Classification of Unstructured Text Data

REVIEW REPORT

Submitted by

Sumona Sud (20BDS0156)
Garima Agarwal (20BCE2034)

Prepared For

INTELLIGENT DATABASE SYSTEMS (BCD3006)

PROJECT COMPONENT

Submitted To

Dr. Swetha N G

Assistant Professor Senior Grade 1

School of Computer Science and Engineering

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Abstract

This project focuses on identifying and flagging duplicate questions on Quora, a platform for gaining and sharing knowledge. With over 100 million visitors each month, Quora aims to provide a space for people to ask questions and connect with individuals who offer unique insights and quality answers. However, multiple questions with similar intents can create confusion and waste time for both seekers and writers. By predicting whether a pair of questions are duplicates or not, this project aims to provide an efficient solution for seekers to find answers and for writers to focus on answering unique questions. This could be useful to instantly provide answers to questions that have already been answered. The goal is to create a better experience for active parties and offer more value to both groups in the long term. We will be training our model using the dataset from Kaggle which consists of a total of over 4 lakh data.

Table of Contents

CHAPTER 1: INTRODUCTION

1.1 Problem Statement

Quora aims to solve the problem of information overload and the challenge of finding reliable sources of knowledge by providing a centralized platform for knowledge sharing and acquisition. The platform connects individuals seeking answers to questions with people who have unique insights and can provide quality answers. Through this connection, Quora empowers individuals to learn from each other and gain a better understanding of the world. By leveraging technology, Quora aims to make knowledge more accessible, reliable, and useful, thereby bridging the gap between the knowledge that people seek and the sources of knowledge that are available to them.

Finding question similarity is necessary because many sites value canonical questions and want to improve the experience. Duplicate questions with similar intents waste time and can cause confusion. By identifying these duplicate questions, we can provide an efficient solution for users to find answers quickly and allow writers to focus on answering unique questions. Ultimately, this will provide a better experience for active seekers and writers and offer more value to both groups in the long term.

1.2 Motivation and Need for the Problem Statement

It can be difficult to identify questions that are like Quora. The problem's natural language processing (NLP) component is one of the primary obstacles. It might be difficult to distinguish between similar questions because distinct questions can have similar wording and meanings. Also, the context of the query may alter slightly, which automated systems might not be able to detect. There are millions of questions on Quora, and more are being uploaded every day, so the size of the issue is another difficulty. Hence, effective algorithms are required to process duplicates rapidly and precisely. The issue of protecting data security and privacy while handling significant amounts of user-generated material is the last one. To provide an efficient method of determining question similarity on Quora, several issues must be addressed.

Finding duplicate questions is a challenge that Stack Overflow also has. Users can ask technical questions on Stack Overflow and receive responses from the community. Stack Overflow is a well-known question and answer website for programmers and developers. Due to the high volume of questions posted on the platform, it can take a while for users to look for the best solution because there are sometimes duplicate inquiries.

1.3 Definition of Terminologies

Question similarity is the degree of resemblance or similarity between two or more questions depending on a variety of factors, such as context, context, and semantic meaning.

The level of connection or similarity between two texts' meanings, including the words, phrases, and ideas they employ, is known as semantic similarity.

Syntactic similarity is the degree of similarity or resemblance between two or more sentences or phrases based on the structure of those sentences or phrases, including their grammar, syntax, and word order.

Cosine similarity is a metric for comparing how similar two vectors are in a high-dimensional space. Cosine similarity is frequently used in the context of question similarity to compare the similarity between two questions based on their word vectors.

Word embedding is a method for representing words in natural language processing as vectors of real numbers so they may be analytically compared and evaluated.

A contiguous sequence of N items, such as a word, a sentence, or a character, from a given sample of text is referred to as an “N-gram." N-grams are frequently used in the context of question similarity to compare the similarities between two questions based on their overlapping N-grams.

1.4 Brief Description of Chapters

The literature on question similarity covers a wide range of topics, including machine learning, information retrieval, and natural language processing. Semantic, syntactic, and machine learning-based methods, including neural networks and deep learning, have all been presented by researchers as ways to measure question similarity.

To find similarities between questions based on their underlying concepts and themes, semantic approaches to question similarity place a strong emphasis on the meaning of the questions. These approaches use methods like word embeddings and semantic analysis.

A system that calculates question similarity often has several components, including a data source, a preprocessing module, a feature extraction module, and a module for familiarity measurement.

The following fields will be present in the dataset: id - the id of a training set question pair.

qid1, qid2 are the specific question identifiers (available only in train.csv) and question1, question2 are the complete question texts.

The goal variable, is duplicate, is set to 1 if questions 1 and 2 fundamentally mean the same thing and to 0 otherwise.

The present methodology for calculating question similarity is based on elementary text feature extraction techniques and machine learning algorithms. To increase accuracy, the project could be enhanced by adding sophisticated neural networks like CNNs and RCNNs as well as intricate feature extraction methods like BERT.

CHAPTER 2: LITERATURE SURVEY

Sl. No.	Details of the Paper	Methodologies Covered	Advantages	Disadvantages
1.	A Hybrid Auto-tagging System for Stack Overflow Forum Questions	The paper helps the users to tag the questions accurately and adds more tags to existing questions. This is performed by using programming language detection, and linear SVM classifier.	• 'Multinomial Naive Bayes Classifier' is good for large datasets and requires low computational cost. • Regular expressions are used to identify frequently used programming languages, which makes the matching easier.	• Low accuracy of 72% is found for the model. • SVM classifiers will require lots of time to train a model, and it is not recommended for large datasets.
2.	PTM4Tag: Sharpening Tag Recommendation of Stack Overflow Posts with Pre-trained Models	Multiple steps include. • Data gathering: The authors gathered a dataset of Stack Overflow postings and the tags that were assigned to them. • Pre-processing: Code snippets,	• Better tag recommendation: When compared to more conventional methods like TF-IDF, using pre-trained models and a neural network strategy may increase the	• Data restrictions: The quality and quantity of the training data, which may not fully reflect the range of posts and tags on Stack Overflow, may place restrictions on the

		Links, and other extraneous information were removed from the postings during pre-processing. • Tag recommendation: Using a TF-IDF based methodology, a fundamental tag recommendation model was created. • Pre-trained models: The dataset was used to fine-tune a number of pre-trained models, including BERT, RoBERTa, and DistilBERT, to provide embeddings for the posts. • Using the pre-trained embeddings as input, a neural network was trained to forecast the tags for a certain post.	accuracy of tags that are recommended. • Flexibility: Because pre-trained models can be improved on various datasets, they can be flexible in terms of the language and subject matter of Stack Overflow posts. • Generalizability: Because pre-trained models can be used to create embeddings for a variety of text data formats, the approach may be generalizable to other text classification applications outside Stack Overflow tag recommendation. • Modern: The study offers a state-of-the-art method for recommending tags	performance of the technique. • Computing power: Using pre-trained models and a neural network technique may call for a lot of computing power and training time, which could be difficult for some businesses or researchers.

		• Evaluation: Many metrics, including precision, recall, and F1-score, were used to assess the PTM4Tag model's performance. • In comparison to other innovative tag recommendation algorithms, PTM4Tag's performance was evaluated. The process, in general, involved creating a model that employed language models that had already been trained to provide embeddings for Stack Overflow postings, which were then fed into a neural network for tag recommendation. To determine the effectiveness of the method, it was reviewed and contrasted with alternative models.	on Stack Overflow, which may be helpful for academics and professionals involved in information retrieval and natural language processing.	
3.	Post2Vec: Learning Distributed Representati	Post2Vec is a deep learning architecture which extracts distributed representations of Stack Overflow posts.	For the tag recommendation task, Post2Vec achieves 15-25 percent improvement in	Effectiveness of state-of-the-art solutions for the three tasks by substantial margins: Relatedness

	ons of Stack Overflow Posts	Post2Vec integrates different types of content present in Stack Overflow posts, i.e., title, description, and code snippets to learn post representations	terms of F1-score@5 at a lower computational cost.	prediction - 10 % in terms of F1 score Post Classification - 7% in terms of F1-score API Recommendation - 10% in terms of correctness
4.	Automatic tag recommendation algorithms for social recommender systems	The paper has used two approaches, graph based and prototype-based method. Each category has tags ranked, and based on joint probabilities recommended to new input documents	• The model is reusable for different applications and systems, scalable to large websites, resulting in being effective for all of them. • Compared to other classification methods, the two-way PMM has the advantage of modelling the multivariate distribution of words at each class. This makes it capable of clustering words simultaneously while classifying documents. Resulting at	• Algorithms performed differently for different datasets namely, CiteULike, Del.icio.us and BibSonomy, showing web page tag recommendation has degraded performance than document tag recommendation. • Algorithm fails if the URL content contains necessary information (i.e., words).

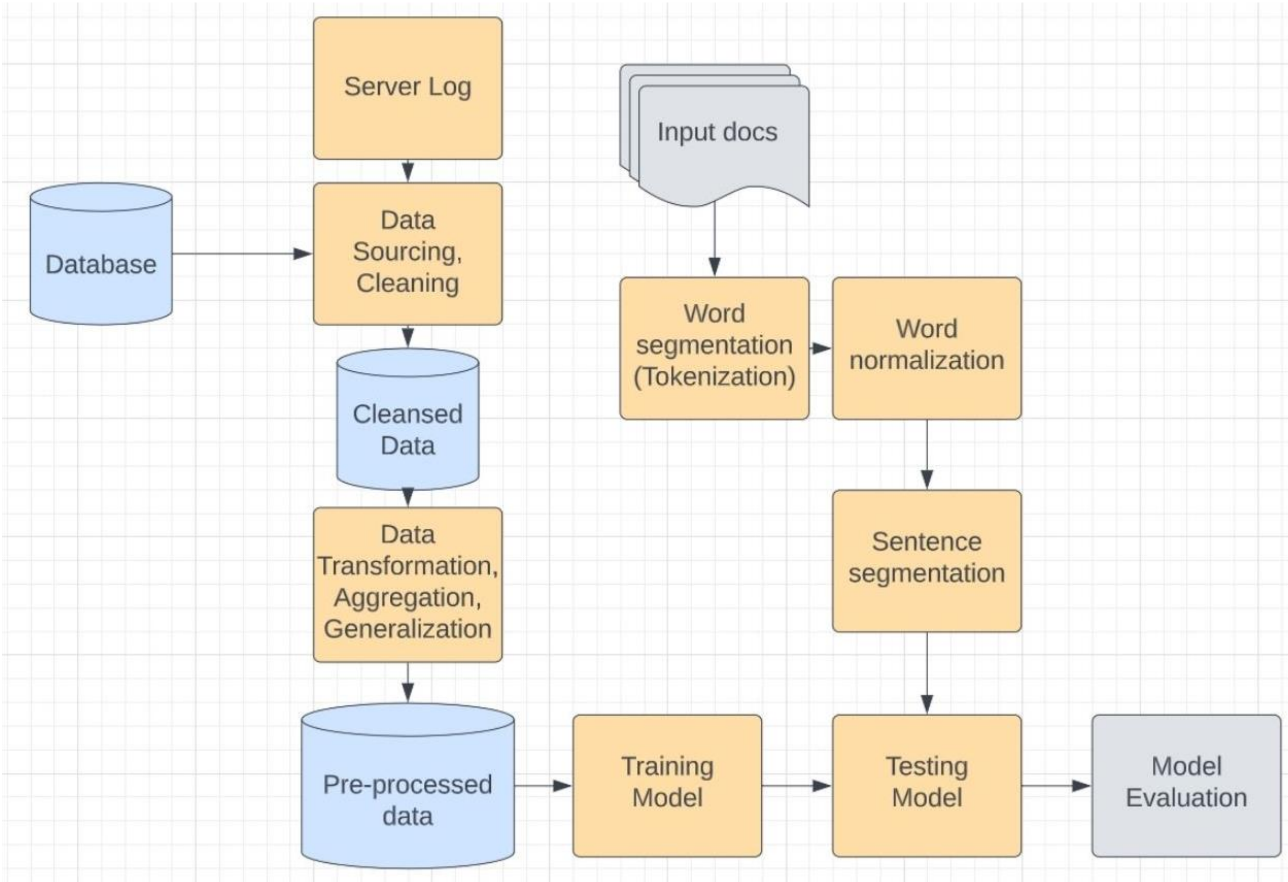
			dimensionality reduction of the document-word matrix.	
5.	Tag Stack: Automated System for Predicting Tags in Stack Overflow	The paper suggests 'TagStack system', being the most effective ML during its performance of predicting tags on Stack Overflow. It experiments on real world datasets, to make tagging of questions easier.	• Feedback is used on the system which makes the accuracy increase from 96% to 99%. • TagStack system extracts features from the questions present at training dataset such as title and body and builds a model based on Naïve Bayesian classifier. The classifier is quick and easily predicts the classes of test dataset.	• Naïve Bayesian classifier predicts 0 for categorical data in dataset. • Dataset is small, better results can be achieved.

CHAPTER 3: ARCHITECTURE DESIGN

3.1 Model Chosen

We chose Random Forest and XG Boost as the models for the project. Random Forest and XGBoost are popular classification algorithms due to their high accuracy and ability to handle complex datasets. Random Forest combines multiple decision trees, while XGBoost sequentially adds trees. The dataset that we are using for the model is a huge dataset with 30,000 rows and 6,000 columns. Hence, both random forest and XGBoost would be efficient algorithms for this task.

3.2 Architecture Design



3.3 Elaboration of Components

Approach 1: Naive-Approach:

We will be finding the Bag of words for the first question and then finding the Bag of words for the second question and will come up with an output variable Y that can have a value of either '0' or '1' which corresponds to the two questions being different and the same respectively. To extract the bag of words from the questions by using the sklearn Count vectorizer function. Since our dataset has around 8 lakh questions, the size of the bag will be arbitrarily large. Thus, to limit the size of the bag, we set the max_features value to 3000.

After getting the bag of words for question 1 and question 2 we use the y array with all the output values and apply random forest algorithm using sklearn library and check the accuracy. The accuracy was found to be 0.74 and when the same input while analyzed using XGBoost resulted in an accuracy of 0.733 rendering the approach not much impactful.

Note: We must note that we didn't perform any pre-processing or feature extraction on our input data.

Approach 2: Bag of Words with Basic Features

We will create a new modified dataset from the previous dataset already generated where the following fields are added:

● Length of Q1

● Length of Q2

● Number of Words in Q1

● Number of Words in Q2

● Number of Words in common

● Total Number of Words

● Words Sharing (Number of Words in common / Total Number of Words)

Where the Words Sharing will be calculated as the Number of Words in
Common divided by the Total Number of Words.

Approach 3: Advanced Tokenization Techniques

First, let’s understand what Token, Words and Stop Words are as follows:

Token: Total number of words

Stop words: Stop words are the words in a stop list filtered out before or after processing natural language data because they are insignificant. (e.g., is was, am, of)

Words: Token - Stop Words

E.g.: I am studying at VIT.

Tokens: [I] [am] [studying] [in] [VIT] = 5

Stop words: [am] [in] = 2

Words: [I] [studying] [VIT] = 3

We incorporate new advanced features in the dataset in approach 3. These advanced features are given below.

Token Features:

- cwc_min: This is the ratio of the number of common words to the length of the smaller question.
- cwc_max: This is the ratio of the number of common words to the length of the larger question.
- csc_min: This is the ratio of the number of common stop words to the smaller stop word count among the two questions
- csc_max: This is the ratio of the number of common stop words to the larger stop word count among the two questions.

- ctc_min: This is the ratio of the number of common tokens to the smaller token count among the two questions.
- ctc_max: This is the ratio of the number of common tokens to the larger token count among the two questions.
- last_word_eq: 1 if the last word in the two questions is same, 0 otherwise.
- first_word_eq: 1 if the first word in the two questions is same, 0 otherwise.

Length-Based Features:

- mean_len: Mean of the length of the two questions (number of words)
- abs_len_diff: Absolute difference between the length of the two questions (number of words)
- longest_substr_ratio: Ratio of the length of the longest substring among the two questions to the length of the smaller question

Fuzzy Features:

- fuzz_ratio: fuzz_ratio score from fuzzy-wuzzy
- fuzz_partial_ratio: fuzz_partial_ratio from fuzzy-wuzzy
- token_sort_ratio: token_sort_ratio from fuzzy-wuzzy
- token_set_ratio: token_set_ratio from fuzzy-wuzzy

Text preprocessing:

1. Converting symbols to words:
   - % is written as percent
   - $ is written as the dollar.
2. Converting numbers with strings:
   - ,000 is written as k.
   - ,000,000 is written as m.
3. Deconstructing words:
   - Isn’t is written as is not.
   - Don’t is written as do not.
4. Remove HTML tags if any and remove punctuations

Finally, we create our trainer, datasetcreator and predictor files which perform the following functions:

1. Trainer.py: It preprocesses the dataset using approach 3 and trains the random forest model on this data so that it can be used by the predictor.
2. Datasetcreater.py: It combines all the questions already provided in the dataset and adds newer ones when asked.
3. Predictor.py: This function takes in a question as the input and returns all the questions that are like it.

CHAPTER 4: RESULTS AND DISCUSSIONS

4.1 Dataset

4.1.1 Source of the Dataset

Kaggle Data Set Link -https://www.kaggle.com/competitions/quora-question-pairs/data

4.1.2 About the Dataset

We will be training our model using the dataset from Kaggle which consists a total of over 4 lakh data points which will be downloaded and saved in a file named train.csv. The data itself has been generated to train models to identify similarity in questions.

4.1.3 Attributes of the Dataset

● id - the id of a training set question pair

● qid1, qid2 - unique ids of each question (only available in train.csv)

● question1, question2 - the full text of each question

● is_duplicate - the target variable, set to 1 if question1 and question2 have essentially the same meaning, and 0 otherwise.

4.1.4 Rows and Columns

The dataset has about 4 Lakhs rows and 6 columns.

4.1.5 General Description and Analysis

The given data set consists of 255027 pairs that are duplicates and 149263 pairs are different, giving us statistical info of 63.080% duplicate pairs and 36.919% different pairs making our dataset unbalanced.

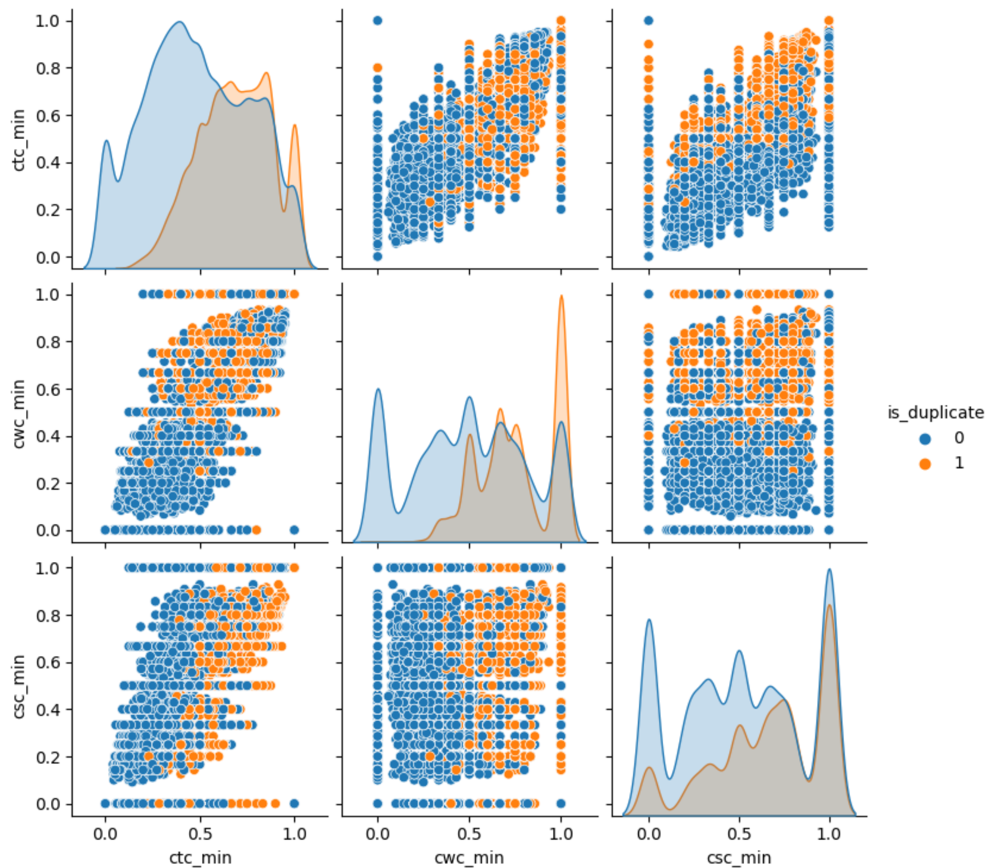
Number of unique questions: 537933

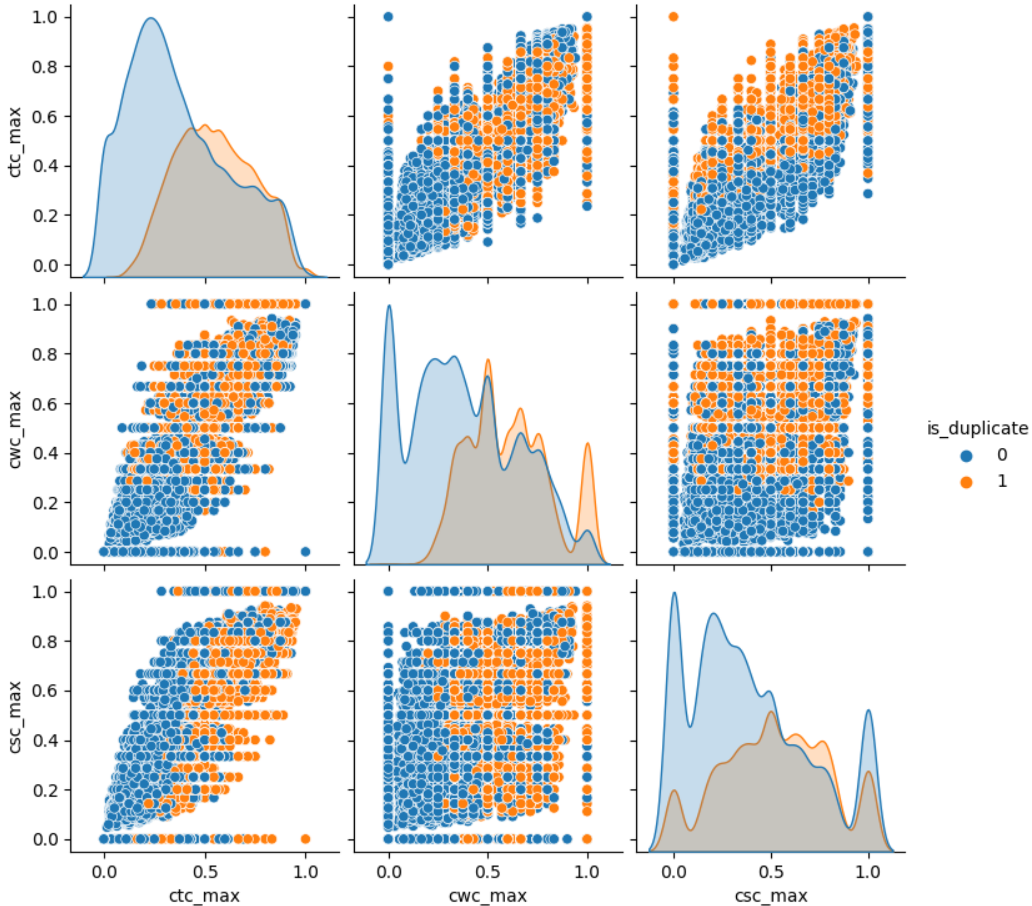
Questions getting repeated: 111780.

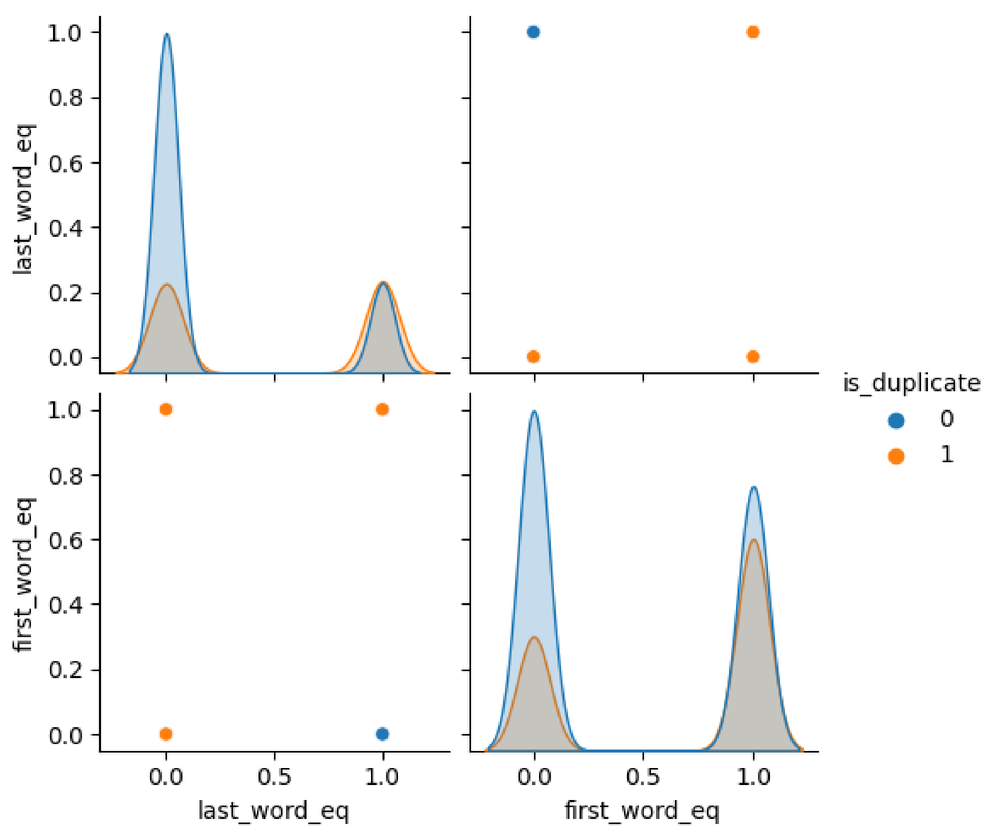
4.2 Results

Pair plots between the various columns of the dataset formed after processing via Approach 3:

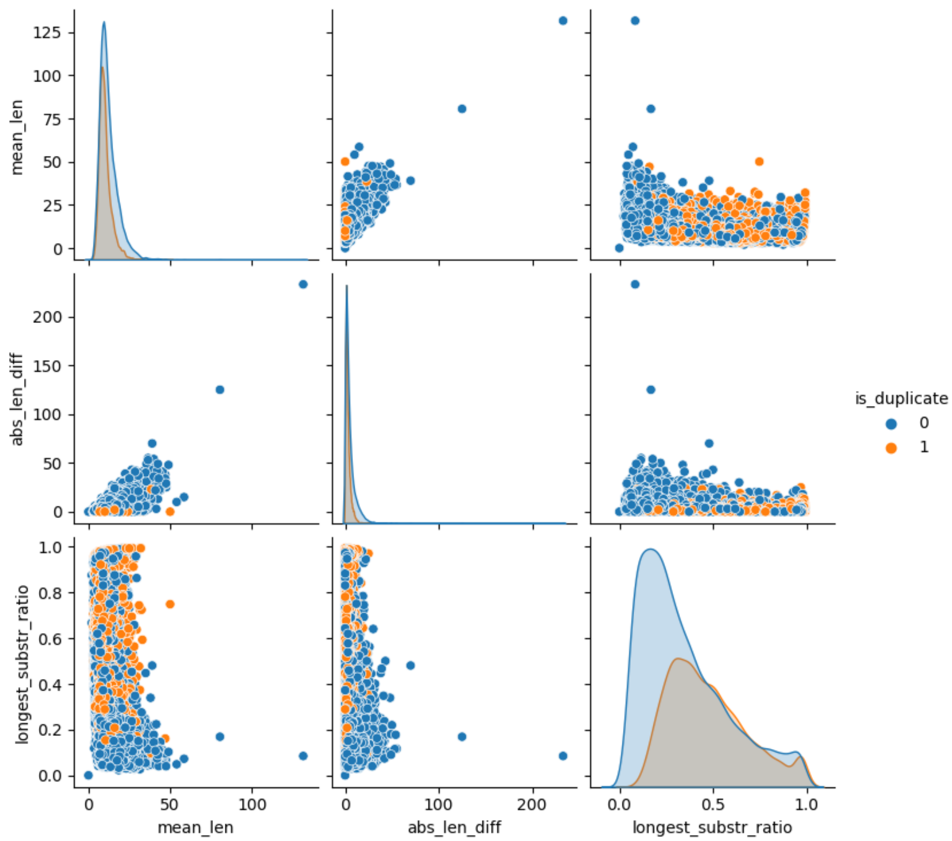
Token features:



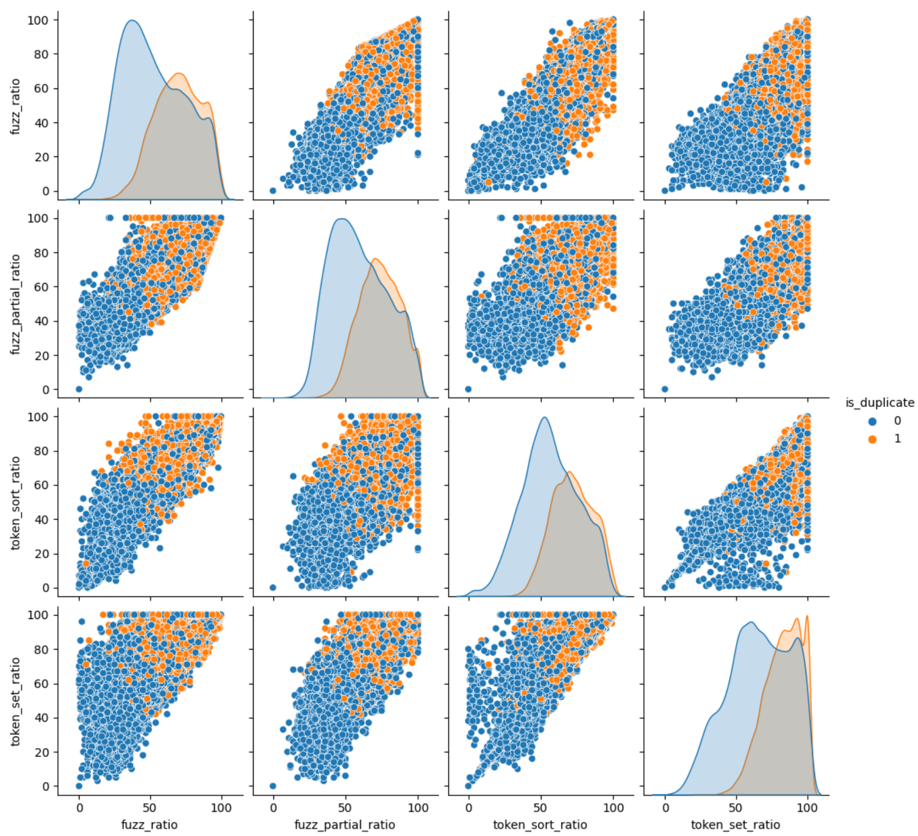

ctc_max
cwc_max
csc_max
is_duplicate
0
1


last_word_eq
first_word_eq
is_duplicate
0
1

Length Features:



Fuzzy Features:



Results of the model:

```
RandomForest Result
Accuracy:  0.7891666666666667
Precision:  0.7435356200527704
Recall:  0.6439670932358318
F1 Score: 0.6901787901053147
Confusion Matrix:
[[3326  486]
 [ 779 1409]]
```

```
XGBoost Result
Accuracy:  0.7958333333333333
Precision:  0.7300525561395127
Recall:  0.6983546617915904
F1 Score: 0.7138519037608035
Confusion Matrix:
[[3247  565]
 [ 660 1528]]
```

Accuracy:

Random forest: 0.79

XGBoost: 0.80

F1 score:

Random forest: 0.69

XGBoost: 0.71

Precision score:

Random forest: 0.74

XGBoost: 0.73

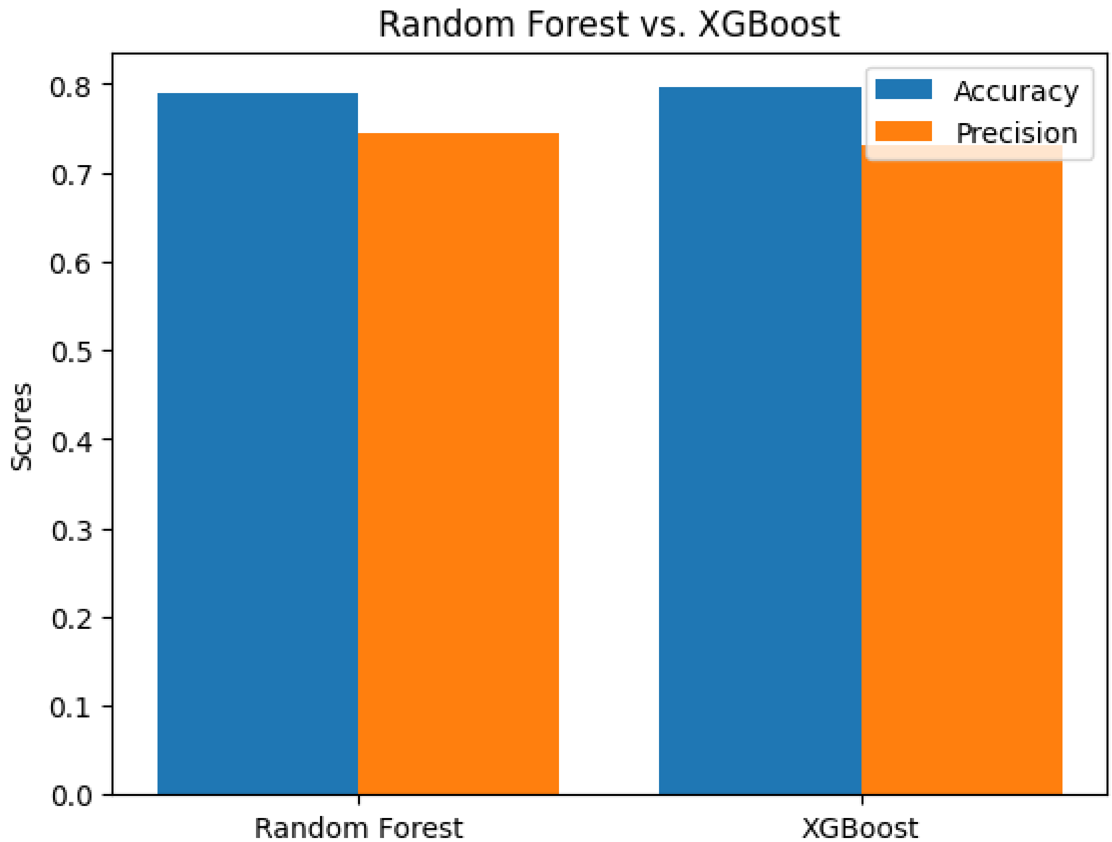
Recall score:

Random forest: 0.64

XGBoost: 0.69

	Random Forest	XGBoost
Accuracy	0.79	80
F1 Score	0.69	0.71
Precision Score	0.74	0.73
Recall	0.64	0.69

4.2.1 Graph on Precision and Accuracy



4.2.2 Comparison and Reason

We have showcased three different approaches for preprocessing the data and deriving the right features from it. The first approach uses a naive bag of words model without any pre-processing or feature extraction, resulting in limited accuracy. The second approach adds basic features to the bag of words model, which slightly improves accuracy. The third approach uses advanced tokenization techniques and fuzzy features, resulting in the highest accuracy.

Comparing the performance of XGBoost and Random Forest based on the given results, we can see that XGBoost has a slightly higher accuracy of 0.7958 compared to Random Forest's accuracy of 0.7892. However, Random Forest has a higher precision score of 0.7435 compared to XGBoost's precision score of 0.7301. This indicates that Random Forest is better at minimizing false positives.

XGBoost, on the other hand, has a higher recall score of 0.6984 compared to Random Forest's recall score of 0.6440. This indicates that XGBoost is better at identifying true positives and minimizing false negatives.

Both algorithms have similar F1 scores, but it's important to note that F1 score is a tradeoff between precision and recall.

4.2.3 Inference

The third approach of using Advanced Tokenization Features is the most effective due to its use of advanced features and techniques. It gives the highest accuracy.
The choice between XGBoost and Random Forest depends on the specific problem requirements and dataset characteristics. If minimizing false positives is a priority, Random Forest may be a better choice. If identifying true positives is more important, XGBoost may be a better choice.

CHAPTER 5: CONCLUSION AND FUTURE WORK

5.1 Conclusion

The project of identifying and flagging duplicate questions on Quora is a valuable initiative towards improving the user experience and efficiency of the platform. With over 100 million visitors each month, Quora's users can benefit greatly from having a streamlined process for finding answers to their questions. By predicting whether a pair of questions are duplicates or not, the project aims to offer a faster and more efficient solution to seekers and writers alike.

The dataset used in the project, sourced from Kaggle, is extensive, with over 4 lakh data. This provides a robust dataset for training the model and ensures that the model is able to capture a wide range of question patterns and intents.

Overall, the project has the potential to provide immense value to users of the Quora platform. With an efficient system for identifying duplicate questions, seekers can find answers faster and writers can focus on answering unique questions. The long-term benefits of this project could lead to improved user satisfaction, increased engagement on the platform, and the potential for Quora to become an even more valuable resource for gaining and sharing knowledge.

5.2 Future Work

Incorporating Neural Networks:

As we have not been exposed to deep learning and neural networks, we have not trained our model with those and have used simple ML algorithms like Logistic Regression, Random Forest and XGBOOST. We can improve this project to increase the accuracy by using CNNs, RCNNs and other complex neural networks as we learn about them in our further semesters.

Advanced text feature extraction:

For text feature extraction we have explored various methods like token feature extraction, fuzzy feature extraction and length-based feature extraction. There are even more complex feature extraction techniques

BIBLIOGRAPHY

[1] Zhang, "Duplicate Detection in Programming QA Communities.," *IEEE International Conference on Software Quality, Reliability and Security Co,* 2018 .

[2] O. Karnalim, "Syntax Trees and Information Retrieval to Improve Code Similarity Detection," *IEEE International Conference on Big Data (Big Data),* Vols. Proceedings of the Twenty-Second Australasian Computing Education Conference [Preprint], 2020 .

[3] Y. Chali, " Question-Question Similarity in Online Forums," *7th International Conference on Social Media Technologies, Communication, and Informatics (SOTICS,* 2018.

[4] X. Z. L. &. L. X. (. Wang, " Deep Learning for Question Similarity Detection in Community Question Answering.," *IEEE International Conference on Web Services* , 2018.

[5] Y. &. L. Y. (. Liu, "A Study on Question Similarity Measurements for Community Question Answering.," *IEEE 14th International Conference on e-Business Engineering* , 2017.

[6] N. A. A. K. &. K. V. R. (. Agarwal, "Unsupervised Tag Generation for Stack Overflow Questions.," *IEEE International Conference on Big Data (Big Data),* 2019 .

[7] S. Sahoo, "Text Analysis & Feature Engineering with NLP.," Towards Data Science. , 11 February 2019. [Online]. Available: https://towardsdatascience.com/text-analysis-feature-engineering-with-nlp-502d6ea9225d.. [Accessed February 2023].

[8] S. Roy, "NLP Text Similarity: How It Works and the Math Behind It.," Towards Data Science, 23 March 2021. [Online]. Available: https://towardsdatascience.com/nlp-text-similarity-how-it-works-and-the-math-behind-it-a0fb90a05095. [Accessed March 2023].

[9] R. Anurag, "NLP Answer Retrieval from Document using Python.," Analytics Vidhya, 8 June 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/06/nlp-answer-retrieval-from-document-using-python/. [Accessed March 2023].

[10] D. Das, "Cosine Similarity – Understanding the math and how it works (with python codes).," Machine Learning Plus. , 16 October 2020. [Online]. Available: https://www.machinelearningplus.com/nlp/cosine-similarity/.. [Accessed April 2023].

[11] R. Ravi, "Keyword Extraction Methods from Documents in NLP," Analytics Vidhya, 21 March 2022. [Online]. Available: https://www.analyticsvidhya.com/blog/2022/03/keyword-extraction-methods-from-documents-in-nlp/. [Accessed March 2023].

[12] Rakesh K, "XGBoost Versus Random Forest," Qwak, 12 Dec 2020. [Online]. Available: https://www.qwak.com/post/xgboost-versus-random-forest. . [Accessed 11 February 2023].

[13] S. S. Borkar, "XGBoost Versus Random Forest," Medium, 21 February 2022. [Online]. Available: https://medium.com/geekculture/xgboost-versus-random-forest-898e42870f30. [Accessed February 2023].

[14] Rashutya, "rashutyagi/Quora-Question-Pair-Similary-Case-Study," GitHub, 28 July 2020. [Online]. Available: https://github.com/rashutyagi/Quora-Question-Pair-Similary-Case-Study. [Accessed March 2023].

[15] R. M, "XGBoost Random Forest Tutorial," XGBoost Documentation, January 2021. [Online]. Available: https://xgboost.readthedocs.io/en/stable/tutorials/rf.html. . [Accessed March 2023].

[16] Kaggle, "Questions and Answers," Kaggle, 22 October 2021. [Online]. Available: https://www.kaggle.com/questions-and-answers/216451. [Accessed January 2023].