UNIVERSITY OF TARTU
Institute of Computer Science
Software Engineering Curriculum

Navedanjum Ansari

Identifying Semantically Duplicate Questions Using Data Science Approach: A Quora Case Study

Master's Thesis (30 ECTS)

Supervisors: Rajesh Sharma

Tartu 2019

Identifying Semantically Duplicate Questions Using Data Science Approach: A Quora Case Study

Abstract:

Two questions are semantically duplicate, given that precisely the same answer can satisfy both the questions. Identifying semantically identical questions on, Question and Answering(Q&A) social media platforms like Quora is exceptionally significant to ensure that the quality and the quantity of content are presented to users, based on the intent of the question and thus enriching overall user experience. Detecting duplicate questions is a challenging problem because natural language is very expressive, and a unique intent can be conveyed using different words, phrases, and sentence structuring. Machine learning and deep learning methods are known to have accomplished superior results over traditional natural language processing techniques in identifying similar texts.

In this thesis, taking Quora for our case study, we explored and applied different machine learning and deep learning techniques on the task of identifying duplicate questions on Quora's question pair dataset. By using feature engineering, feature importance techniques, and experimenting with seven selected machine learning classifiers, we demonstrated that our models outperformed a few of the previous studies on this task. Xgboost model, when fed with character level term frequency and inverse term frequency, achieved superior results to other machine learning models and also outperformed a few of the Deep learning baseline models.

We applied deep learning techniques to model four different deep neural networks of multiple layers consisting of Glove embeddings, Long Short Term Memory, Convolution, Max pooling, Dense, Batch Normalization, Activation functions, and model merge. Our deep learning models achieved better accuracy than machine learning models. Three out of four proposed architectures outperformed the accuracy from previous machine learning and deep learning research work, two out of four models outperformed accuracy from previous deep learning study on Quora's question pair dataset, and our best model achieved accuracy of 85.82% which is close to Quora state of the art accuracy.

Keywords: Quora, Duplicate question, Machine learning, Deep learning, model, neural network

CERCS: P170 - Computer science, numerical analysis, systems, control

Semantselt Kahekordsete Küsimuste Kindlakstegemine: Quora Juhtumi Uurimine

Lühikokkuvõte: Kaks küsimust on semantselt dubleeritud, arvestades, et täpselt sama vastus võib rahuldada mõlemaid küsimusi. Semantselt identsete küsimuste väljaselgitamine selliste sotsiaalmeediaplatvormide kohta nagu Quora on erakordselt oluline, et tagada kasutajatele esitatud sisu kvaliteet ja kogus, lähtudes küsimuse kavatsusest ja nii rikastades üldist kasutajakogemust. Dubleerivate küsimuste avastamine on väljakutseks, sest looduskeel on väga väljendusrikas ning ainulaadset kavatsust saab edastada erinevate sõnade, fraaside ja lausekujunduse abil. Masinõppe ja sügava õppimise meetodid on teadaolevalt saavutanud paremaid tulemusi võrreldes traditsiooniliste loodusliku keeletöötlemise tehnikatega sarnaste tekstide väljaselgitamisel.

Selles teoses, võttes Quora oma juhtumiuuringuks, uurisime ja kohaldasime erinevaid masinõppe- ja sügavõppetehnikaid ülesandel tuvastada Quora küsimuse paari andmesetikul Kahekordne küsimused. Kasutades omaduste inseneritehnikat, eristavaid tähtsaid tehnikaid ning katsetades seitsme valitud masinõppe klassifikaatoriga, näitasime, et meie mudelid edestasid paari varasemat selle ülesandega seotud uuringut. Xgboost mudelil, mida söödetakse tähetaseme termilise sagedusega ja pöördsagedusega, saavutati teiste masinõppemudelite suhtes paremad tulemused ning edestati ka paari Deep learningi algmudelit.

Meie kasutasime sügava õppimise tehnikat, et modelleerida neli erinevat sügavat neuralivõrgustikku, mis koosnevad Glove embeddingist, Long Short Term Memory, Convolution, Max pooling, Dense, Batch normaliseerimisest, Aktuaalsetest funktsioonidest ja mudeli ühendamisest. Meie süvaõppemudelid saavutasid parema täpsuse kui masinõppemudelid. Kolm neljast väljapakutud arhitektuurist edestasid täpsust varasemast masinõppe- ja süvaõppetööst, kaks neljast mudelist edestasid täpsust varasemast sügava õppimise uuringust Quora küsitluspaari andmestik ning meie parim mudel saavutas täpsuse 85.82% mis on kunstilise seisundi Quora lähedal Täpsus.

Võtmesõnad: Quora, Kahekordne küsimus, masinaõpe, õpe, Sügav õppimine, modell, neuralivõrgustik
CERCS: P170 - arvutiteadus, arvuline analüüs, süsteemid, kontroll

Table of Contents

1. Introduction

Social media platforms are a great success as can be witnessed by the number of the active user base. In the age of internet and social media, there has been a plethora of social media platforms, for example, we have Facebook, for user interaction, LinkedIn, for professional networking, WhatsApp for chat and video calling, Stack Overflow for technical queries, Instagram for photo sharing. Along the line, Quora is a Question & Answer platform and builds around a community of users to share knowledge and express their opinion and expertise on a variety of topics.

Question Answering sites like Yahoo and Google Answers existed over a decade however they fail to keep up the content value of their topics and answers due to a lot of junk information posted; thus their use base declined[1]. On the other hand, Quora is an emerging site for the quality content, launched in 2009 and as of 2019, it is estimated to have 300 million active users, and the company has a valuation of $2Bn[1]. Quora has 400,000 unique topics[2] and domain experts as its user so that the users get the first-hand information from the experts in the field.

With the growing repository of the knowledge base, there is a need for Quora to preserve the trust of the users, maintain the content quality, and also discard away the junk, duplicate and insincere information. Quora has successfully overcome this challenge by organizing the data effectively by using modern machine learning and deep learning technology to eliminate question duplication.

1.1 Scope and motivation

The most popular Q&A platforms have been Stack Overflow, Reddit, and Quora. In this thesis, we take Quora for our case study and investigate the reason behind the Quora's success in regards to organizing the content. In this thesis, we focus on aspects of how Quora maintain the knowledge repository by detecting the duplicate question such that all the answers for semantically similar questions to be organized so that users get quantity and quality of responses. We will analyze and run our experiments to predict the duplicate questions using Machine learning and Deep learning methods.

1.2 Research problem

As for any Q&A, it has become imperative to organize the content in a specific way to appeal users to be an active participant by posting questions and same time share their knowledge in respective domain of expertise. In keeping the users' interest, it is also essential that users do not post duplicate questions and thus multiple answers for a semantically similar question, this is avoided if semantically duplicate questions are merged then all the answers are made available under the same subject. Detecting semantically duplicate questions and finding the probability of matching also helps the Q&A platform to recommend questions to the user instead of posting a new one. Given our focus of study, we defined the following research questions:

RQ1: How does the structure and features of Quora help in presenting the most valuable information to its users from a massive pool of data contents?

[1] Vox - https://www.vox.com/recode/2019/5/16/18627157/quora-value-billion-question-answer
[2] Statistics 2019 - https://foundationinc.co/lab/quora-statistics/

RQ2: How can we detect duplicate questions on Quora using machine learning and deep learning methods?
RQ3: How can we achieve the best possible prediction results on detecting semantically similar questions?

The first research question has been studied in the past in terms of specific areas such as only the topic organization, user organization or general organization but we aim to put together the holistic view of the content organization both in terms of user base and the contents. Research question two and three have also been studied on the first dataset released by Quora[3] however we aim to achieve the higher accuracy in detecting semantically duplicate questions on same datasets, and we will employ both machine learning and deep learning methods to achieve better prediction results.

1.3 Summary of contribution and structure description

We have extracted different features from the existing question dataset and explored various machine learning algorithm. After employing feature engineering upon raw dataset, we experimented with different machine learning algorithms to draw our baseline. We also showed that not all features were useful in predicting duplicate question and after analyzing and dropping a few of the features, our result for ML models slightly improved but did not degrade at all. We also have the existing baseline from the works of literature, which we will surpass. We then tried many deep learning methods to finally experiment with our four best deep learning architectures, using the tensor flow and keras python library[4]. With our experiment results, we have shown that deep learning methods are suitable for solving the problem of detecting semantically similar questions. Our deep learning methods win over not only our baseline machine learning models but also performs better than baselines from previous research studies.

Moreover, our machine learning ensemble model TF-IDF achieved the accuracy of 82.33% and higher F1 score compared to literature[2]. We showed that our machine learning model achieved better F1 score and accuracy than those machine learning algorithm result in [2]. Also, our deep learning model achieved an accuracy of 85.82%. Three out of four presented deep learning models outperformed the results from the literature [2]–[5], and fourth architecture results achieved close to Quora's state of the art accuracy presented by Quora engineering team on their website[6].

The structure of the thesis is as follows:
Chapter 2 presents the related work done on analyzing Quora and detecting duplicate questions.
Chapter 3 explains the details about data, exploratory data analysis, and data cleaning.
Chapter 4 presents the description of machine learning and deep learning methods used to build the prediction models.
Chapter 5 describes the evaluation metrics used in machine learning and deep learning approach.
Chapter 6 discusses the feature engineering, experiment design, and approach employed.
Chapter 7 presents the models used in this research and evaluation of results.
Chapter 8 presents a summary of the result and discussion on future work.

[3] https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs
[4] Tensorflow keras - https://www.tensorflow.org/guide/keras

2. Literature Review

This chapter presents the review and description of works of literature. In the later section, we present a summary of the performance baseline selected from the previous study.

2.1 Different approaches

Structure and Effective Features of Quora

The previous studies that are most relevant to the research question on the structural organization of content are discussed thoroughly in "*Wisdom in social crowd: an analysis of Quora*" [1]. Quora is composed of three types of graphs. The user-topic graph, social graph, and associated questions to questions graph to recommend related questions. This paper has presented a detailed analysis of these three graphs to understand how they help users filter out a few exciting questions-answers from a large group of less appealing questions-answers.

The paper also tried to compare the effectiveness of attracting answers using social ties to that of attracting answers from the following topics. The result was that questions received more answers from users who follow the associated topic rather than those who follow the asker but still both social ties and topics play an important role in attracting answers. The other aspect examined was the answer quality of answers contributed by followers taking the vote count as a measure. The outcome was that for more than 50% of the questions asked by super users, answers from followers were of high quality; this goes hand in hand with the similar survey on Q & A behaviors on Facebook[7] which suggests that close friends have more incentive to provide right answers.

One of the features of Quora that is strongly affected by social connection is upvotes. The primary purpose of upvotes is to promote the quality of answers. The paper has tried first tried to analyze the impact of voting on the ranking of answers. The result shows that in 85% of the questions, the best-ranked answer got the highest number of votes; this shows that the number of votes is the main factor for ranking of answers; this raises a concern that potential bias in the voting process could result in an incorrect ranking of answers. Authors in Facebook[7] have analyzed votes on answers provided by super users. The outcome of was that in 40% of the questions, answers from super users received the highest votes while 60% of the case, their answers are in the top two most voted. The implication here is that regardless of the quality of their answer, superusers can often get more votes over other users [8].

Detection of duplicate questions

There have been quite a few works of research on the detection of duplicate data on the first release Quora dataset. The previous works related to our research questions, which discusses the best performing Machine learning and Deep learning approach are reviewed as follows.

The previous work to detect duplicate question pairs using Deep learning approach[5], shows that deep learning approach achieved superior performance than traditional NLP approach. They used deep learning methods like convolutional neural network(CNN), long term short term memory networks (LSTMs), and a hybrid model of CNN and LSTM layers. Their best model is LSTM network that achieved accuracy of 81.07% and F1 score of 75.7%. They used GloVe word vector of 200 dimensions trained using 27 billion Twitter[5] words in their experiments.

The deep learning approaches in [5] utilize a Siamese network structure[9], where each question goes through a separate branch of the network with identical parameters and weights. Questions are first changed into vectors of vocabulary indexes then converted into word embedding using pre-trained Glove [10]. The embedding is then passed through the encoding layer to change the word matrix into a feature vector, and then the feature vectors are connected using the method proposed by Bowman et al. [11]. In the end, the connected feature vectors are passed through a multi-layer perceptron that gives the final output. The authors have experimented with three different encoding methods. The first method was CNN based on what was proposed by Koon[12] and Bogdanova et al. [13]. This encoding method had an output, a feature vector of size 1x328. The second encoding method was bidirectional LSTM based on what was proposed by Wang et al. [14]. This encoding method produced (1x10N) features where N is the number of words in a sentence. The third encoding strategy was a hybrid of the above two methods, which applies the bidirectional LSTM and the CNN method consecutively resulting in a feature vector of 1x328. The final step was to combine the feature vectors from the two branches of the Siamese neural network to produce the final prediction. For this, they used a multilayer perceptron, which resulted in a 2% performance gain compared to a mere concatenation of the two feature vectors. The final output layer was composed of a single (1x2) vector where each element corresponds to the duplicate and non-duplicate class.

The method proposed in the Stanford report[15] makes use of Siamese GRU neural network to encode each sentence and apply different distance measurements to the sentence vector output of the neural network. Their approach involves a few necessary steps. The first step was data processing, which involves tokenizing the sentences in the entire dataset using the Stanford Tokenizer[6]. This step also involved changing each question to a fixed length for allowing batch computation using matrix operations. The second step involves sentence encoding, where they used both recurrent neural network(RNN) and gated recurrent unit (GRU). They initialized the word embedding to the 300-dimensional GloVe vectors[10].

The next step was determining the distance measure[16] that are used in combining the sentence vectors to determine if they are semantically equivalent. There were two approaches for this step, the first being calculating distances between the sentence vectors and running logistic regression to make the prediction. The paper has tested cosine distance, Euclidean distance, and weighted Manhattan distance. The problem here is that it is difficult to know the natural distance measure encoded by the neural network. To tackle this issue, they replaced the distance function with a neural network, leaving it up to this neural network to learn the correct distance function. They provided a row concatenated vector as input to the neural network and also experimented using one layer and two-layer in the neural network. The paper utilized data augmentation as an approach to reduce overfitting. They also did a hyperparameter search by tuning the size of the neural network hidden layer (to 250) and the standardized length of the input sentences (to 30 words) which led to better performance.

[5] Twitter Glove 200d, 27B token - https://nlp.stanford.edu/projects/glove/
[6] https://nlp.stanford.edu/software/tokenizer.shtml

In the literature *Determining Entailment of Questions in the Quora Dataset* [3], authors have used word ordering and word alignment using a long-short-term-memory(LSTM) recurrent neural network[17], and the decomposable attention model respectively and tried to combine them into the LSTM attention model to achieve their best accuracy of 81.4% . Their approach involved implementing various models proposed by various papers produced to determine sentence entailment on the SNLI dataset.[7] Some of these models are Bag of words model, RNN with GRU and LSTM cell, LSTM with attention, Decomposable attention model. Some of the challenges they faced in implementing these models were the issue with memory because of the hugeness of the dataset and also issues with overfitting which they tried to tackle by introducing drop out and regularization.

Doing a sentence analysis showed that different models have their pros and cons in a different type of sentence pairs. Sentences similar grammatically but with words out of vocabulary were better classified with word-by-word and two-way-word-by-word attention models. On the other hand, LSTM attention model performed well in classifying sentences with words tangentially related. However, in cases were words in the sentences have a different order; the decomposable attention model[18] achieves better performance. This paper tried to combine the GRU/LSTM model with the decomposable attention model to gain from the advantage of both and come up with better models with better accuracy like LSTM with Word by Word Attention, and LSTM with Two Way Word by Word Attention.

In the relevant literature, *"Detection of Duplicates in Quora and Twitter Corpus"*[2], the authors have experimented with six traditional machine learning classifiers. They used a simple approach to extract six simple features such as word counts, common words, and term frequencies(TF-IDF)[19] on question pairs to train their models. The best accuracy reported in this work is 72.2% and 71.9% obtained from binary classifiers random forest and KNN, respectively.

Finally, we reviewed the experiments by Quora's engineering team[6] on solving the problem of detecting semantically duplicate questions. In production, they use the traditional machine learning approach using random forest with tens of manually extracted features. Three architectures presented in their work use LSTM in combination with attention, angle, and distances. The point noted from this literature is that Quora uses the word embedding from its Quora Corpus and therefore state of the art has higher accuracy reported compared to all other selected baselines from the literature review that used GloVe pre-trained word to vectors from the glove project[8].

2.2 Application of the research

The content organization speaks about the victory; Quora has achieved in terms of popularity and quality contents. Therefore, the content organization and features implemented by Quora can serve as a benchmark for other social media platforms in general and precisely a standard method to organize and moderate contents for other Q&A platforms.

[7] SNLI - https://nlp.stanford.edu/projects/snli/
[8] https://nlp.stanford.edu/projects/glove

Identifying duplicate texts has an advantage in various domains, such as information retrieval for building efficient search engines and recommendation systems. Detecting semantically similar question is a hard problem since there are multiple ways to describe textually, the same meaning.

The model can be useful in cases where content-categorization is required or for sorting user-generated contents online. It can be helpful to build automatic chatbots that reply to user queries online and thus reduces the human effort by avoiding to cater to each individual's queries. User can search for their answers from the pool of available questions, or they can be offered the recommendations to look for question items which are similar to what they intend to ask. Such models can utilized in Online Chat, Information retrieval search engines, Q&A forums, call center support desks.

2.3 Summary

By summarizing the related work, it is noted that the various approach used by the authors of the reviewed paper can be utilized to ensemble a better model for detecting duplicate questions. The chain network of users represented helps any social network platform to organize their user base and content effectively. Previous studies [2]–[5] are most relevant to this research work in detecting duplicate questions as these pieces of literature have used traditional machine learning and deep learning algorithms on the duplicate question datasets. We use results from [2]–[5] as the baseline results for our experiments, and we aim to outperform their results using our proposed approach using traditional machine learning and deep neural networks. The Quora blog from Quora's engineering team[6] is very relevant to our research experiments as it is the Quora state of the art produced in the duplicate question pair dataset and influence the selection of our deep learning algorithms. The other reviewed works also influence the selection of features and models in our experiments.

Table 1. Performance Baseline from selected literature

Paper	Model	Technique	Accuracy	F1 score
Detection of Duplicates in Quora and Twitter Corpus[2]	Logistic regression	Machine Learning	0.671	0.66
	Decision Tree		0.693	0.69
	SVM		0.6	0.55
	KNN		0.719	0.72
	Naïve Bayes		0.637	0.5
	Random Forest		**0.722**	**0.73**
Determining Entailment of Questions in the Quora Dataset[3]	LSTM	Deep learning	0.784	0.8339
	LSTM with Attention		0.81	0.8516
	LSTM with Two Way Word by Word Attention		**0.814**	**0.8523**
	Decomposable Attention Model		0.798	0.8365
Quora Question Duplication[4]	Siamese with bag of words	Deep learning	77.3	73.2
	Siamese with LSTM		83.2	79.3
	Seq2Seq LSTM with Attention		80.8	76.4
	Ensemble		**83.8**	**79.5**
Duplicate Question Pair Detection with Deep Learning[5]	LSTM (twitter word embedding 200d)	Deep learning	**0.8107**	**0.757**
Quora State of the Art[6]	LSTM with concatenation	Deep learning	0.87	0.87
	LSTM with distance and angle		0.87	0.88
	Decomposable attention		0.86	0.87

We aim to produce better results from the baseline selected from the previous study, the results achieved from each of the studies on Quora duplicate question pair dataset is summarized as presented in above Table 1

3. Data

In this chapter, we briefly describe the data collection, exploratory data analysis, data visualization, and data cleaning process.

3.1 Data collection

The data for this research work is taken from the First Quora Dataset release hosted on Amazon S3[9]. There is a total of 404290 rows in the dataset, which indicates that there are total 404290 question pairs, and the overall file size is 55.4 MB.

GloVe pre-trained word vectors are used for word embeddings. GloVe pre-trained vectors are available at SNLI project site Glove[10] . The total file size is 1.53GB in ZIP format '.gz.' These are vectors of dimension 300 are used to convert word to vectors in our machine and deep learning models. In the feature engineering process to convert word to vector for distance calculation, we used Google news vectors[11] *GoogleNews-vectors-negative300.bin.gz*, of 3 million words and 300 dimensions.

3.2 Exploratory Data Analysis

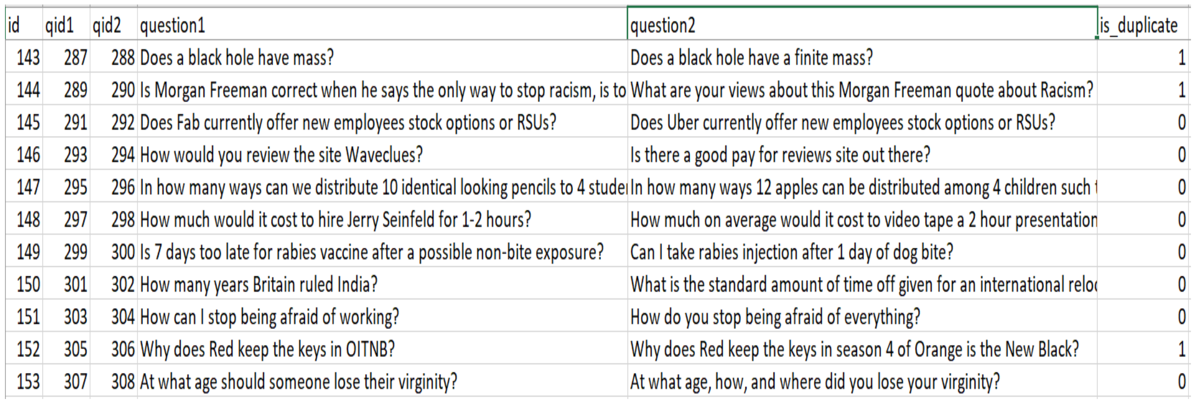
We performed the necessary statistics on the dataset, which helps us to give a more detailed understanding of the duplicate Quora question dataset. There is a total of six columns in the dataset. Each of the columns is meaningful and describe the characteristic of the row. The description of the columns is as described below in Table 2.

Table 2. Description of columns in dataset

Colum Name	Description
id	A unique identifier assigned to each row in the dataset. The first row has an id of 0, and the last row has id 404289
qid1	A unique identifier for the question in question1 column.
qid2	A unique identifier for the question in question2 column.
question1	question1 contains the actual question to be compared with question2
question2	question2 contains the actual question to be compared with question2
is_duplicate	is_duplicate is the result of a semantical comparison of question pair. 0 indicates false i.e. question pair is not duplicate 1 indicates true i.e. question pair is duplicate

[9] http://qim.fs.quoracdn.net/quora_duplicate_questions.tsv
[10] https://nlp.stanford.edu/projects/glove/
[11] https://s3.amazonaws.com/dl4j-distribution/GoogleNews-vectors-negative300.bin.gz

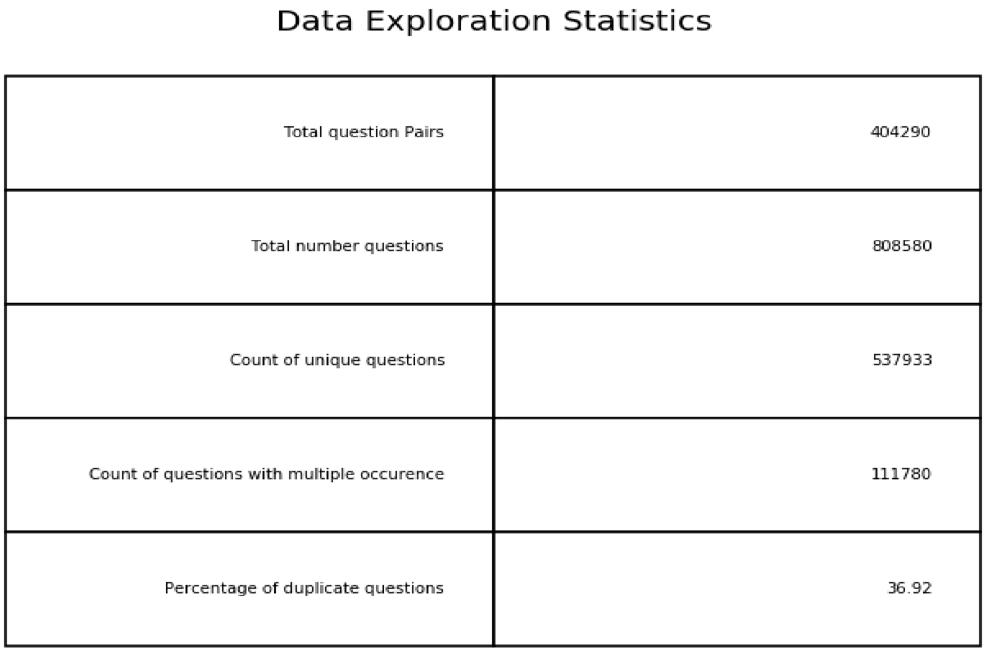
The snapshot below in Figure 1 shows the actual data in the raw tab-separated format.



id	qid1	qid2	question1	question2	is_duplicate
143	287	288	Does a black hole have mass?	Does a black hole have a finite mass?	1
144	289	290	Is Morgan Freeman correct when he says the only way to stop racism, is to	What are your views about this Morgan Freeman quote about Racism?	1
145	291	292	Does Fab currently offer new employees stock options or RSUs?	Does Uber currently offer new employees stock options or RSUs?	0
146	293	294	How would you review the site Waveclues?	Is there a good pay for reviews site out there?	0
147	295	296	In how many ways can we distribute 10 identical looking pencils to 4 studer	In how many ways 12 apples can be distributed among 4 children such t	0
148	297	298	How much would it cost to hire Jerry Seinfeld for 1-2 hours?	How much on average would it cost to video tape a 2 hour presentation	0
149	299	300	Is 7 days too late for rabies vaccine after a possible non-bite exposure?	Can I take rabies injection after 1 day of dog bite?	0
150	301	302	How many years Britain ruled India?	What is the standard amount of time off given for an international reloc	0
151	303	304	How can I stop being afraid of working?	How do you stop being afraid of everything?	0
152	305	306	Why does Red keep the keys in OITNB?	Why does Red keep the keys in season 4 of Orange is the New Black?	1
153	307	308	At what age should someone lose their virginity?	At what age, how, and where did you lose your virginity?	0

Figure 1. Snapshot of raw data file in tsv format

In the dataset using python, we ran the underlying statistics to look for the total number of unique questions, several questions that occur in the dataset more than once and the number of positive pair i.e. the question pair identified as semantically similar.



Data Exploration Statistics

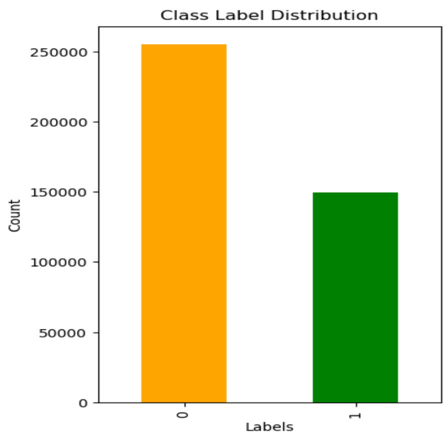
Total question Pairs	404290
Total number questions	808580
Count of unique questions	537933
Count of questions with multiple occurence	111780
Percentage of duplicate questions	36.92

Figure 2 Statistics on the dataset

As can be seen from Figure 2 above, we have total 537933 unique questions, and we could identify this from the unique qids from both qid1 and qid2. qids uniquely identify the questions; therefore, the repetition of qid suggested the questions occurred multiple times as part of another question column or question pair set. In our dataset, we have 36.92% of the duplicate question pair set, which is our positive sample identified as 1 in our class label column *is_duplicate*. Since our dataset consists of approximately 37% positive label and 63% negative label, we will not re-sample our dataset as this is an adequately balanced dataset.

3.3 Visual Dataset Representation

The plot below is Figure 3 is the visual representation of class label distribution in the dataset.



Positive Sample	149263
Negative Sample	255027
Total Question Pairs	404290

Figure 3. Class Label distribution

In the histogram plot in Figure 4 shows the distribution of question occurrence count.

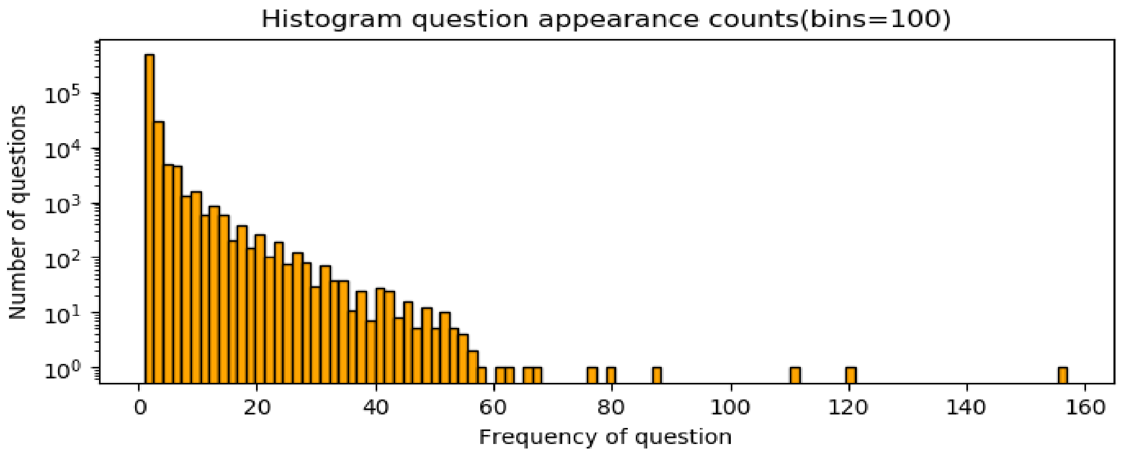


Figure 4 Distribution of question occurrence in dataset

In the histogram plot Figure 4, the x-axis represents the number of times question occurs, and the y-axis or height of the bar represents how many such questions with occurrence count exist in the dataset. As can be visualized from the graph the majority of questions occurs less than 60 times, and the first bar shows the unique occurrence and second bar the number of the appearance of question twice and so on.

3.4 Data Cleaning

We perform additional statistics using Python, that helps us to clean the data because this is a cumbersome task to perform manually on 404290 pair of questions. Table 3 presents additional stats computed from question pairs.

Table 3 Statistics on question1 and question2

Statistics	Average	Sum	Count
q1 length	59.53672	24070099	404290
q2 length	60.10838	24301217	404290
Max length q1			623(char count)
Max length q2			1169(char count)
q1 length - q2 length	-0.57166	-231118	404290
q1 length <=5	-	-	53
q2 length <=5	-	-	19

Mostly these questions short length questions are one word, one and two length questions are just the question marks and special characters, foreign characters. We discard as these data rows in the data cleaning process. From the table above, we observe that the q2 length on an average is greater, and therefore, we have an average negative difference. These basic statistics gives us an overall understanding of our dataset and help discard some of the data rows which are not useful. For example, we can eliminate all that dataset rows where question length is not significantly meaningful.

We dropped a total of 72 rows from our raw dataset based on the logic that both question1 length and question2 less than 6 or either one of the question length is less than 6.
Effective count in clean dataset = 404218.

We have utilized the cleaned set as an input to only our machine learning models. Elimination of 72 data rows will have a negligible impact on our prediction. However, we decided to discard the 72 rows and continue to work with 404218 data rows in our machine learning approach, and we continue with the usual data with 404290 rows in the original form for our deep learning approach.

4. Background

This chapter briefly explains the following:

- Different types of connection graphs Quora uses to deliver the contents, briefly introduced in Chapter 2, Section2.1
- Introduction to deep learning concepts and techniques.
- Presents the features extracted using feature engineering.
- Explains various machine and deep learning models used in the experiments of this thesis.

4.1 Quora Graphs

This section gives an overview of the different types of graphs, which are mainly discussed and analyzed in the reviewed work[1], [7], [8]. Quora is composed of three types of the user topic graph, the social graph, and the related question graph.

The user topic graph

The user topic graph enables users to get news and notification about the questions created under the topics they are following. The first analysis of this graph tries to compare and understand the relation between the number of followers and number of questions. The result shows that the top four followed topics are not the top four when it comes to the number of questions. So this means that a higher number of followers does not always produce more questions [1]. The other analysis tries to examine whether users interest in a topic will attract more activities in the questions under that topic and this was done by looking at the correlation between the number of views or answers per question, and the number of followers of each topic. The observation showed that questions under topics followed by many tend to have a higher number of average page view and answers. The above analysis shows that topics are an effective way of leading users to questions that they find interesting.

The social graph

A survey on Quora users[8] shows that most users follow people whom they find interesting and knowledgeable. To validate the claim, the paper[1] has analyzed the correlation between the number of followers that a user has to the quantity and quality (votes) answers that the user has posted. The result was especially users with less than 100 followers, which compose 91% of the total population showed a strong relationship between the number of followers and quality of answers.

The other analysis tries to understand the impact of social connection with question answering by asking the question, do super users draw more and better answers form their followers. The result showed that users do not get more answers to their questions just because they have many followers[8]. The next examination was to see the percentage of answers received from followers. It showed that even half of the questions asked by super users received no answer from their followers. The case might be because followers instead seek answers from followees. The other observation is that compared to ordinary users, super users attract more answers from followers, which shows that sociality has some level of influence on question answering.

The related question graph
In Quora, each question has a list of related questions ranked with the measure of similarity which forms the related questions graph. The paper has tried to analyze this graph to determine if this structure help user find top questions. One observation about the connectivity of the graph was that the question graph was dominated by one significant connected component that covers 98% of all questions. The 2% are mostly new questions whose related questions have not been computed yet. The research paper[1] tried to analyze was the stability of the question graph, to do this, a comparison was made between two snapshots of the question graph taken at different times with two-month difference. The result was that 60% of the of all the questions did not show any change while 30% had only one new entry in their related question list.

4.2 Feature Engineering

We dropped the first three columns id, qid1, and qid2 from the initial raw dataset and created additional useful features so that we have two columns question1, question2, and class label is_duplicate. Following are the new features designed from initial raw data, and we have a total of 31 columns in our featured dataset, which will serve as input to our machine learning models. Thus to summarize, we have thirty features and one class label column that is our binary class either 0 or 1.

Set 0 Base Feature

1. **Question 1 dataset:** This is the question 1 column in the dataset
2. **Question 2 dataset:** This is the quest 2 column in the dataset
3. **Is duplicate:** This is the class label which is a binary classification of whether given question pair is duplicate or not represented by 0 and 1 respectively.

Set 1 Basic Features

4. **Length of question1:** Length of the question1 feature is derived from the corresponding question1. Length includes all the characters, punctuation and white spaces.

5. **Length of question 2:** Length of the question2 feature is derived from the corresponding question2.Length includes all the characters, punctuation and white spaces.

6. **Difference in the length of questions:** Difference in the length feature is calculated as the difference between the length of corresponding question1 and question2.

7. **Number of characters in q1:** Number of characters in question1 feature is calculated as a distinct number of characters excluding white spaces in corresponding question 1.

8. **Number of characters in q2:** Number of characters in question2 feature is calculated as a distinct number of characters excluding white spaces in corresponding question 2.

9. **Number of words in q1:** Number of words in question 1 feature is calculated as the number of words in corresponding question 1 including repeated words.

10. **Number of words in q2:** Number of words in question 2 feature is calculated as the number of words in corresponding question 2 including repeated words.

11. **Number of common words in q1 and q2:** Number of common words in q1 and q2 feature is calculated as distinct common words in corresponding question 1 and question 2.

Set 2 Fuzzy Features

12. **Qratio:** Qratio feature is the quick ratio comparison of the two question strings and has value range from 0 to 100. More similar questions have a higher score.

13. **Wratio:** Wratio feature is the weighted ratio that uses different algorithms to calculate the matching score and returns the best ratio for two question strings. Score range from 0 to 100.

14. **Partial ratio:** Partial ratio feature calculates the best score for partial string matching against all substrings of the greater length and returns the best score. Score range from 0 to 100

15. **Token set ratio:** Token set ratio feature is calculated on the strings by segregating the strings into three parts. First part of common strings which are then arranged as sorted intersection, and other parts from each of the questions as sorted remainders. It then computes scores from compares sorted intersection with each of combination of sorted intersection and sorted remainders of that string. Token set ratio returns the highest score from the comparison on sorted intersection versus (sorted intersection + sorted remainder from question1) and sorted intersection versus (sorted intersection + sorted remainder from question2). Score range from 0 to 100.

16. **Token sort ratio:** Token sort feature tokenizes the strings and then sort the strings alphabetically and join back into strings. It then compares the transformed strings using ratio to return score. Score range from 0 to 100.

17. **Partial token set ratio:** Partial token set feature is similar to token set ratio except that after it tokenizes string it uses partial ratio in place of ratio to calculate the matching score. Score range from 0 to 100.

18. **Partial token sort ratio:** Partial token sort ratio is similar to token sort ratio except that it uses partial ratio in place of ratio, after sorting the token to compute matching score. Score range from 0 to 100.

Set 3 Distance Features

19. **Word mover's distance(wmd):** World mover's distance[20]feature calculates the distance between two documents, in our case, it gives the distance between two corresponding questions in our dataset. It uses word2vec embedding to find the distance between similar or semantically similar words. The stop words like '*the,*' '*to*' etc. are removed using nltk[12] library. We used pre-trained word2vec *GoogleNews-vectors-negative300.bin.gz* embedding and python's genism library to compute word mover's distance, a small value of wmd indicates that the two questions are related. Wmd uses Euclidean distance to calculate the distance.

20. **Normalized word mover's distance (norm wmd)**: Normalized word mover's is similar to word mover's distance just that word2vec vectors are normalized such that vectors have equal length because Euclidean distance could become large if the difference in the two vectors length differs. Normalizing helps in reducing risk of miscomputing.

21. **Cosine distance:** Cosine distance feature calculates the angle between the word vectors of two question sentences. The cosine distance value of 1 indicates that two sentences are in the same direction and hence related. Value of 0 means they are perpendicular with very less to no similarity and -1 means there is no similarity at all. Cosine distance is calculated using scipy's spatial[13] python library.

22. **Minkowski distance:** Minkowski distance feature is a generic distance metric that can be computed as the summation of differences of vector dimensions raise to the power p and whole raise to the inverse of power p. We have used p=3 to calculate the Minkowski distance.

$$d(x,y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/p} \tag{1}$$

23. **Cityblock distance:** Cityblock distance feature is a special case of Minkowski distance metric when we use the value of p=1 in the above equation of Minkowski distance. Cityblock is also known as the Manhattan distance.

24. **Euclidean distance:** Euclidean distance feature is also a special case of Minkowski distance metric when we use the value of p=2 in the equation of Minkowski distance.

25. **Jaccard distance:** Jaccard distance feature is computed as a ratio of intersection between two vectors sets to the union of two vector sets. The two vector sets are derived from the two question sentences in our dataset.

[12] NLTK – Natural Language Toolkit - https://www.nltk.org/
[13] https://docs.scipy.org/doc/scipy/reference/spatial.distance.html

26. **Canberra distance**: Canberra distance is computed as the sum of the absolute difference of two vector points divided by the absolute sum of individual vector points.

$$d(x,y)_{cab} = \sum_{i=1}^{n} \frac{|x_i - y_i|}{|x_i| + |y_i|} \tag{2}$$

27. **Braycurtis distance:** Braycurtis distance is also called as Sorenson distance. It is also a variant of Manhattan distance normalized by the sum of the vector points in two objects x and y.

$$d(x,y)_{sd} = \frac{\sum_{i=1}^{n} |x_i - y_i|}{\sum_{i=1}^{n} (x_i + y_i)} \tag{3}$$

Set 4 Vectors Features

28. **Skew question1 vector:** Skewness is the measure of distribution. Skewness indicates a deviation tendency from the mean in one of the direction. Skewness is computed over question 1 vector. A normal distribution has a skew value equal to 0.

29. **Skew question2 vector:** Skewness is computed over question 1 vector.

30. **Kurtosis question1 vector:** Kurtosis distance is the measure of dense distribution towards the tails of the distribution. A normal distribution has a value equal to 0. Kurtosis vector is computed over question 1 vector.

31. **Kurtosis question2 vector:** Kurtosis is computed over question 2 vectors.

4.3 Machine Learning Algorithms

We have used the following seven machine learning algorithms to draw our initial baseline on duplicate question pair dataset

K-Nearest neighbors: The k-nearest neighbors (KNN)[21] algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems. A supervised machine learning algorithm (as opposed to an unsupervised machine learning algorithm) is one that relies on labeled input data to learn a function that produces an appropriate output when given new unlabeled data.

Decision Tree: Decision tree classifiers are used successfully in many diverse areas. Their most important feature is the capability of capturing detailed decision-making knowledge from the supplied data. Decision Tree is the most powerful and accessible tool for classification and prediction. A Decision tree is a flowchart like a tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

Random forest: Decision trees are the building blocks of the random forest model.
Random forest[22], like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each unique tree in the random forest spits out a class prediction, and the class with the most votes becomes our model's prediction. The random forest is a classification algorithm consisting of many decisions trees. It uses bagging and feature randomness when building each individual tree to try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree.

Extra Trees: Extra tree[23] classifier is a type of ensemble learning technique which aggregates the results of multiple de-correlated decision trees collected in a "forest" to output its classification result. In concept, it is very similar to a Random Forest Classifier and only differs from it in the manner of construction of the decision trees in the forest. The main difference between random forests and extra trees (usually called extreme random forests) lies in the fact that, instead of computing the locally optimal feature/split combination (for the random forest), for each feature under consideration, a random value is selected for the split (for the extra trees).

Adaboost: AdaBoost[24] is a popular boosting technique which helps you combine multiple "weak classifiers" into a single "strong classifier". A weak classifier is simply a classifier that performs poorly but performs better than random guessing. AdaBoost can be applied to any classification algorithm, so it is really a technique that builds on top of other classifiers as opposed to being a classifier itself. AdaBoost is a type of "Ensemble Learning" where multiple learners are employed to build a stronger learning algorithm. AdaBoost works by choosing a base algorithm (e.g., decision trees) and iteratively improving it by accounting for the incorrectly classified examples in the training set. We assign equal weights to all the training examples and choose a base algorithm. At each step of the iteration, we apply the base algorithm to the training set and increase the weights of the incorrectly classified examples. We iterate n times, each time applying base learner on the training set with updated weights. The final model is the weighted sum of the n learners.

Gradient Boosting Machine: Gradient boosting[25] is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion as other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function. Gradient boosting involves three elements which include a loss function to be optimized, a weak learner to make predictions and additive model to add weak learners to minimize the loss function.

XGBoost: XGBoost[26] is an implementation of gradient boosted decision trees designed for speed and performance. XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. In prediction problems involving unstructured data (images, text, etc.), artificial neural networks tend to outperform all other algorithms or frameworks. However, when it comes to small-to-medium structured/tabular data, decision tree-based algorithms are considered best-in-class right now. XGBoost is short for extreme gradient boosting. It is a library designed and optimized for boosted tree algorithms. Its main goal is to push the extreme of the computation limits of machines to provide a scalable, portable and accurate for large scale tree boosting.

TF-IDF: TF-IDF[19] stands for term frequency-inverse document frequency, is a scoring measure widely used in information retrieval (IR) or summarization. TF-IDF is intended to reflect how relevant a term is in a given document. The intuition behind it is that if a word occurs multiple times in a document, we should boost its relevance as it should be more meaningful than other words that appear fewer times (TF). At the same time, if a word occurs many times in a document but also along with many other documents, maybe it is because this word is just a frequent word; not because it was relevant or meaningful (IDF). That is, the most relevant words are those that would help us, as humans, to better understand a whole document without reading it all. TF-IDF is computed at the word level or character level. In word level, as the name suggests frequency computed for words and for character level, the frequency is computed over characters.

We will implement this over the question dataset and apply TF-IDF and then apply the machine learning model to evaluate our results. In Python, we can use *TfidfVectorizer* function from the *sklearn.feature_extraction* Library[14]. To extract TF-IDF, we only need to select the analyzer type as either character for character level computation or word for world level TF-IDF computation.

4.4 Deep Learning Introduction

Deep learning is an AI-based machine learning technique that instructs computers to do tasks that falls naturally to mankind that is to learn by model. Deep learning is a crucial innovation behind automatic cars and autos empowering them to understand a stop signal or to differentiate between a person on foot and a street lamp post. It is the critical technology that enables voice control in gadgets like mobiles, laptops, Television, and headphones. In recent times, Deep learning is in the limelight because it is accomplishing results that were impractical earlier. With the help of deep learning, a computer-based model can self-learn classification tasks just like humans, directly from images, videos, texts, or voice. Deep learning models are capable of achieving superior accuracy that surpasses human-level results. These deep learning models are trained on an enormous amount of labeled data-set and neural architectures of multiple layers.

Majority of deep learning techniques utilize neural networks, and therefore, deep learning models are also known as deep neural networks. The term "deep' in deep-learning usually suggests the multiple hidden neural layers in the neural network. A conventional neural network is built with a small number of hidden layers usually 2-3, whereas deep neural networks can have a large number of hidden layers up to 150. The models that are built upon deep neural networks are trained on a massive amount of labeled data set, and the neural layers are capable of learning features directly from the data set without the need of manual feature extraction. Figure 5, given below, shows the visual representation of neural architecture.

[14] https://scikit-learn.org/stable/modules/feature_extraction.html

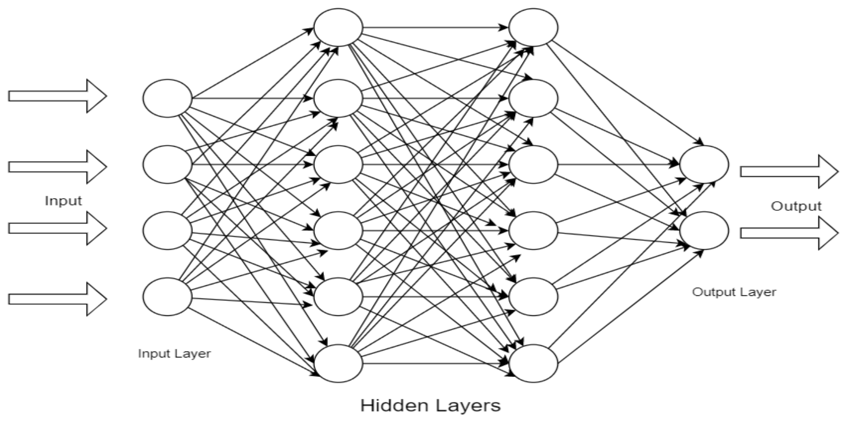


Figure 5. Neural Network Architecture

Methods to create and train Deep learning models are:

Train from scratch

The process of training a deep neural networks from scratch requires, collection of a large amount of labeled data set and development of a network architecture that will get familiar with the features and model from the input data. This method is useful for new application systems, or systems that will produce a large number of output labels. This technique is less common in deep learning because these deep neural networks typically take weeks or months to train the model due to a large amount of input data and a slow learning rate.

Transfer Learning

The transfer learning method is most commonly used in the majority of deep learning applications. In this process, we utilize a pre-trained model and then fine-tune as per our requirement and input data. We pick up some existing model such as GoogLeNet[27] and feed in new input data that contains previously unknown classification labels. We then make changes to the neural network to try out new tasks, for example, to detect numbers from 0 to 9 from image pixels in place of 1000's of distinct labels. Transfer learning does not require a tremendous amount of input data, and classification task can be achieved by training the tweaked model on small data set, this reduces the total computation time.

Feature Extraction

Feature extraction is a less commonly used method in deep learning; in this method, the neural network is treated as a feature extractor. Every layer is responsible for learning feature from the input data set and thus can at any given time extract the learned feature during training from these neural network layers. The extracted features can then be fed as input to the traditional machine learning classifiers such as random forest, support vector machines(SVM), etc.

4.5 Deep Learning Algorithms

1. LSTM[17]: Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It can process not only single data but also entire sequences of data. For example, LSTM applies to tasks such as unsegmented, connected handwriting recognition, or speech recognition. A standard LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals, and the three gates regulate the flow of information into and out of the cell. LSTM networks are well-suited to classifying, processing, and making predictions based on time series data since there can be lags of unknown duration between essential events in a time series. LSTMs were developed to deal with the vanishing gradient descent problem.

2. Word Embedding[20]: Word embeddings are a family of natural language processing techniques aiming at mapping semantic meaning into a geometric space. This is done by associating a numeric vector to every word in a dictionary, such that the distance between any two vectors would capture part of the semantic relationship between the two associated words. The geometric space formed by these vectors is called an embedding space.

3. Glove Embedding[10]: GloVe is used for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

4. Time Distributed(Dense)[15]: Time distributed dense layer is used on RNN, including LSTM, to keep one-to-one relations on input and output. Assume we have 60-time steps with 100 samples of data (60 x 100 in another word) and you want to use Recurrent Neural Network(RNN) with the output of 200. If we do not use time distributed dense layer, we will get 100 x 60 x 200 tensor. So we have the output flattened with each time step mixed. If we apply the time distributed densely, we are going to apply fully connected dense on each time step and get output separately by time steps.

5. Lambda: Lambda layer is a layer that wraps an arbitrary expression. For example, at a point, we want to calculate the square of a variable, but we cannot only put the expression into our model because it only accepts layer, so we need Lambda function to make our expression be a valid layer in keras. Lambda is similar to python's Lambda function where we can interact with keras layer using our own expression.

[15] Time Distributed Dense - https://keras.io/layers/wrappers/#timedistributed

6. Convolution 1D[16]: A CNN works well for identifying simple patterns within our data that will then be used to form more complex patterns within higher layers. A 1D CNN is handy when we expect to derive interesting features from shorter but mostly fixed-length segments of the overall data set and where the location of the feature within the segment is not of high relevance.

7. GlobalMaxPooling 1D[28]: This block performs precisely the same operation as the 1D Max pooling block except that the pool size is the size of the entire input of the block, i.e., it computes a single max value for all the incoming data. The 1D Global max pooling block takes a vector and computes the max value of all values for each of the input channels. The output is thus a tensor of size is 1 x 1 x (input channels). Use global max pooling blocks as an alternative to the Flattening block after the last pooling block of our convolutional neural network. Using 1D Global max pooling block can replace the fully connected blocks of our CNN

8. Merge[29]: Merge is used to join multiple neural networks together. A good example would be where we have two types of input, for example, tags and an image To combine these networks into one prediction and train them together, we merge these Dense layers before the final classification.

9. Dense[30]: A dense layer is just a regular layer of neurons in a neural network. Each neuron receives input from all the neurons in the previous layer, thus densely connected. The layer has a weight matrix W, a bias vector b, and the activations of previous layer a. The following is the docstring of class Dense from the keras documentation output = activation (dot (input, kernel) + bias) where activation is the element-wise activation function passed as the activation argument, the kernel is a weights matrix created by the layer, and bias is a bias vector created by the layer.

10. Batch Normalization[31]: Batch normalization is a technique for improving the performance and stability of neural networks, and also makes more sophisticated deep learning architectures work in practice. The idea is to normalize the inputs of each layer in such a way that they have a mean output activation of zero and standard deviation of one. This is comparable to how the inputs to networks are standardized. How does this help? We know that normalizing the inputs to a network helps it learn. However, a network is just a series of layers, where the output of one layer becomes the input to the next. That means we can think of any layer in a neural network as the first layer of a smaller subsequent network. Thought of as a series of neural networks feeding into each other, we normalizing the output

[16] Convolution 1D - https://keras.io/layers/convolutional/#conv1d

of one layer before applying the activation function, and then feed it into the following layer (sub-network).

11. Dropout[32]: Dropout is a regularization technique, which aims to reduce the complexity of the model to prevent overfitting. Using “dropout," we randomly deactivate specific units (neurons) in a layer with a certain probability p from a Bernoulli distribution. So, if we set half of the activations of a layer to zero, the neural network will not be able to rely on particular activations in a given feed-forward pass during training. As a consequence, the neural network will learn different, redundant representations; the network cannot rely on the particular neurons and the combination (or interaction) of these to be present. Another good side effect is that the training will be faster. Dropout is a technique used to tackle Overfitting. The Dropout method in “keras.layers” module takes in a float between 0 and 1, which is the fraction of the neurons to drop. Dropout consists of randomly setting a fraction rate of input units to 0 at each update during training time, which helps prevent overfitting.

12. PreLU[33]: Parametric Rectified Linear Unit(PreLU), Parametric ReLU is inspired by ReLU, which, as mentioned before, has a negligible impact on accuracy compared to ReLU. Based on the same ideas that of ReLU, PreLU has the same goals: increase the learning speed by not deactivating some neurons. The primary argument for Parametric ReLu’s over standard ReLu’s is that they do not saturate as we approach the ramp. In most other ways, they do not offer a distinct advantage. Think of it as an advantage in being able to tell the difference between a wrong answer and a horrible answer. The effect may not seem dramatic, but in some instances, it can be genuinely advantageous.

13. Activation[34]: Applies an activation function to the output of a layer such as tanh, sigmoid activation. It takes into consideration the effects of different parameter interaction and applies the transformation where it filters the value from which neuron to be passed to the next layer or the output.

5 Methodology

This chapter describes the process flow for our machine learning and deep learning experiments. In this chapter, a general approach to training our machine learning classifiers, the process flow for feature importance analysis, the process of TF-IDF with ML classifiers and four different deep learning architectures that we modeled for our experiments are presented.

5.1 Experimental and research design

Influenced by the literature and the previous study, we started our experiments with the binary classification of whether a given pair of question is a semantically duplicate question. We began with feature engineering to produce as many as 28 new features from the given question pair dataset and apply different machine learning algorithms. Figure 6 presented below shows the flow of traditional machine learning pipeline.

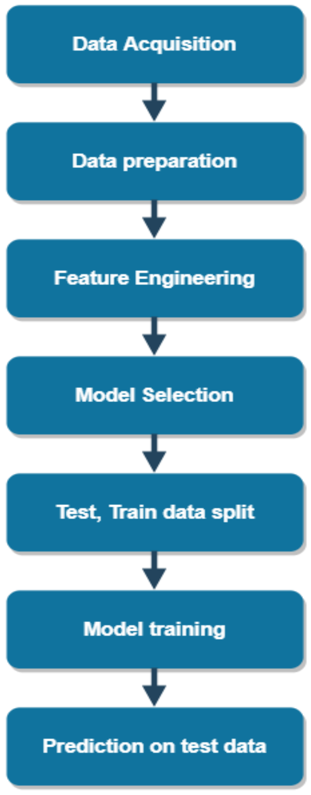


Figure 6. The flow of the experiment with traditional machine learning classifiers

Quora question pair dataset is collected and cleaned, as described in Section 3. Once the data is clean, we apply a set of operations to obtain new feature listed in Section 4.2. We obtain 28 new feature set from feature engineering technique, and then a dataset with a total of 30 columns are split into 80:20 training and test set. Python libraries *scipy spatial distance*, *genism*, *nltk*, *fuzzywuzzy*, *numpy* are used to extract features. Word2vec features are obtained using *genism* and

Google pre-trained vectors[17] of 3 million words and 300 dimensions. Machine learning classifiers are trained on the training set, and prediction results are evaluated on the test set.

5.2 Feature Importance

We analyzed and studied the features extracted using feature engineering to validate the positive contributions from each of the features, and then we retrain our models by dropping the least important features. The flow of feature importance is as presented below in Figure 7

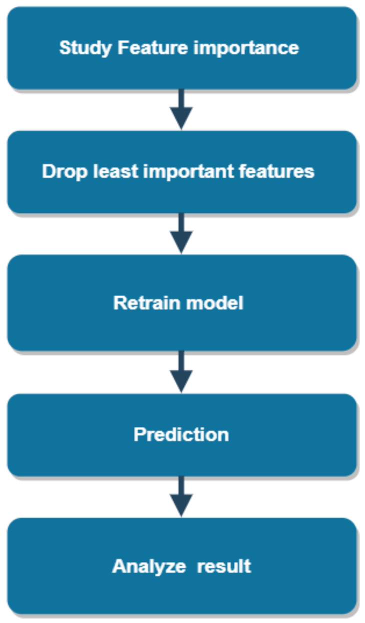


Figure 7. Flowchart of experiment with feature importance

We start by computing the feature importance value of all the features in the dataset. We have a total of 28 new features extracted in the experiment stage of Section 6.1. We analyze and select the top twenty features that are helpful to our machine learning classifiers, and then dropped eight features and re-train our model with only 20 new features and then evaluate new results from classifiers on the validation dataset.

5.3 Machine Learning Pipeline with TF-IDF

TF-IDF word level

The flow of term frequency and inverse term frequency(TF-IDF) word-level model with machine learning classifiers is presented in Figure 8. TF-IDF word level as the name suggests computes TF-IDF at word level in the document, in our case, it is a question sentence.

[17] https://s3.amazonaws.com/dl4j-distribution/GoogleNews-vectors-negative300.bin.gz

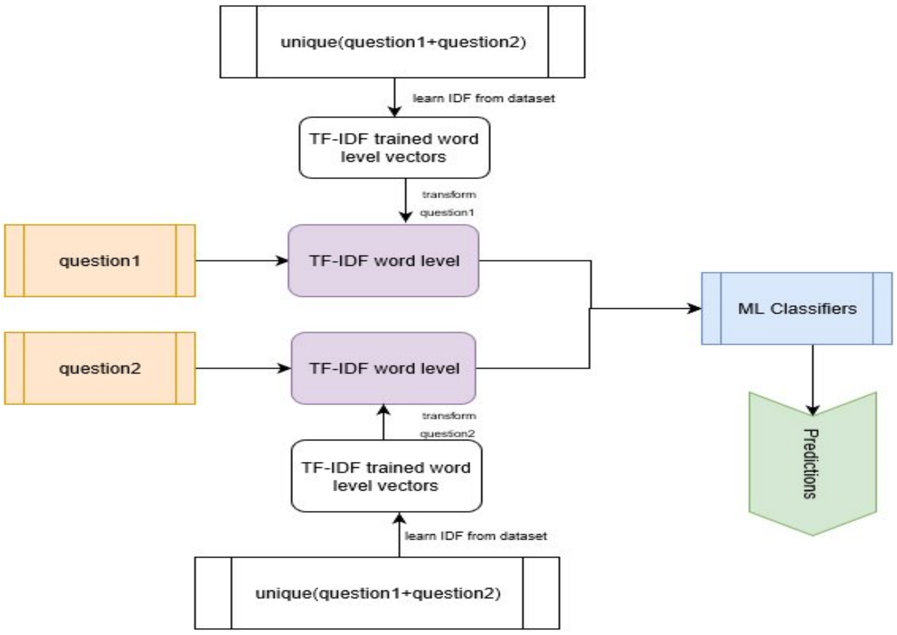


Figure 8. The flow of TFIDF word-level features fed to machine learning classifier

TF-IDF word-level feature for question1 and question2 is extracted using the python library from sklearn TFIDFVectorizer[18]. The model learns the inverse frequency of words from the set of combined unique question1 and question2 word set. The corresponding TF-IDF feature obtained for each of the questions in the pair is then passed as input to the different machine learning classifiers. The classifiers are then trained on the training dataset, which is 80% of total dataset and tested on 20% of the validation set.

TF-IDF character level

The flow of term frequency and inverse term frequency(TF-IDF) character level model with machine learning classifiers are presented in Figure 9. TF-IDF word level as the name suggests computes TF-IDF at character level in the document, in our case, it is a question.

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

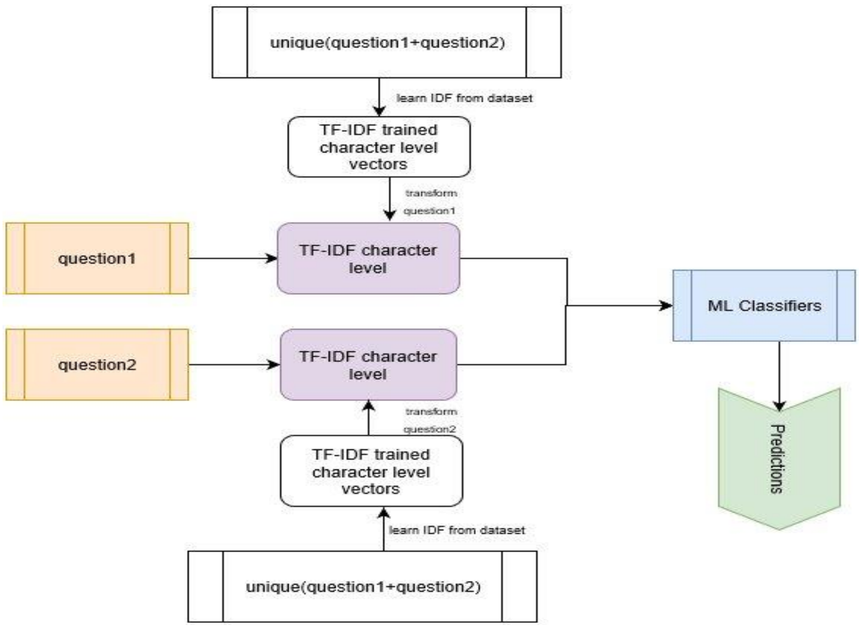


Figure 9. The flow of TFIDF character level features fed to machine learning classifier

TF-IDF character level feature for question1 and question2 is extracted using the python library from sklearn TFIDFVectorizer[19]. The model learns the inverse frequency of characters from the set of combined unique question1 and question2 character set. The corresponding TF-IDF, character level feature, obtained for each of the questions in the pair is then passed as input to the different machine learning classifiers. The classifiers are then trained on the training dataset, which is 80% of total dataset and tested on 20% of the validation set.

5.4 Deep Learning Design and Set-up

Architecture-1: In this simple neural network architecture, we use a pair of questions as the two inputs. The architecture consists of the Embedding layer, LSTM layer applied separately on each of the question inputs, and then the model is merged using the Merge layer from keras library[20]. The output from the merged model layer is then passed through the series of Batch Normalization, Dense, Parametric rectified linear unit, Dropout and Sigmoid Activation function is applied at the final output layer. The Visualization and sequential ordering of the different layers in the simple neural network can be visualized, as presented in Figure 10. The use of each of the layers applied to train our simple neural network is presented in the Chapter 4, section 4.5. Embedding layers is the first hidden layer of a network that uses word embedding to represent a word as a dense vector, and we specify three arguments to the Embedding function, the input dimension, output dimension, and the input length. We use the input length, i.e. number of words as 40 and output dimension as 300. Input dimension is computed as the index of words + 1 in the sequence.

[19] https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
[20] https://keras.io/

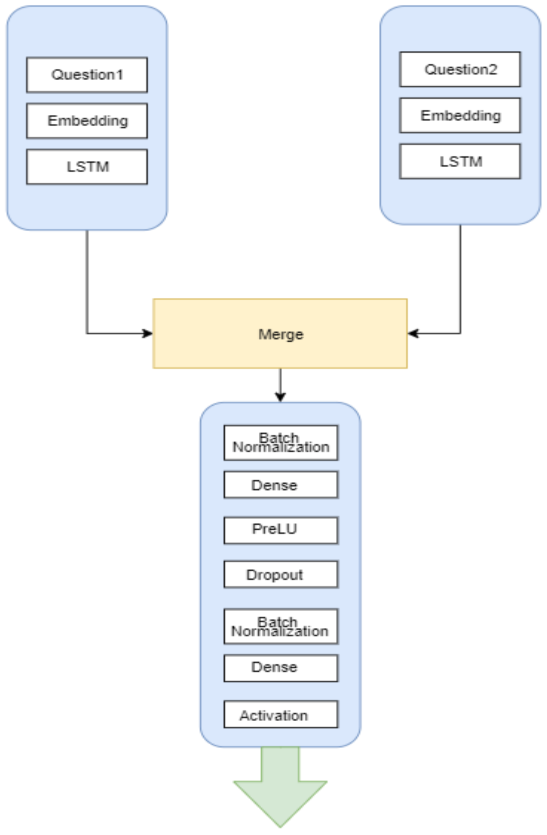


Figure 10. Architecture-1 Simple Neural network architecture with two inputs

In this model, we are not using any special pre-trained vectors like GloVe. The output of Embedding layer is fed to the LSTM layer. We used the dropout weight of 0.2 within LSTM to avoid overfitting. Each of the models merged as passed through a sequence of layers, as shown in Figure 10. The output from the intermediate Dense layer is 300, and the final Dense layer always has output dimension one, which then fed to sigmoid activation to give us the classification result.

Architecture-2: Neural network architecture-2 is modeled slightly different before applying to merge of different models otherwise after merge it very similar and trained on exactly same hyper-parameters as simple neural network presented as in Figure 10. In Architecture-2, we increase the number of independent models before merge to four, which are then merged and trained to produce the classification result. Architecture-2 with four inputs, two different networks are used for each of the questions; the architecture can be visualized as present in Figure 11 below.

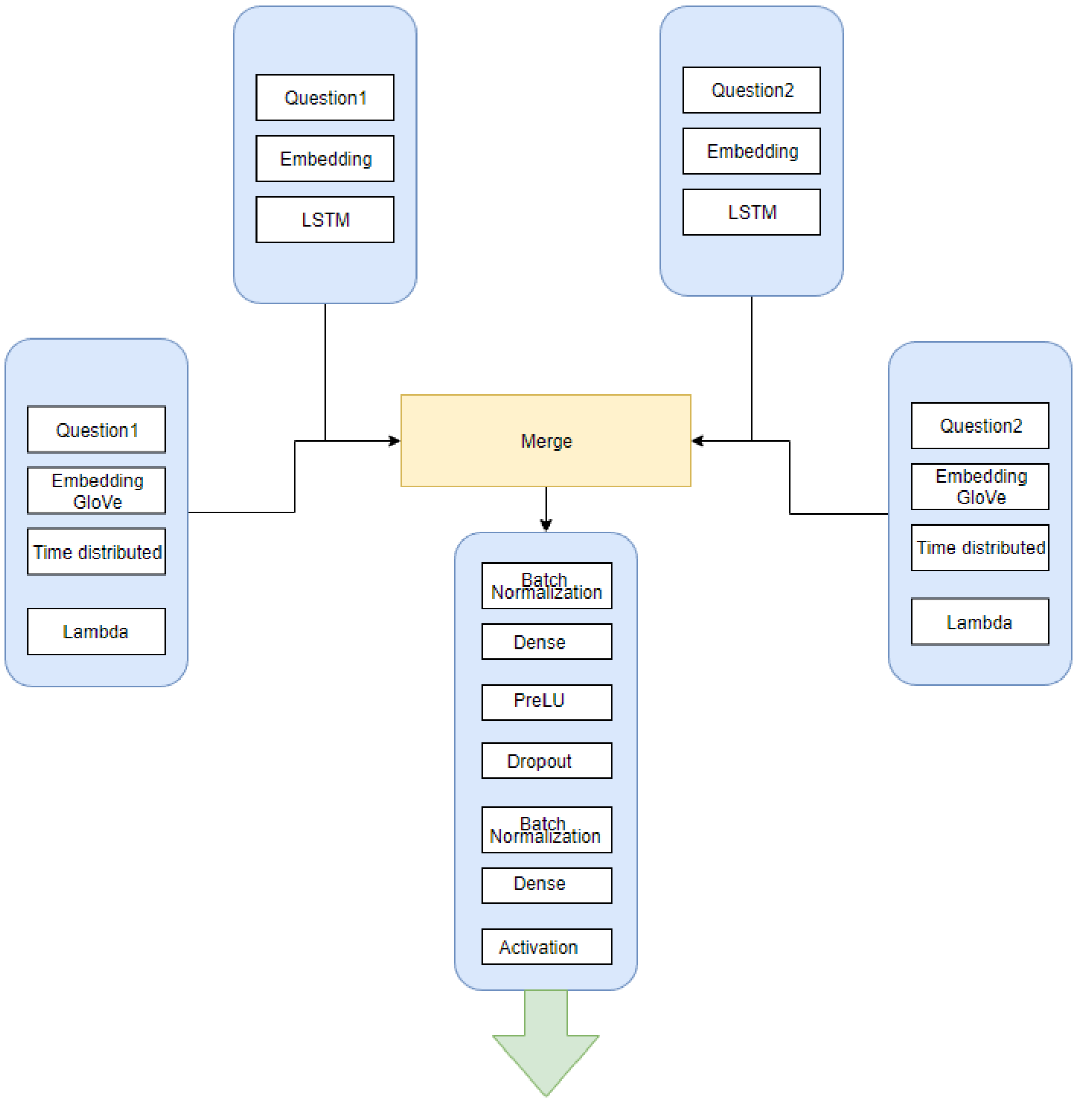


Figure 11. Architecture-2 Deep neural network architecture with four inputs

Additional models before the merge, consist of Embedding layer using GloVe[21] pre-trained vector of 300 dimensions with 840B tokens. Embeddings are then fed to Time distributed dense layer to maintain one to one relationship over time-distribution. Lambda sum is applied along the axis to produce the output of 300 dimensions. Thus all the four independent models producing the output of 300d are then merged and passed through hidden layers of Batch Normalization, Dense, PreLu, Dropout, Batch Normalization, Dense and Sigmoid Activation to produce the classification result.

[21] https://nlp.stanford.edu/projects/glove/

Architecture-3: Architecture-3 uses four sub-model or independent model from Architecture-2 with all the hyper-parameters tuned with the exact same value; the model differs after the merge of the four independent models. The modeled neural network architecture-3 can be visualized, as presented in Figure 12 below. In this deep neural network, we used additional hidden layers of Batch Normalization, Dense, and Dropout. Final Dense layer has output dimension 1, and then it passed through Activation to predict the classification result.

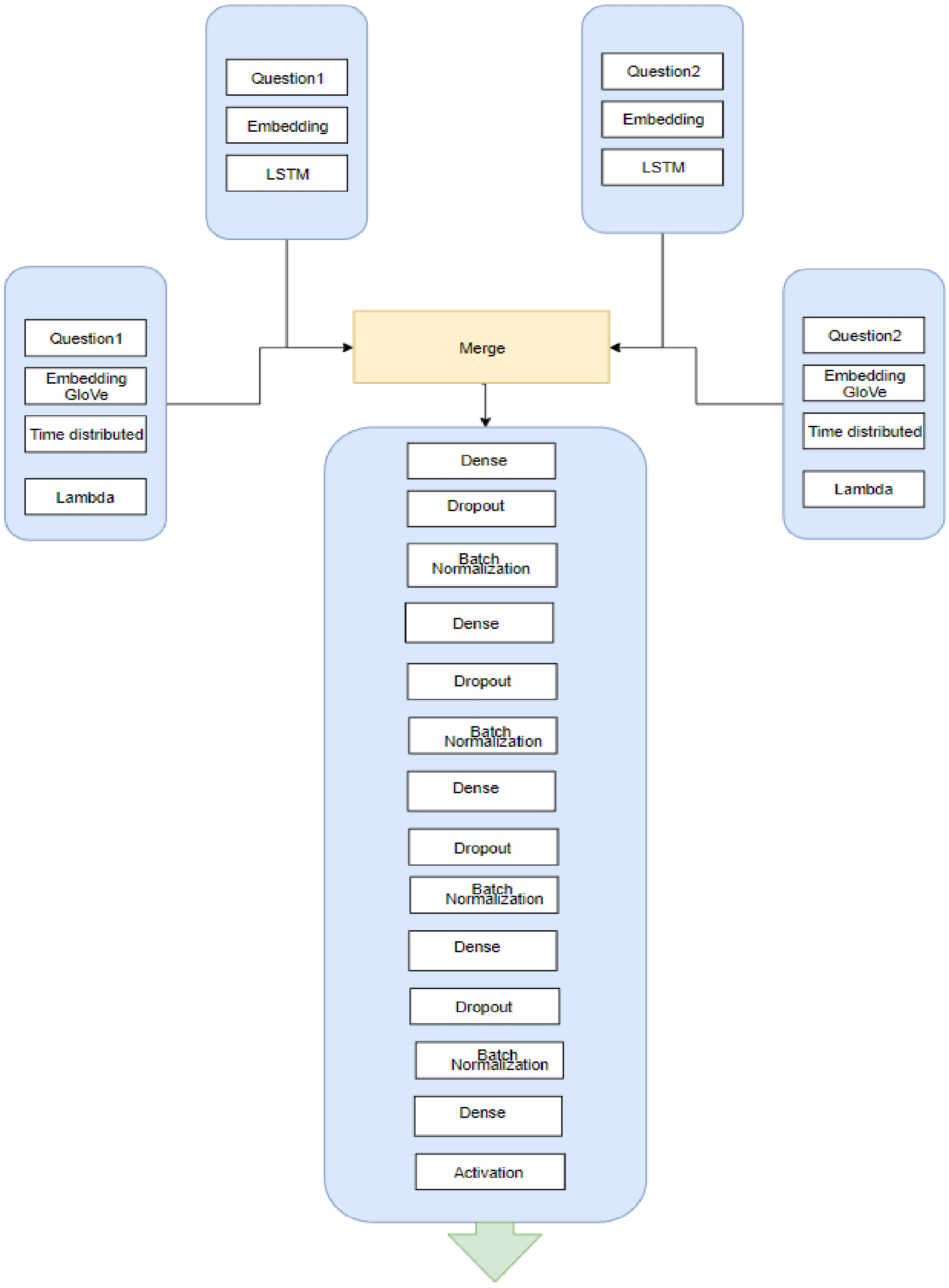


Figure 12. Architecture-3 Deep neural network with four inputs and dense hidden layers

Architecture 4: Deep neural network architecture-4 can be visualized, as presented in Figure 13 below:

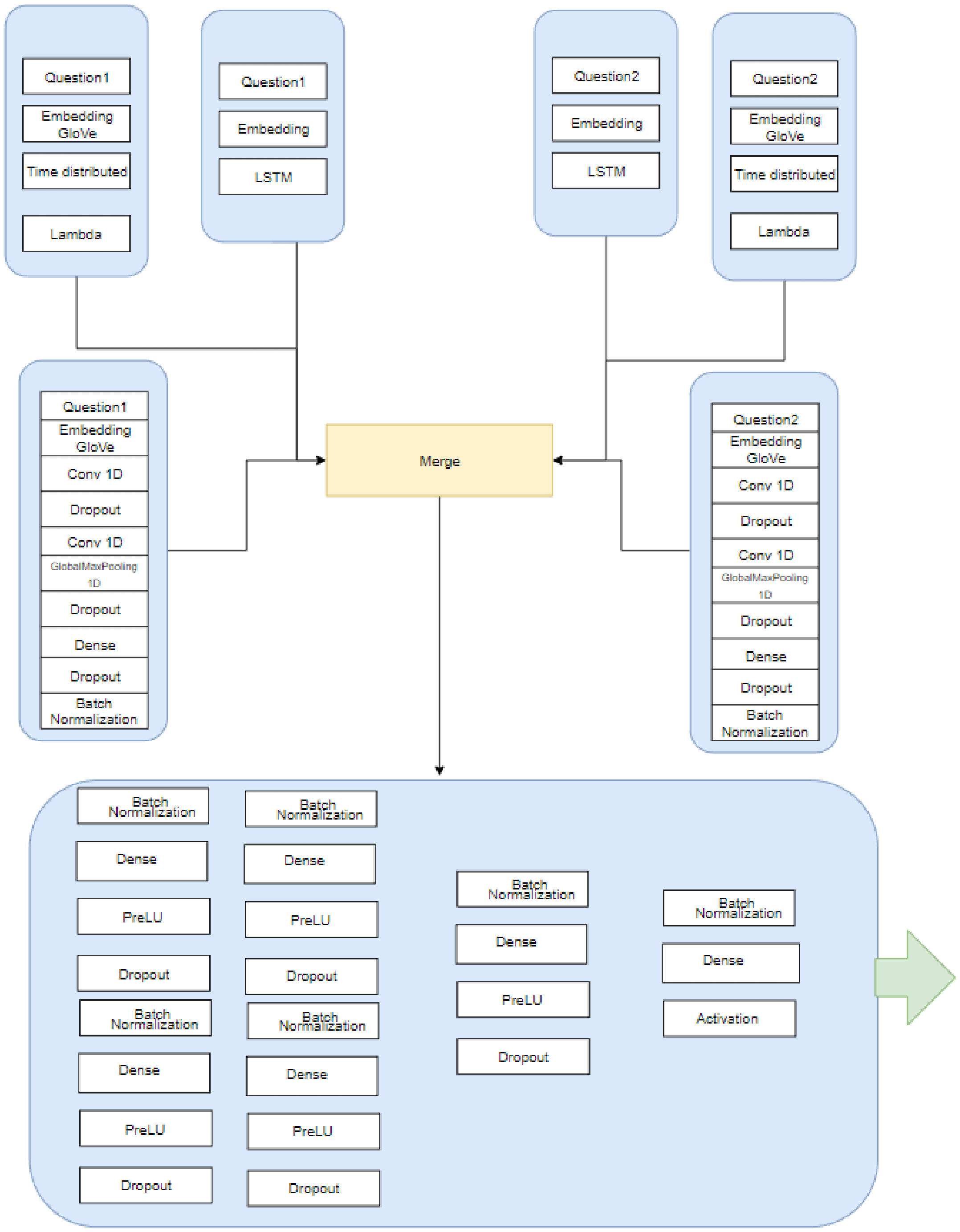


Figure 13. Architecture-4 Deep neural network with six inputs and dense hidden layers

The deep neural network architecture-4 is modeled in such a way that it takes the six input which are then passed through six independent models and then merged into a single model consisting of twenty-three layers.

Four out of six independent or sub-models are similar to that of the four sub-models before the merge as presented in Figure 12. The two new sub-models that we added consist of GloVe based Embedding layer, Convolution Neural Network layer applied multiple times before and after Dropout layer. The output from the Convolution 1D layer is maxed out using Global Max Pooling 1D[22] layer. Global Max Pooling output is then passed through hidden layers of Batch Normalization, Dense and Dropout. The Dropout layer has shown to perform well within our experiments with a weight of 0.2; therefore, throughout our neural network modeling; dropout weigh used is 0.2. All six layers produce the output of dimension 300 which is then merged as a single model and passed through another twenty-six layer consisting of repeated units of Dense, Dropout and Batch Normalization and finally a Dense layer with the output of dimension size one which is fed to sigmoid Activation to predict the classification result. See section 4.5 for description and functionality of each of these deep learning algorithms used in the layers. We have used TensorFlow[23] keras python library to model each of the neural network architecture presented in this section. All models are trained on the batch size of 300 and number of epoch iterations as 150.

[22] https://keras.io/layers/pooling/
[23] https://www.tensorflow.org/tutorials/keras

6 Description of models and results evaluation

This chapter presents the result obtained from our experiments by using the approach described in Chapter 5. Metrics used for model evaluation and comparative analysis of the result is also discussed.

6.1 Content Organization:

Investigation of content organization has been a peripheral part of this research work which on studying the relevant literature [1], [7], [8] shows that having connections within the social network as Quora does in the form of social, user topic and related question graph helps the social media to attract users by notifying through features like upvotes or activity of the followers and followees. The related question graph helps users to formulate their question is a better way or chose from one recommended question. Thus Quora has done an excellent job in building a social network and creating a link between users as well as links between data in the form of topic, question, and relevant answer. Quora graphs are explained in Section 4.1.

6.2 Evaluation Metrics

In this section, we present the evaluation metrics used for comparison of results. The selection of metrics is the most crucial step in the evaluation of our models as it influences how we measure the performance of our model against each other and the baselines selected. The metrics used in this research work are presented below.

Accuracy: Accuracy is the ratio of the total number of correct predictions made by the models to the total number of predictions requested to the model. In simpler terms, it is the total number of correct predictions, in case of binary classification such as our case 0 predicted as 0 and one predicted as 1. It is often expressed in terms of percentage.

F1-Score: F1-score or F1-measure is harmonic mean of precision and recall. To understand F1-Score, we need to understand Precision, also known as Specificity and Recall, also known as Sensitivity.

$$F1 = \frac{Precision * Recall}{Precision + Recall} \tag{4}$$

Precision: Precision or Specificity is the ratio of predicted positive samples that are actually positive to the total number of positive predictions made by the models. For example, in the test sample of 100, let us say our model predicts 50 samples to be positive but in actual only 40 of them are genuinely positive which means the model has predicted 40 samples correctly and ten samples incorrectly. Thus here the precision will be 40/50, i.e., 0.8 or 80%.

Recall: Recall or Sensitivity is the ratio of predicted positive samples that are actually positive to the total number of actual positive predictions. For example, we have 50 positive predictions as in the previous case, but only 40 of them are actually positive, but in the total sample of 100 we have actually 60 positives, i.e., out of 60, only 40 predicts are correctly done. Thus our recall is 40/60, i.e., 0.6667 or 66.7%.

Precision and Recall appears to be confusing at first, but the easiest way to get hold of the concept is to understand the distinct difference that is Precision is the proportion of positive predictions done correctly from the predicted positives and Recall is the proportion of positive samples identified from the actual positive labels.

Log loss: Log loss is also known as cross-entropy, and when the classification type is of binary as in our research, then it is known as binary cross-entropy. Log Loss value lies in the range of {0,1} where ideal models will have log loss of 0, and the worst model will have log loss of 1. Log loss indicates how badly our model predicted the probability of our classification. For example, let us say if a positive sample 1 is predicted to have a probability of 0.2, then an error in prediction is high and log loss increases as the predicted probability value has a huge difference. Similarly, if negative sample 0 is predicted to have a probability of 0.2, then the error in prediction is less, and log loss decreases as the predicted probability value is close to the actual class label. Thus log loss as performance metrics suggests how good our model is at predicting probabilities. In our work, we used log loss as one of the metrics to evaluate our deep learning models.

6.3 Baseline Model Classifiers

The most basic approach is to clean the data set after exploratory data analysis and extract features from feature engineering to produces the input to the model with 30 features and 1 class label as described in section 4.1 and 4.2. These feature files serve as input to our models to detect whether a given pair of question sets are duplicate or not. We split the dataset into 20 percent as validation dataset, and we train our models on 80 percent of the dataset.

We trained our model and then evaluated the prediction on our test data set to achieve the baseline for our machine learning algorithms used in this research. Table 4 shows validation accuracy and F1 score of our baseline machine learning models. The bar plot of accuracy achieved by classifiers is shown in Figure 14.

Table 4. The baseline performance of traditional machine learning classifiers on the dataset with thirty features

Classifiers	Validation Accuracy	Validation F1-Score
K Nearest Neighbors	0.7275	0.7031
AdaBoost	0.7041	0.6936
XGBoost	0.7417	0.7326
Gradient Boost	0.7271	0.7176
Decision Tree	0.7054	0.6992
Random Forest	0.7099	0.7016
ExtraTrees	0.7039	0.6849

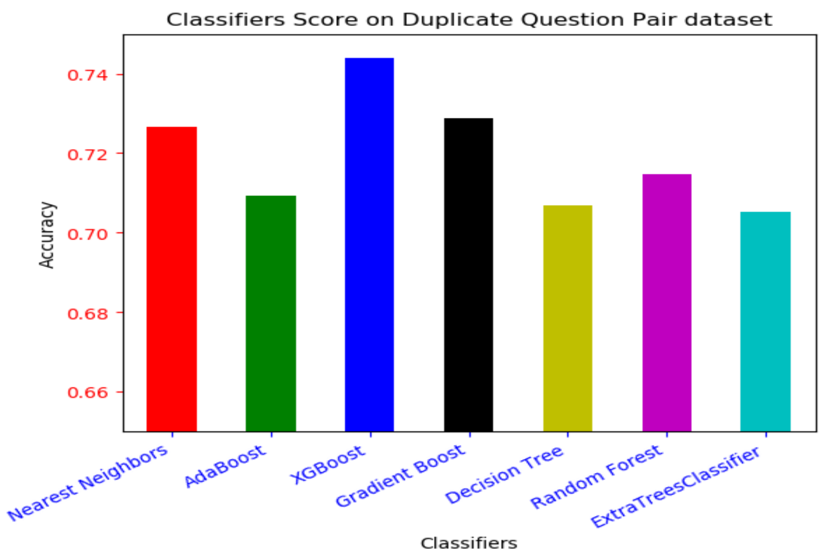


Figure 14. Accuracy of machine learning classifiers obtained from the featured dataset

As can be observed from Table 4 and Figure 14, clearly the Xgboost model outperforms all the other selected classifiers with the Accuracy of 0.7416 and F1 score of 0.7326.
The top three performing models in our baseline set are Xgboost, Gbm, and KNN. These three models have the highest accuracy and F1 scores.

6.4 Feature Importance Analysis

We plotted feature importance of our baseline classifiers to analyze the essential features for all our classifiers so that we can eliminate the non-contributing features and re-run our experiments to validate if dropping the features help our experiments. As long as we do not suffer any degradation in the performance of our model by dropping the features, we establish that we have dropped the unimportant features. We plotted feature importance graph for six classifiers, except for the KNN for which we studied feature importance from the result output matrix.

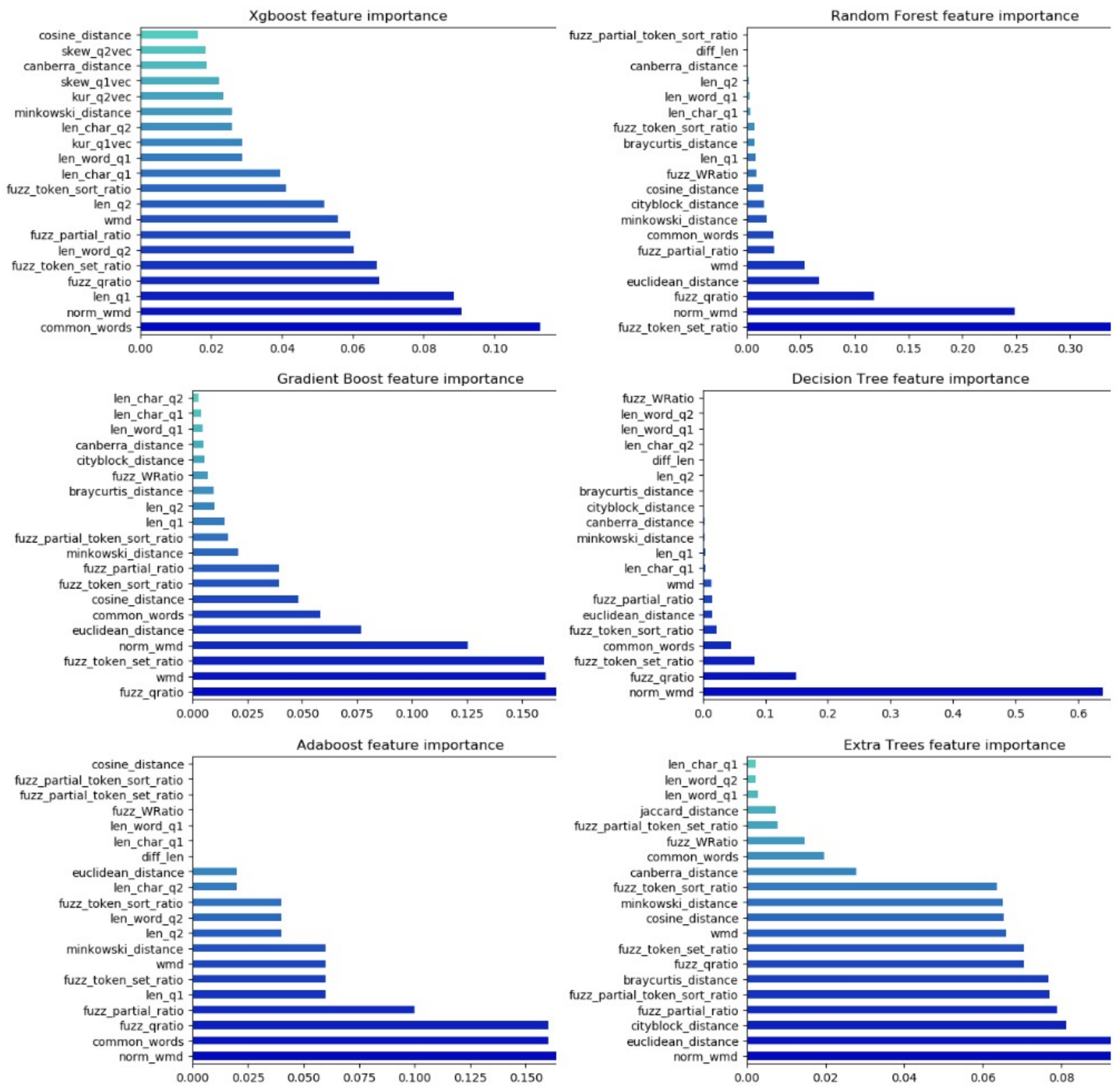


Figure 15. Feature Importance plot of machine learning classifiers

Based on our feature importance plot shown in Figure 15, we selected the top 20 features based on their importance concerning all the classifiers. The performance result achieved after feature importance analysis and feature drop is as presented below in Table 5.

Table 5. Performance of traditional machine learning classifiers after feature drop

Classifiers	Validation Accuracy	Validation F1-Score
K Nearest Neighbors	0.7311	0.7076
AdaBoost	0.7048	0.6938
XGBoost	0.7431	0.7349
Gradient Boost	0.7289	0.7196
Decision Tree	0.7054	0.6992
Random Forest	0.7085	0.7021
ExtraTrees	0.7069	0.6914

Xgboost, Gbm and KNN after feature drop still stood to be the top three performers in our baseline model set, and none of the classifiers suffers from any degradation. However, the gain achieved after feature drop is minimal. The plots below in Figure 16 and Figure 17 shows the comparative visualization of Accuracy and F1 score before and after the feature drop.

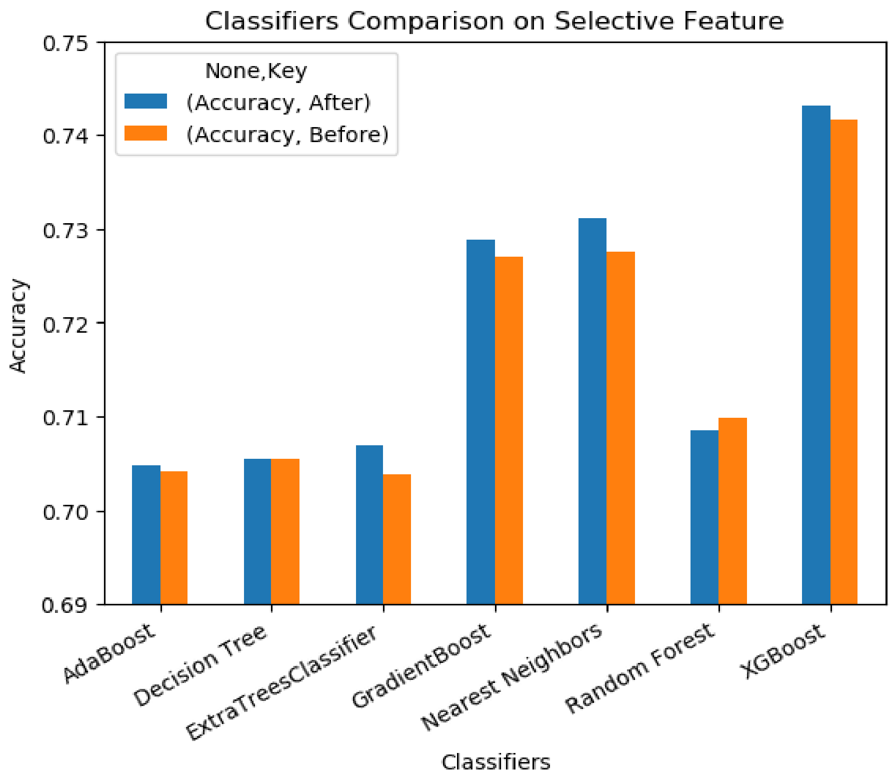


Figure 16. Accuracy comparison of ML classifiers Before versus After feature drop

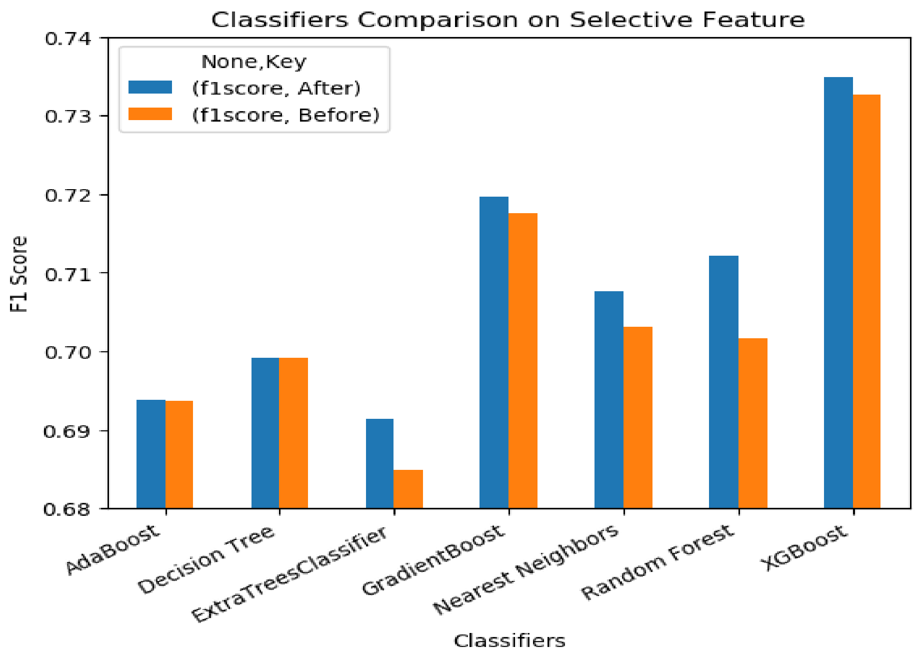


Figure 17. F1 score comparison of ML classifiers Before versus After feature drop

So far, Xgboost gives the highest accuracy score amongst all the classifiers. With Xgboost, we achieved the accuracy of 0.7431 and F1 score of 0.7439, followed by Gbm with an accuracy of 0.7289 and F1 score of 0.7196. KNN with an accuracy of 0.7311 and F1 score of 0.7076. The table below summarizes the actual Accuracy and F1 score achieved by our classifiers after selecting the top 20 features. The eight dropped features are *difference in the length, WRatio, jaccard distance, braycurtis distance, Euclidean distance, cityblock distance, partial token set ratio, partial token sort ratio*.

6.5 TF-IDF with ML Models

Xgboost algorithm achieved an F1 score of 80.44 % compared to F1 score 79% published in "*Quora Question Duplication" by Albert.T & Eric Xu."* The accuracy achieved is 82.44%, which is very close to that of 83.7% achieved by the same literature. Thus our results show that ML models like Xgboost can also produce effective results similar to the Deep learning algorithms like LSTM. Table 6 below presents the performance result of machine learning models with TF-IDF word and character level.

Table 6. Performance of ML classifiers with TF-IDF word and TF-IDF character level

Classifiers	TF-IDF word level		TF-IDF character level	
	Val Accuracy	Val F1-Score	Val Accuracy	Val F1-Score
K Nearest Neighbors	0.7513	0.7359	0.7845	0.7543
AdaBoost	0.6883	0.6076	0.6871	0.6201
XGBoost	0.7881	0.7596	**0.8244**	**0.8044**
Gradient Boost	0.6756	0.5339	0.6951	0.6009
Decision Tree	0.6677	0.5651	0.6672	0.5767
Random Forest	0.6284	0.3866	0.6484	0.4066
ExtraTrees	0.6281	0.3864	0.6581	0.4059

Presented below in Figure 18-23 is the classification report for boosting algorithms, representing the ensemble of TF-IDF word level, character level fed as a feature to machine learning models.

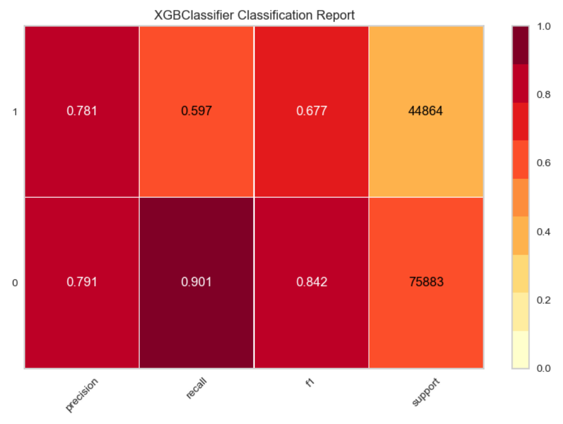


Figure 18. TF-IDF word with Xgboost

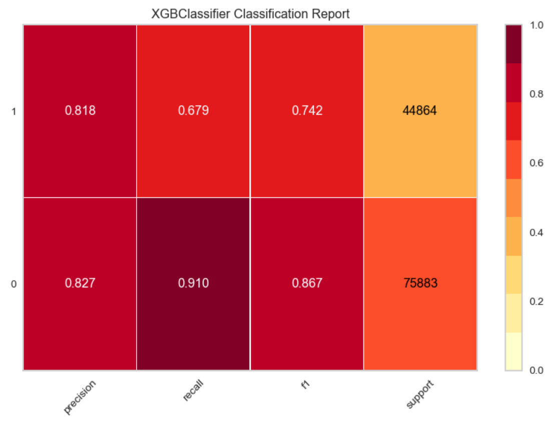


Figure 19. TF-IDF character with Xgboost

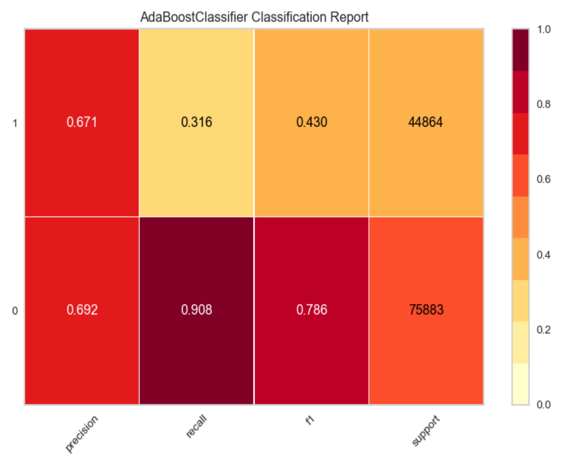


Figure 20. TF-IDF word with Adaboost

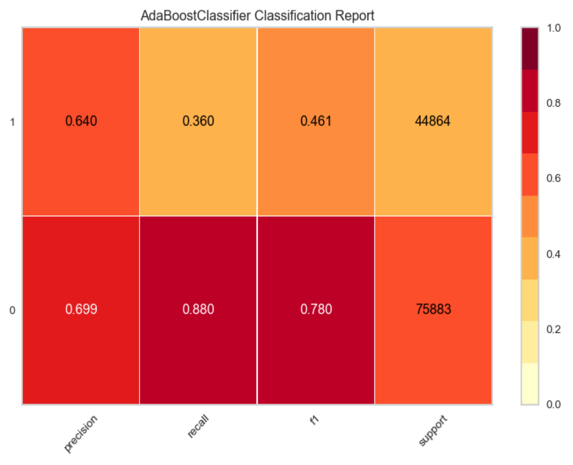


Figure 21. TF-IDF character with Adaboost

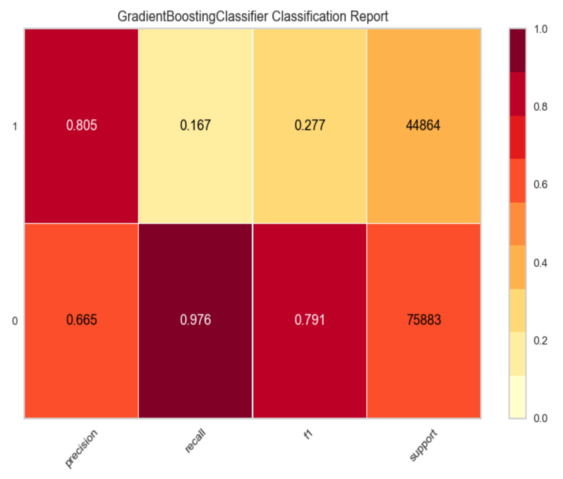


Figure 22. TF-IDF character with Gbm

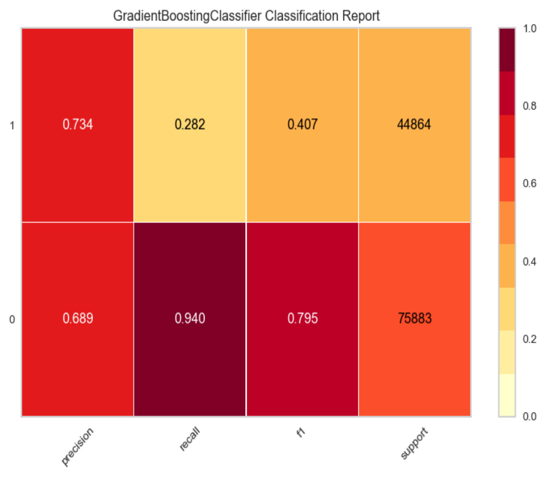


Figure 23. TF-IDF word with Gbm

Classification report provides additional performance metrics such as precision, recall, F1 for each of the class labels distribution in the test dataset.

6.6 Deep Learning Models

Table 7. Accuracy and Log loss performance of deep neural network architectures

Network	Training Loss	Training Accuracy	Validation Loss	Validation Accuracy
Architecture- 1	0.2902	0.8715	0.4062	0.8133
Architecture -2	0.2502	0.9012	0.4172	0.8312
Architecture- 3	0.1728	0.9127	0.4393	0.8522
Architecture- 4	0.0997	0.9674	0.38501	0.8582

Presented in Table 7 above, training and validation accuracy and log loss metrics obtained from the deep neural network architectures presented in Figure 10 Architecture-1, Figure 11 Architecture-2, Figure 12 Architecture-3 and Figure 14 Architecture-4 in Section 6.4. The models are trained on 80% of the dataset and validated on the rest of 20% dataset, which has been a consistent parameter for data split throughout our research work. Since we modelled and experimented with applied deep learning techniques using Tensorflow Keras python library which offers only accuracy as the metrics at the end of each epoch and finding additional metrics like F1 score require us to run additional tests on test dataset and, calculate other metrics from prediction results either manually or programmatically by using other python libraries like *sklearn metrics*[24] *or yellowbrick*[25]. In our case accuracy is the very suitable performance measure since on Quora platform, if duplicate questions are not identified then only data content duplication but if non-duplicate questions are merged as one question which also means answers will also be merged under one of the selected questions, then the problem is more critical. Quora engineering team uses accuracy as an important performance metric for this problem[6]. Therefore, identifying as many correct labels is very important. Therefore we use accuracy to measure the performance of our proposed deep neural networks.

[24] https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html
[25] https://pypi.org/project/yellowbrick/

7 Conclusions and future work

In this thesis, we investigated how the content of Quora is organized and how the problem of detecting duplicate question helps Quora presents the value of information to its users. The baseline from studies [2]–[5] has shown to achieve the accuracy of 83.8%, and we tried the series of experiments to outperform this result. We started by establishing another baseline results from the featured engineered initial dataset. Xgboost appears to be the most promising algorithm in the category of our baseline. We initially worked with a total of 30 features, including the original dataset question1 and question2. We then analyzed feature importance and selected only the top 20 features which were shown to be contributing positively in our prediction model. Re-running the experiments under the same hyperparameters tuned as in the initial baseline models has been demonstrated that eliminated features were not helpful at all for our models in boosting the prediction results. Thus our new result on baseline showed a minimal increase in Accuracy and F1 score. Our best model Xgboost gave a performance result of 74.31% and F score of 0.7349 which outperformed the literature reviewed machine learning baseline[2]

We then explored with calculating the term frequency and inverse term frequency for both the question sets and run the initial model on the same set of tuned hyper-parameters. Our results show that not all models performed well in ensemble with TF-IDF character level, but our best model Xgboost achieved the accuracy of 82.44% and F1 score of 0.8044. TF-IDF character level with Xgboost ensemble results shows that machine learning models are efficient in solving natural language problem of detecting semantically similar question and compared to other baseline achieved from few of the deep learning methods such as LSTM and LST with Siamese listed in table 1, our machine learning TF-IDF with Xgboost outperformed them.

Finally, we experimented with many different deep network layers and chose the four architecture to present which outperformed the results obtained by literature [4], our best performance from architecture-4 that achieved accuracy of 85.82%. We used log loss measures for our deep learning as an indicator of the model performance along with accuracy. We reached the best training accuracy of 96.74% and log loss of 0.09; however, in our work, the validation accuracy and validation loss is our main focus. We achieved a better result and outperformed the results from the previous study on the duplicate question pair dataset. Our best performance from this thesis work is the accuracy of **85.82% and log loss of 0.385**.

Our accuracy result is very near to the Quora state of the art[6] accuracy of 87%. While studying the literature, we realized that the main difference in results exist because Quora has used their own word embedding's from the Quora corpus dataset which is very specific to the Quora's question format, etc. whereas we have used the GloVe general embedding; thus our results are methods are more relevant to any general question and answering system.

Another way, Quora could achieve a better by pre-processing the original question pair dataset. Since knowing the context in which question is asked, a proper replacement of some of the pronouns can be done, and higher accuracy can be achieved. For example, pronoun like us, we, they can be replaced if the topic under which question exist thus replacing it with their relative context like "American," "Programmers" and "Prisoners' etc. during the pre-processing data stage can help achieve a better result. Since we are unaware in which context questions were asked we could not do such pre-processing on the original dataset.

The limitations expressed in the paragraph above if known in any context in case of any other Social Media platforms or Quora can be used as the future development of this thesis. As also we worked on standard intel core seven laptop without additional GPU capacity it took about total 32 days to train all our four deep learning models and also the TF-IDF+Xgboost model training process took close to 7 hours. With better GPU capacity, we assume to achieve a slightly better result, and the experiment could have been performed with constructing more deep learning models and parameter tuning.

8 References

[1] G. Wang, K. Gill, M. Mohanlal, H. Zheng, and B. Y. Zhao, “Wisdom in the social crowd: An analysis of Quora,” *WWW 2013 - Proc. 22nd Int. Conf. World Wide Web*, pp. 1341–1351, 2013.

[2] S. Viswanathan, N. Damodaran, and A. Simon, *Advances in Big Data and Cloud Computing*, vol. 750, no. January. Springer Singapore, 2019.

[3] A. Tung and E. Xu, “Determining Entailment of Questions in the Quora Dataset,” pp. 1–8, 2017.

[4] E. Dadashov, S. Sakshuwong, and K. Yu, “Quora Question Duplication,” pp. 1–9, 2017.

[5] T. Addair, “Duplicate Question Pair Detection with Deep Learning.”

[6] N. Jiang, Lili, Chang, Shuo, Dandekar, “Semantic Question Matching with Deep Learning,” *Blog Post*. [Online]. Available: https://www.quora.com/q/quoraengineering/Semantic-Question-Matching-with-Deep-Learning. [Accessed: 04-May-2019].

[7] M. R. Morris, J. Teevan, and K. Panovich, “What do people ask their social networks, and why?,” p. 1739, 2010.

[8] S. A. Paul, L. Hong, and E. H. Chi, “Who is Authoritative? Understanding Reputation Mechanisms in Quora,” no. 2010, 2012.

[9] M. Nicosia and A. Moschitti, “Accurate Sentence Matching with Hybrid Siamese Networks,” pp. 2235–2238, 2017.

[10] J. Pennington, R. Socher, and C. Manning, “Glove: Global Vectors for Word Representation,” *Proc. 2014 Conf. Empir. Methods Nat. Lang. Process.*, pp. 1532–1543, 2014.

[11] S. R. Bowman, J. Gauthier, A. Rastogi, R. Gupta, C. D. Manning, and C. Potts, “A Fast Unified Model for Parsing and Sentence Understanding,” *Proc. 54th Annu. Meet. Assoc. Comput. Linguist. (Volume 1 Long Pap.*, pp. 1466–1477, 2016.

[12] Y. Kim, “Convolutional Neural Networks for Sentence Classification,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[13] D. Bogdanova, C. dos Santos, L. Barbosa, and B. Zadrozny, “Detecting Semantically Equivalent Questions in Online User Forums,” pp. 123–131, 2015.

[14] Z. Wang, W. Hamza, and R. Florian, “Bilateral multi-perspective matching for natural language sentences,” in *IJCAI International Joint Conference on Artificial Intelligence*, 2017.

[15] Y. Homma, S. Sy, and C. Yeh, “Detecting Duplicate Questions with Deep Learning,” *30th Conf. Neural Inf. Process. Syst. (NIPS 2016)*, no. Nips, pp. 1–8, 2016.

[16] J. O. JOSEPHSEN, “Similarity Measures for Text Document Clustering,” *Nord. Med.*, vol. 56, no. 37, pp. 1335–1339, 1956.

[17] F. Gers, “Long short-term memory in recurrent neural networks,” *Neural Comput.*, 2001.

[18] A. Parikh, O. Täckström, D. Das, and J. Uszkoreit, “A Decomposable Attention Model for Natural Language Inference,” *Proc. 2016 Conf. Empir. Methods Nat. Lang. Process.*, pp. 2249–2255, 2016.

[19] S. Robertson, “Understanding inverse document frequency: On theoretical arguments for IDF,” *J. Doc.*, 2004.

[20] M. J. Kusner, Y. Sun, I. K. Nicholas, and Q. W. Kilian, “From Word Embeddings To

Document Distances Matt," *Washingt. Univ. St. Louis, 1 Brookings Dr., St. Louis, MO 63130*, no. 7, 2015.

[21] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "KNN Model-Based Approach in Classification," 2010.

[22] T. K. Ho, "Random decision forests," in *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, 1995.

[23] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees," *Mach. Learn.*, vol. 63, no. 1, pp. 3–42, 2006.

[24] Y. Freund and R. E. Schapire, "Experiments with a New Boosting Algorithm," *Proc. 13th Int. Conf. Mach. Learn.*, 1996.

[25] J. H. Friedman, "Greedy Function Approximation : A Gradient Boosting Machine 1 Function estimation 2 Numerical optimization in function space," *North*, 1999.

[26] T. Chen and C. Guestrin, "XGBoost : Reliable Large-scale Tree Boosting System," *arXiv*, 2016.

[27] C. Szegedy *et al.*, "GOOGLENET," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2015.

[28] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout Networks," 2013.

[29] Y. M. Chou, Y. M. Chan, J. H. Lee, C. Y. Chiu, and C. S. Chen, "Unifying and merging well-trained deep neural networks for inference stage," *IJCAI Int. Jt. Conf. Artif. Intell.*, vol. 2018-July, pp. 2049–2056, 2018.

[30] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017.

[31] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," 2015.

[32] G. Hinton, "Dropout : A Simple Way to Prevent Neural Networks from Overfitting," vol. 15, pp. 1929–1958, 2014.

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers," *Proc. IEEE Int. Conf. Comput. Vis.*, 2015.

[34] B. Karlik and A. Vehbi, "Performance Analysis of Various Activation Functions in Generalized MLP Architectures of Neural Networks," *Int. J. Artif. Intell. Expert Syst.*, no. 1, pp. 111–122, 2011.

Appendix

I.A Hyperparameters used for the machine learning models

The machine models are built using scikit-learn, sklearn python library[26].The hyperparameter values used in the experiments are presented as in Table 8.

Table 8. Hyperparameters used in the machine learning models

Classifiers	Hyperparameters
KNN	n_neighbors=10

Adaboost	algorithm='SAMME.R'
	base_estimator=None
	learning_rate=0.8
	n_estimators=50

Xgboost	learning_rate=0.1
	max_depth=5
	n_estimators=100

Gbm	learning_rate=0.1 min_samples_leaf=1
	n_estimators=100 subsample=1
	max_depth=3 max_features='sqrt'
	min_samples_split=2 random_state=10

Decision Tree	max_depth=5

Random Forest	max_depth=5
	n_estimators=100
	max_features=7

Extra Trees	max_depth=5
	n_estimators=100
	max_features=7

TfidfVectorizer character level	analyzer='char'
	token_pattern=r'\w{1,}'
	ngram_range=(2,3)
	max_features=5000

TfidfVectorizer word level	analyzer='word'
	token_pattern=r'\w{1,}'
	max_features=5000

[26] https://scikit-learn.org/0.20/documentation.html

I.B Hyperparameters used for the deep learning layers

The deep learning models are built using keras[27].The hyperparameter values used in the experiments are presented as in Table 9.

Table 9. Hyperparameters for deep neural network layers

Layer	Hyperparameters
Dense - as Intermediate Layer	output = 300
Dense - as pre-final Layer	output = 1
BatchNormalization	Default
Dropout	weight = 0.2

LSTM	output= 300
	dropout_W=0.2
	dropout_U=0.2

ConV1D	nb_filter=nb_filter
	filter_length=filter_length
	border_mode='valid'
	activation='relu'
	subsample_length=1

PReLU	Default
GlobalMaxPooling1D	Default
Activation	sigmoid

Embedding without GloVe	len(word_index) + 1
	300
	input_length=40
	dropout=0.2

Embedding with GloVe	len(word_index) + 1
	output = 300
	weights=[embedding_matrix]
	input_length=40
	trainable=False

mode.fit	batch_size=300
	nb_epoch=150
	verbose=1
	validation_split=0.2
	shuffle=True

mode.compile	loss='binary_crossentropy'
	optimizer='adam'
	metrics=['accuracy']

[27] https://keras.io/

II. License