7036808 -1.55533 0.63607806 -0.09730083 1.8054376 -1.8210206 1.0017585 -0.66110635 -1.9702251 0.30722475 -3.1247196 1.1484314 0.8200226 -2.7029994 -0.95896465 -0.4110208 -0.56016797]
1	prince	0.916680097579956	[-0.5889641 1.0294877 0.47653213 0.34074122 0.02304216 -0.82068247 -0.60913444 -0.13486323 -0.24660684 0.15537524 -0.04742643 0.40668535 0.65379345 -0.6942042 -0.29548606 0.5446437 -0.7581655 0.34339455 -0.02819815 0.84811175 -0.77211136 0.66940707 -0.77531224 -0.06260004 -0.23046036 -1.5743388 -0.3553846 0.39698267 -1.5661048 -0.5224344 0.01328079 0.14272085]
1	robert	0.9032487273216248	[-2.4331644 0.88121885 1.0748357 -0.863973 -0.43201208 -1.6329553 -2.0719585 0.11564737 -0.11405259 0.5089016 0.5130483 1.284562 0.9737573 -1.5380981 -0.79329115 0.56025636 -0.9769954 0.55508566 -0.6160865 1.65429 -1.8471282 0.33202523 -0.8405775 -0.09796639 0.5280329 -2.6113956 -0.50730187 -0.18148285 -1.6998575 -0.11195284 0.03041785 -1.231389]
1	princess	0.9025406837463379	[-0.75034595 0.6662391 0.17670667 0.06551352 -0.0070325 -0.95871717 -0.8093771 -0.03449096 -0.02144096 -0.5880667 0.04166729 0.5462285 0.7932926 -0.59016585 -0.04916194 0.484042 -0.38953274 0.07132016 -0.4153422 1.0794101 -0.7474116 0.33096075 -0.09078473 -0.5637059 0.11742189 -1.3029588 0.28489834 0.14683394 -1.0375118 -0.83775455 0.52181613 -0.00960252]
1	queen	0.8957031965255737	[-2.3368468 0.56403255 1.2188283 0.88877916 -0.56647384 -2.5451996 -1.7815655 0.45772582 0.37839198 0.9612027 -0.8821726 0.46528545 0.72048724 -1.7655395 -1.4997103 0.50216806 -1.0658889 -0.21059665 -0.5104892 0.99170005 -1.3849756 0.7045598 -0.39090183 -0.8171422 -0.2892701 -2.3560243 -0.98737466 0.5163146 -3.0432348 -1.7552925 0.69047606 -0.98238117]
1	lee	0.8944235444068909	[-1.3471355 1.8021047 0.29679886 0.69539064 -0.60864884 -1.0214051 -1.3482727 -0.0549046 0.52168715 -0.61433184 0.16619708 0.6626704 0.35210207 -0.39707842 -1.3058664 0.67429984 -0.8247559 0.16933796 -0.5037662 1.699125 -0.99733186 0.54908943 -1.0015546 -2.0125577 -0.15905334 -2.8170931 -0.52599645 0.06383558 -3.0889134 -0.46590734 -0.0317693 -0.23081084]
1	jr	0.8937932252883911	[-1.2810696 1.3038214 0.19196847 -0.6937162 -0.2702044 -0.94713956 -1.7343367 -0.90979344 -0.02841623 -0.25282878 -0.17654832 0.77701527 0.7947526 -0.9995183 0.49875268 0.6822012 -0.4965229 0.48641548 0.18260807 0.7456698 -0.5552112 0.5703377 -0.86355495 -0.79701334 0.03770216 -2.351846 -0.67782015 -0.00948282 -1.6861732 -0.68987983 -0.1615522 -0.09588674]
1	henry	0.8900725841522217	[-0.69552875 0.35889998 0.25512025 0.14466994 -0.01004068 -0.6378789 -0.8375622 0.31665844 -0.05821016 -0.11429265 0.0682332 0.8304073 0.43358138 -0.6647132 0.13055694 0.43622643 -0.5813792 0.23624226 -0.23708269 0.7274418 -0.40872994 0.7429334 -0.60793847 -0.2620143 0.1800267 -1.6076065 -0.05279432 0.33003125 -1.0892603 -0.48642656 -0.2667088 -0.1515848]
1	abraham	0.8835678100585938	[-0.9950697 0.7653032 0.01909612 0.23608783 -0.33849132 -0.7518092 -1.5344894 0.24661376 0.5391564 0.9280216 0.35056022 0.56232315 0.67872554 -0.77419806 -0.24629022 1.1739826 -0.7608316 0.14465691 0.14417253 0.7685522 -0.4075629 0.86533505 -0.6065444 -0.11334624 0.22445689 -2.0236216 -0.7044581 0.02418419 -1.4125146 -0.7318153 0.2062156 -0.09157664]
1	luther	0.8791694045066833	[-1.4895452 1.2473191 0.00613075 -0.5220099 -0.47997975 -0.6117198 -1.5259736 -0.5456508 0.02430855 0.4774176 0.24506676 0.8787376 1.2112114 -0.8461511 0.2915254 0.70703256 -0.5606873 0.39349508 -0.05647301 0.64538515 -0.38343522 0.88363755 -0.73476446 -0.13504884 0.30875263 -2.2222042 -0.3979941 0.0756369 -1.700901 -0.6677542 -0.26834327 0.06512772]

Figure 4.3: Lookup similar words from word2vec model.

CHAPTER 5
RESULTS AND ANALYSIS

5.1 Confusion Matrix

A Confusion Matrix is a popular representation of the performance of classification models. The matrix (table) shows us the number of correctly and incorrectly classified examples, compared to the actual outcomes (target value) in the test data. One of the advantages of using confusion matrix as evaluation tool is that it allows more detailed analysis (such as if the model is confusing two classes), than simple proportion of correctly classified examples (accuracy) which can give misleading results if the dataset is unbalanced (i.e. when there are huge differences in number of between difference classes).

The matrix is n by n, where n is the number of classes. The simplest classifiers, called binary classifiers, has only two classes: positive/negative, yes/no, male/female Performance of a binary classifier is summarized in a confusion matrix that cross-tabulates predicted and observed examples into four options:

- True Positive (TP): Correctly predicting a label (we predicted yes, and its yes),
- True Negative (TN): Correctly predicting the other label (we predicted no, and its no),
- False Positive (FP): Falsely Predicting a label (we predicted yes, but it's no),
- False Negative (FN): Missing and incoming label (we predicted no, but its yes).

The corresponding table of confusion, for the duplicate and non-duplicate, would be as shown:

	Predicted Values		
Acutal Values	*is_duplicate*	0	1
	0	36462	12792
	1	8954	20455

Table 5.1: Corresponding Confusion Matrix of ANN Model

Measure	Value	Derivations
Sensitivity	0.6955	TPR = TP / (TP + FN)
Specificity	0.7403	SPC = TN / (FP + TN)
Precision	0.6152	PPV = TP / (TP + FP)
Negative Predictive Value	0.8028	NPV = TN / (TN + FN)
False Positive Rate	0.2597	FPR = FP / (FP + TN)
False Discovery Rate	0.3848	FDR = FP / (FP + TP)
False Negative Rate	0.3045	FNR = FN / (FN + TP)
Accuracy	0.7236	ACC = (TP + TN) / (P + N)
F1 Score	0.6529	F1 = 2TP / (2TP + FP + FN)

Table 5.2: Summary of Confusion Matrix

5.2 Accuracy and Loss Functions

In running our experiments, a sample batch size of 30 was used and trained for a total of 10 epochs. Adam optimization was used with a learning rates as shown in table below. Binary cross entropy was chosen for the loss. All of the implementation was written in Keras with TensorFlow backend, with separate run scripts for training and evaluation. Across all three of the models of experiments studied, a similar pattern of convergence was found. The model accuracy would sharply increase over the few training epoch, then quickly flatline after about 13 to 15 epochs. A similar pattern in the loss was observed, where it decreases for the first few epochs, then begins to flatten consistently as the magnitude of the accuracy increases.

S.N.	Learning Rate	Epoch	Batch Size	Training Accuracy
1	0.01	30	10	0.6264
2	0.005	30	10	0.6938
3	0.001	30	10	0.7256

Table 5.3: Training ANN at different learning rate.

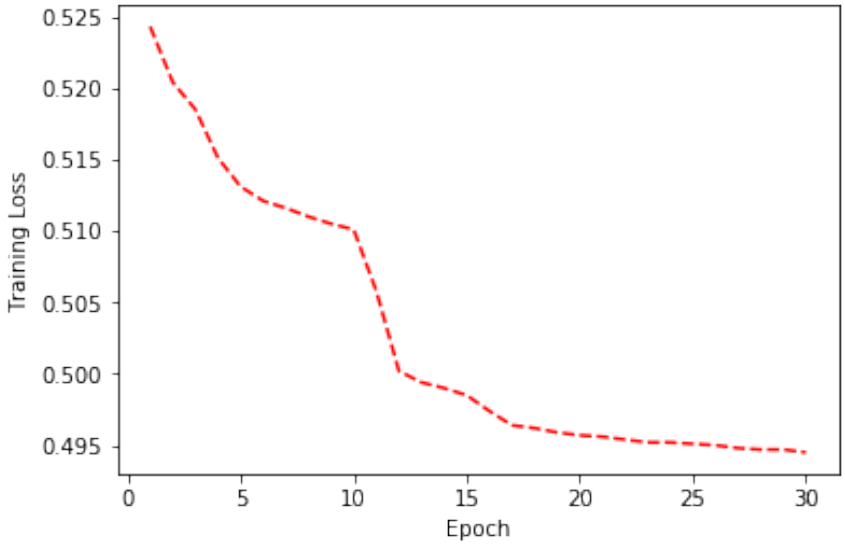

Figure 5.1: Training Loss of ANN

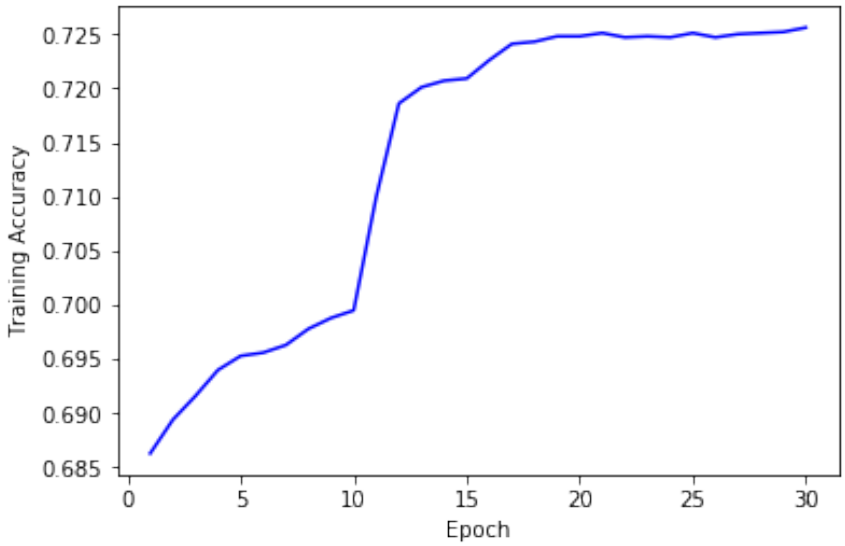

Figure 5.2: Training Accuracy of ANN

5.3 Receiver Operating Characteristic Curve

ROC curves typically feature true positive rate on the Y axis, and false positive rate on the X axis. This means that the top left corner of the plot is the ideal point - a false positive rate of zero, and a true positive rate of one. This is not very realistic, but it does mean that a larger area under the curve (AUC) is usually better.

The steepness of ROC curves is also important, since it is ideal to maximize the true positive rate while minimizing the false positive rate.

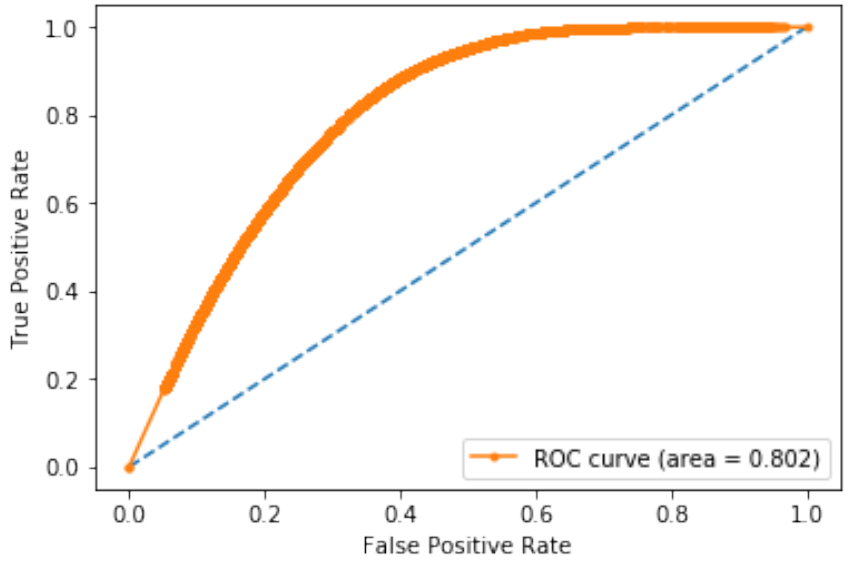

Figure 5.3: ROC Curve

5.4 Work Schedule

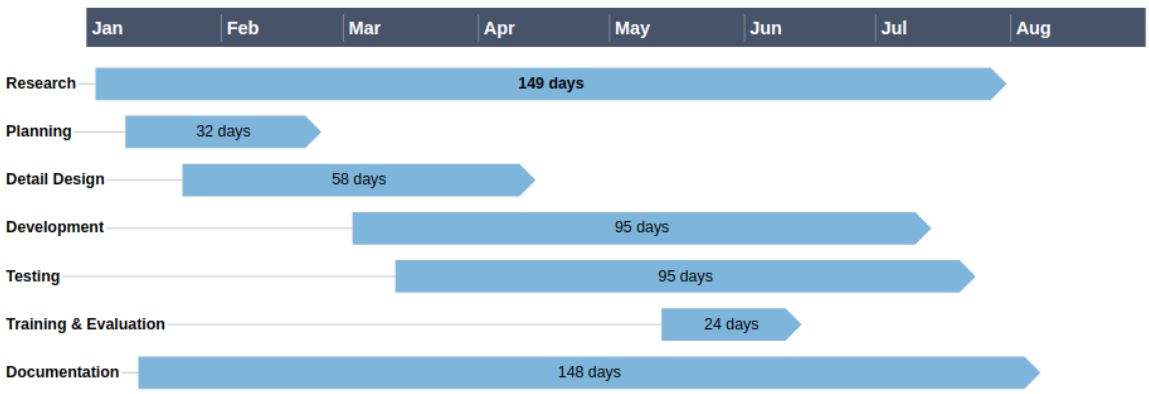

Figure 5.4: Gantt Chart

CHAPTER 6
CONCLUSION

6.1 Conclusion

The main objective of this project was to develop a module to analyze and identify the questions having same semantic meaning so as to prevent the duplication for Q&A forums like Quora and which would be able to assist mitigatation of redundancy in answers to support high-quality knowledge base. Using the machine learning techniques on the dataset, the main goal was fulfilled. The project involved understanding the challenges in the collection of relevant features from sentences and designing a system to decide upon the question of duplication of question pairs, and the semantic relatedness of sentences from words.

In conclusion, our projects is capable of adding values to a Q&A forums and its users as mentioned in the applications.

6.2 Future Enhancement

Regardless of the fulfilment of the project objectives, there are few issues in the process of analysis and decisions, where room for improvements persists. The major limitations of the existing system can be summarized in the following points:

- Our system doesn't perform POS tagging explicitly. Explicit relationship with adjacent and related words in a phrase, sentence, or paragraph is ignored.
- Not designed for NER, the subtask of information extraction that seeks to locate and classify named entity mentions in unstructured text into pre-defined categories such as the person names, organizations, locations, medical codes, time expressions, quantities, monetary values, percentages, etc.
- Quora's dataset cannot cover the entirety of questions necessary for a machine model to interpret and so, there are cases where the system fails to do so.

We, in the field of statistics and machine learning tried to develop theoretical models

aiming to predict the behaviour of a certain process, the idea of the quote, “All models are wrong, but some are useful” is that every single model will be wrong, holds for every machine model, meaning that it will never represent the exact real behaviour. Having said that, even if a model cannot describe exactly the reality, it could be very helpful if it is close enough, where the areas of further improvements and enhancements are presented as follows:

- **Data from multiple Q&A forums**

  Our only source of question pairs is Quora. Collecting data from other sources helps to cover more of the entirety of questions, as one source cannot cover all the questions. This helps to get the complete picture of the situation.
- **Perform Part of Speech Tagging**

  Part-of-speech tagging is harder yet more accurate than just having a list of words and their parts of speech, because some words can represent more than one part of speech at different times, and because some parts of speech are complex or unspoken. This is not rare, in natural languages, a large percentage of word-forms are ambiguous.
- **Perform Named Entity Recognition**

  Classification of named entities that are present in a text into pre-defined categories like individuals, companies, places, organization, cities, dates, product terminologies etc. is not implemented in the system, but if performed, it could add a wealth of semantic knowledge to the content and helps to promptly understand the subject of any given text.

REFERENCES

[1] G. Marvin. (2018) Quora introduces broad targeting, says audience hits 300 million monthly users. [Online]. Available: https://marketingland.com/quora-introduces-broad-targeting-says-audience-hits-300-million-monthly-users-248261

[2] L. Jiang, S. Chang, and N. Dandekar. (2017) Semantic question matching with deep learning. [Online]. Available: https://engineering.quora.com/Semantic-Question-Matching-with-Deep-Learning

[3] K. Dey, R. Shrivastava, and S. Kaushik, "A paraphrase and semantic similarity detection system for user generated short-text content on microblogs," 2016.

[4] L. Guo, C. Li, and H. Tian, "Duplicate quora questions detection," 2017.

[5] T. G. Addair, "Duplicate question pair detection with deep learning," 2017.

[6] A. Thakur, "Is that a duplicate quora question?" 2017. [Online]. Available: https://www.linkedin.com/pulse/duplicate-quora-question-abhishek-thakur

[7] K. Csernai. (2017) First quora dataset release: Question pairs. [Online]. Available: https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs

[8] J. Ramos, "Using tf-idf to determine word relevance in document queries," 01 2003.

[9] X. Rong, "word2vec parameter learning explained," *CoRR*, vol. abs/1411.2738, 2014. [Online]. Available: http://arxiv.org/abs/1411.2738