TRIBHUWAN UNIVERSITY

Institute of Science and Technology

"Duplicate Question Detection Using Random Forest Algorithm"

A Project Report

Submitted to:

Office of the Dean

Institute of Science and Technology, Tribhuwan University

Kirtipur, Nepal

In the partial fulfillment of the requirement for the bachelor's Degree in computer Science and Information Technology

Submitted By:

Arjun Shrestha (T.U. Roll No. 7925/072)

Sanjeev Roka (T.U. Roll No. 7957/072)

Sushant Khakurel (T.U. Roll No.7965/072)

Vijay Dhakal (T.U. Roll No. 7967/072)

Asian School of Management and Technology

16 August, 2019

ACKNOWLEDGEMENT

We have put much effort on this project. However, it would not have been possible without the help and support from our college administration, the supervisors and our college coordinator. We would like to extend our sincere thanks to all of them.

We are highly owed to our supervisor Mr. Surya Bam for his immense support and the guidance through the entire project. His supervision was so crucial for us to give this project a shape. It was him showing us the direction since the start. Likewise, we are thankful towards our coordinator Mr. Chakra Rawal for providing the co-operative environment and support as much as possible from his side.

We could not thank less to our respected principal Er. Anil Lal Amatya and all the faculty members including our teachers and every other individual who has helped us with our project directly or indirectly. Every mentioned members has put the helping hands for preparing this project. We present our sincere gratitude to all of them.

Thanking you,

Arjun Shrestha (T.U. Roll No. 7925/072)

Sanjeev Roka (T.U. Roll No. 7957/072)

Sushant Khakurel (T.U. Roll No.7965/072)

Vijay Dhakal (T.U. Roll No. 7967/072)

ABSTRACT

Duplicate Question Detection is the web-based platform where a user can ask questions and other users can answer them. With growing community discussion platforms and automated chat bots, it is quite common that different people may ask the same/similar question. Therefore, the system would increase the load on its database with redundant questions as well as the answerers would have to repeat same answers. Hence Duplicate Question Detection is the system that uses the basic concept of Natural Language Processing and a machine learning algorithm in order to classify whether the questions pair can be considered duplicate or not.

To sum it up, initially there will be some set of questions and their corresponding answers. The system allows a user to post a question. The post question will be then compared to every question in the existing question set. If any duplicate is found, the system displays that question-answer pair identified as duplicate. In case no duplicate is found, the user asked question is added to the set of questions in the system and can be answered by the other users. Doing so, we can minimize the size of the database as well as reduce the need of answering the same questions redundantly.

Table of Contents

List of Figures

List of tables

Abbreviation

API:	Application Programming Interface
CART:	Classification and Regression Trees
CMD:	Command Prompt
CSV:	Comma-Separated Values
CUI:	Character User Interface
DQD:	Duplicate Question Detection
GUI:	Graphical User Interface
HTML:	HyperText Markup Language
ML:	Machine Learning
NLP:	Natural Language Processing
SVM:	Support Vector Machine
TF-IDF:	Term Frequency-Inverse Document Frequency

CHAPTER ONE
INTRODUCTION

1.1 Introduction

Recent applications of natural language processing present a need for an effective method to compute the similarity between very short texts or sentences. An example of this is a conversational agent/dialogue system with script strategies in which sentence similarity is essential to the implementation. Sentence similarity will have internet-related applications [1] as well. In Web page retrieval, sentence similarity has proven to be one of the best techniques for improving retrieval effectiveness. In text mining, sentence similarity is used as a criterion to discover unseen knowledge from textual databases. In a context of customer support via a chat channel, with the help of duplicate question detection, previous interactions between customers and human operators can be explored to provide an increasingly automatic question answering service. Another major application of computing the sentence similarity can be to find the questions that have the same semantic meanings and can be considered duplicate. This gives rise to our project entitled Duplicate Question Detection (DQD).

Duplicate Question Detection (DQD) is the system, which warns about duplication of questions. The main task of the DQD project is to predict whether two questions can be considered same or not. This shall help us identify whether a similar type of questions ever existed in the past, or not. DQD aims to fetch a question as an input, pair it with existing question, process them to gather their significant features and finally put feed them to our algorithm to make the final prediction. Some obvious text-based features are a number of words matching between the two questions, length of the two questions, number of words, number of characters, number of stop words. It can be done with Tf-idf scores, but that is both not very useful and computationally expensive. Instead, one of the most important features that can be used as weighted word match share [2], where each word's weight is the inverse frequency of the word in the corpusif the user has a rare word in both the

questions, they might be discussing similar topics. It also requires features like cosine distance, Jaccard distance, Euclidean distance etc.

The proper implementation of the available algorithms and a new technique is a big challenge in such natural language processing. Time complexity, space complexity, resource management add further complexity in the projects.

Machine Learning (ML) is expected to bring heavy changes to the world of technology. Machine learning is a subfield of artificial intelligence and computer science that allows software applications to be more accurate in predicting results. The prime objective of machine learning technology is to build algorithms that can get input data and leverage statistical analysis to predict an acceptable output value. Fortunately, we could implement some machine learning features that are already available and give our project a direction.

1.2 Problem Definition

Many interactive web sites and applications provide users with a platform to ask questions that other users on the site may answer. However many of the questions being asked at any time might have already been asked with different wording and phrasing. Two questions can be duplicate when question askers express the same intention when writing the question.

Since finding out the intention of the questions could be quite ambiguous to consider its syntactic dimension only. Rather, computing semantic similarity between the question would give a more precise decision on whether they can be considered the same or not.

Given a questions pair, indicate whether they are same or not. For example:

Question 1: Who will win the election 2019 in India?

Question 2: Who has the most chance of winning an election in India 2019?

Given this pair, the model should indicate them as same questions as basically, both questions seem to seek for the same answer.

1.3 Objectives

The major objectives of our project is to allow the user to ask the question and detect whether the question has already been asked to the system or not i.e. detect the duplication.

1.4 Scope and Limitation

There are many platforms where a person can express his/her questions and misunderstanding related to several domain. Such platforms are mostly termed as community Discussion forum or Community Q/A Forum [1]. In such platforms, it might not always be guaranteed that each time different question is asked and each question is answered only once. Therefore, it would be simpler if the questions that repeats would be detected and informed so that the answerer does not have to answer the same question repeatedly.

Thus, our system is a step towards detecting such duplication. Our system will be able to predict, whether the given question is duplicate or not. However, the system have some limitations:

- Might classify the question pair based on irrelevant features.
- Since the user provides the data (question), it might be quite noisy which might make the system hard to preprocess and ultimately predict badly.
- Might take lot of time to prepare itself to give out the predicted output.

1.5 Report Organization

In chapter 2, we talk about the feasibility study that is needed for constructing the system along with the functional and non-functional requirements. It also includes the structuring system requirement i.e. process modeling and data modeling.

Chapter 3 includes the system design, i.e. the architecture of the system, input output definition and the user interface for the user. Data Collection, tools used for the development, algorithm implementation and the testing of data in the system is described briefly in the chapter 4. Chapter 5 consists of the conclusion and enhancements that can be made in the system in the future.

CHAPTER TWO
REQUIREMENT ANALYSIS AND FEASIBILITY STUDY

2.1 Literature Review

Since the Quora released its dataset [3] in 2017, many developers, machine-learning as well as deep learning experts have contributed lot and proposed different model for prediction the question pair duplication. We can find numerous approaches on how this problems are solved using several machine learning and deep learning algorithms. The prediction depends on several features that are extracted from the data and how different algorithms work on those features. The various related approaches and review of our project is discussed in this chapter.

2.1.1 Identifying Duplicate Questions: A machine learning case study

Here [4], the author has discussed the problem in brief and invoked some machine learning methods to solve the problem. The author has first cleaned the data and then preprocessed them to get the useful features. Under, features the author has come up with exciting features as fuzzy wuzzy [1] features for measuring similarity. Finally, the author has implemented some machine learning techniques as xgboost [5], logistic regression [6] etc. and deep learning techniques as keras, tensor flow, hyperas etc.

2.1.2 Quora Question Pair Similarity

The author in this research [7] has indicated that this is a binary classification problem. He has gone through all the data and tried to find out the major pattern in the data such that the prediction would be more accurate on using that pattern by the model. He has removed Html tags, punctuations, expanded contractions etc. to clean the text. The author has implemented Linear SVM with hyperparameter tuning along with several other ML algorithms.

2.1.3 Random Forest Algorithm

In this article [8] the author has explained very briefly about the random forest, how it can be used for solving different kinds of problems as classification and regression. The article describes what parameters the random forest take and what each parameter means.

2.1.4 Siamese Neural Networks with Random Forest

The other paper entitled Siamese Neural Networks with Random Forest for detecting duplicate questions pairs [9] has used Gated Recurrent Units (GRU) in combination with other highly used machine learning algorithms like Random Forest, Adaboost and SVM for similarity prediction task on a dataset released by Quora. They pointed out that they achieved the best result using the Siamese adaptation of a Bidirectional GRU with a Random Forest Classifier. They showed the comparison based on the Cross-Entropy Loss.

Method	**Cross-Entropy Loss**
GRU + SVM	0.27761
GRU + Adaboost	0.64782
GRU + Random Forest	0.24365

Fig 2.1: Table of comparison using 3 different algorithms

2.2 Requirement Analysis

A requirement analysis is a description of a prediction system to be developed, laying out functional and non-functional requirement, and may include a set of use cases that describes about how duplicate question detection works within the system. This requirement analysis enlists enough and necessary requirements that are required for project development. To derive the requirements, clear and thorough understanding of the system to be developed or being developed is needed. This is achieved and refined with detailed and continuous communications with the project team.

2.2.1 Functional requirement

Functional requirements are the statements of services that the system should provide, how the system should react to particular input and what the system should give output according to the input. It includes the functions performed by the particular module.

2.2.1.1 Use case diagram

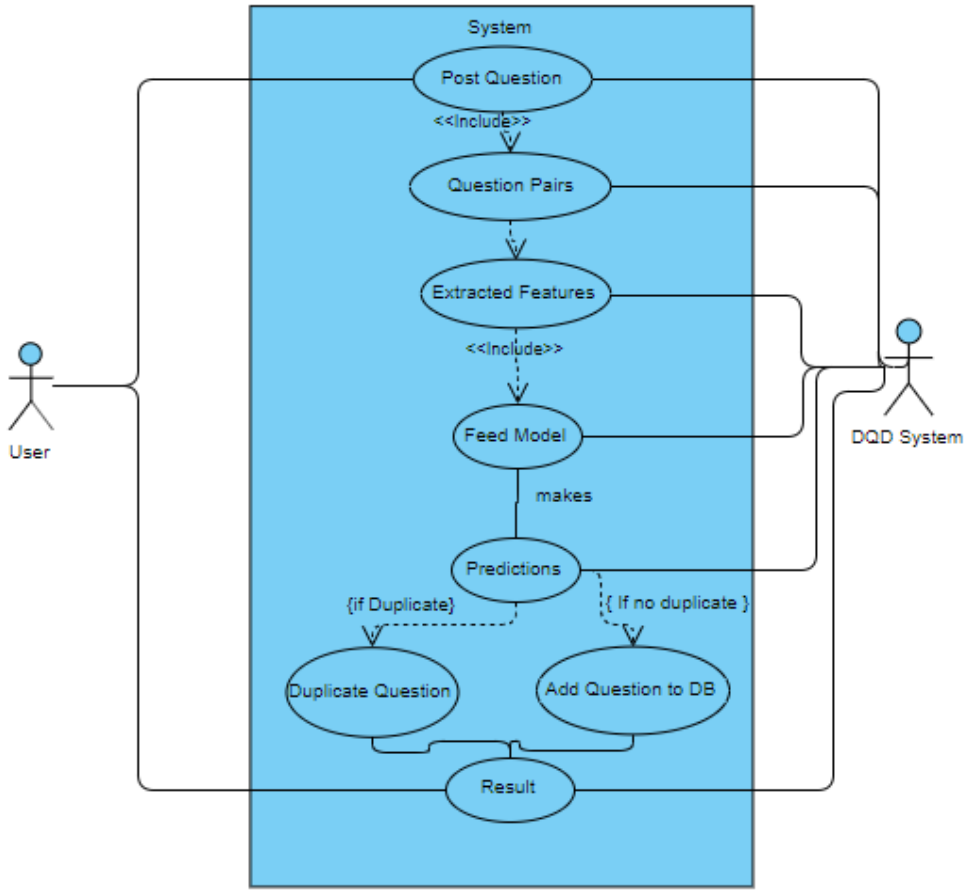


Fig 2.2: Use Case Diagram for Duplicate Question Detection

2.2.2 Non-Functional requirement

There are constraints on the services or functions offered by the system. It describes overall qualities or attributes of the system and how well or to what standard a function should be provided. The non-functional requirements used in Duplicate Question Detection are as follows:

2.2.2.1 Interface

Interface could have been designed in web using Django or desktop application like gui using tkinter or even simple CUI in CMD. However, the most important part is that it should be user friendly and easy to use.

2.2.2.2 Performance

The system, which is developed, must be able to provide information to the user and the information displayed must be accurate and correct.

2.2.2.3 Usability

The system is for duplicate question detection so it must be able to predict correct output.

2.2.2.4 Adaptability

The system should work on all major web browsers such as Firefox, Chrome, etc. if implemented in web. Likewise, it should work on any operating system if implemented using GUI interface.

2.3 Feasibility Study

The feasibility study determines if the system can be built successfully with available cost, time and effort. The study is conducted by analyzing the collected requirements.

2.3.1 Technical Feasibility:

All the tools and software product required to construct this project is easily available in the web. It do not require special environment to execute. It needs an IDE. All these aspects are easily affordable. The application requires simple user interfaces but implementation of algorithm and real calculation are complex. It can be done with some assistance from our supervisor.

2.3.2 Operational Feasibility:

The system is reliable, maintainable, usable, sustainable, supportable and affordable. Therefore, this system is operationally feasible.

2.3.3 Economical Feasibility:

As this system is not tested in the working field and no more equipment needs to be bought, also the system is made from the equipment's that is already present, so the new system is economically feasible.

2.4 System Requirement

The system to be built requires several resources. The requirements can be vastly categorized into two major parts.

2.4.1 Software Requirement

Following are the software requirement necessary for the project.

- Python Programming Language.
- Python IDE
- Python APIs
- Windows/UNIX OS (Operating System)
- Jupyter Notebook

2.4.2 Hardware Requirement

Following are the hardware requirement necessary for this project.

- Computer Device

2.5 Structuring System Requirements

2.5.1 Data Modeling

2.5.1.1 Class Diagram

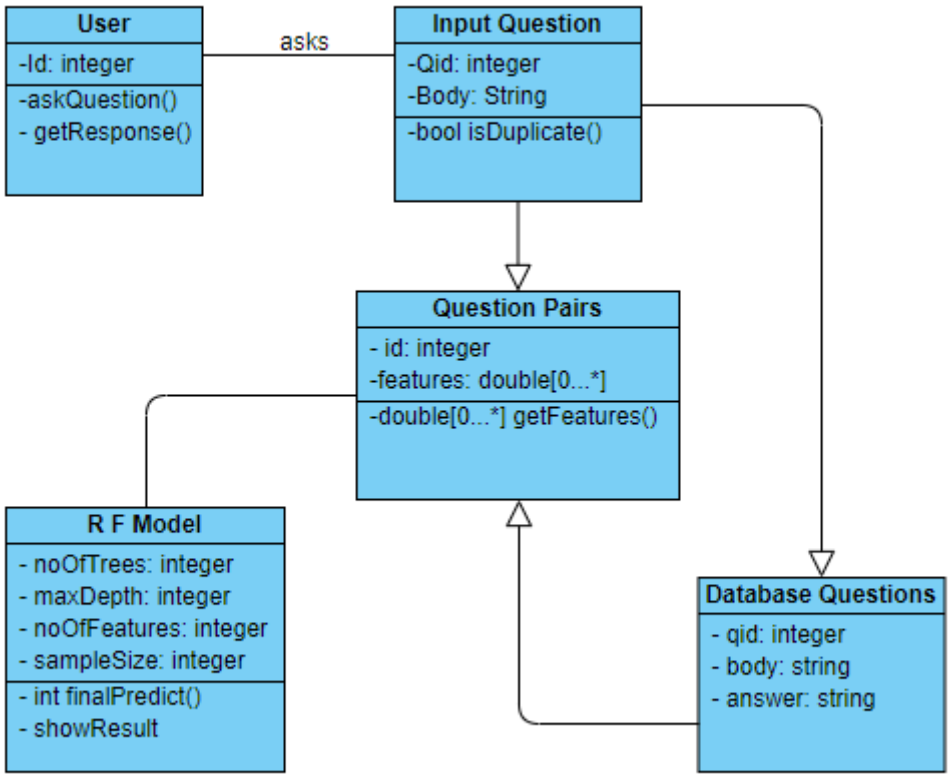


Fig 2.3: Class Diagram

The data modeling part focuses on showing how the data has been involved and maintained in the system. For this, we have constructed a simple class diagram where each class represents the data and its attributes shows the properties of that class. Likewise, the methods show how they can behave. The major data source are the questions asked by the users to our system.

2.5.2 Process Modeling

2.5.2.1 Sequence Diagram

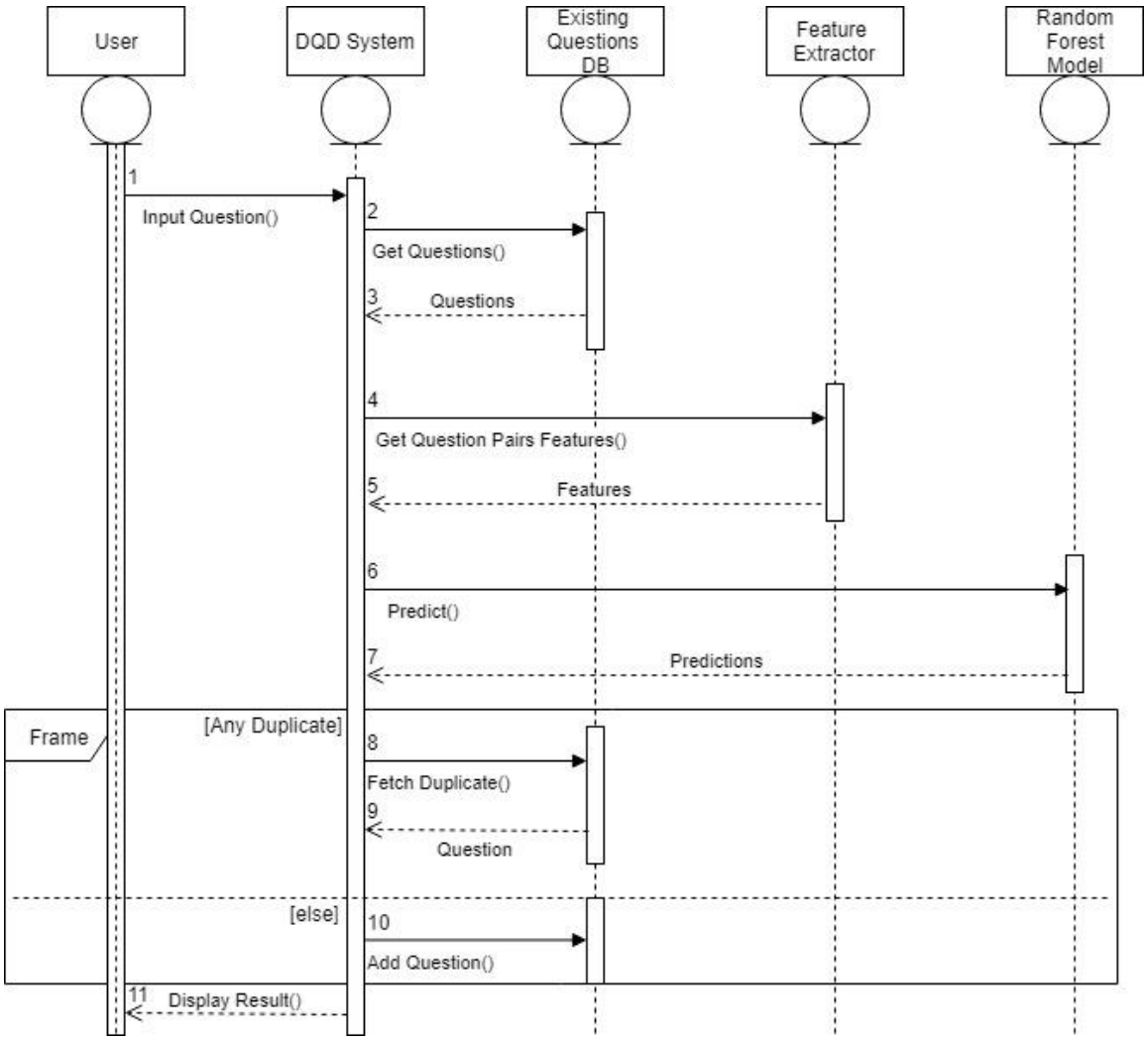


Fig 2.4: Sequence Diagram of DQD

CHAPTER THREE

SYSTEM DESIGN

3.1 System Architecture

The system takes in a question as an input. The question is taken in string format, which is then paired with every other questions in the database. The paired questions are sent for the preprocessing in order to generate the appropriate representation, which could then be fetched onto our actual prediction system. That is the trees in the random forest. Finally, through the data sent to the system, prediction is made on either the given questions were duplicate or not.

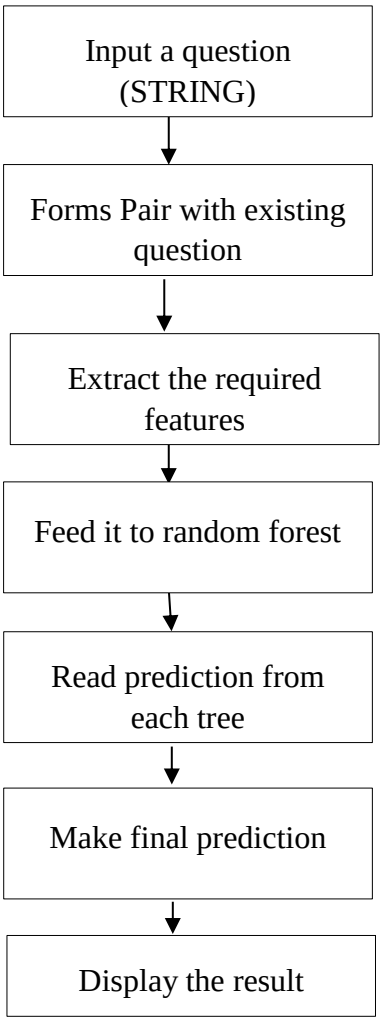


Fig 3.1: Architecture of Duplicate Question Detection

3.2 Interface Design

We have used Python Django framework for creating a simple looking web application. The interface consists of a home page where a user can see the previously posted questions and answers and link to the page where user can input Question.

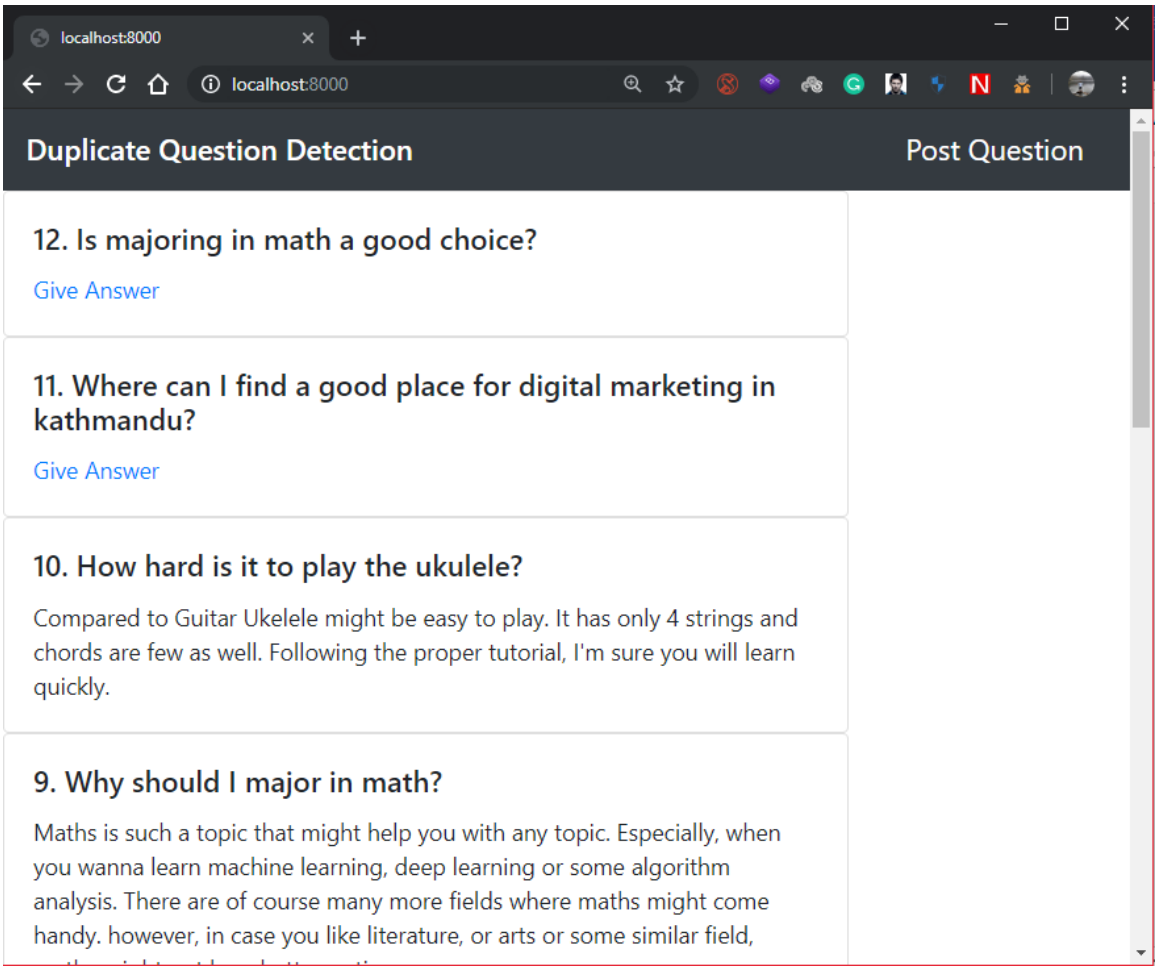


Fig 3.2.1: Interface Home Page

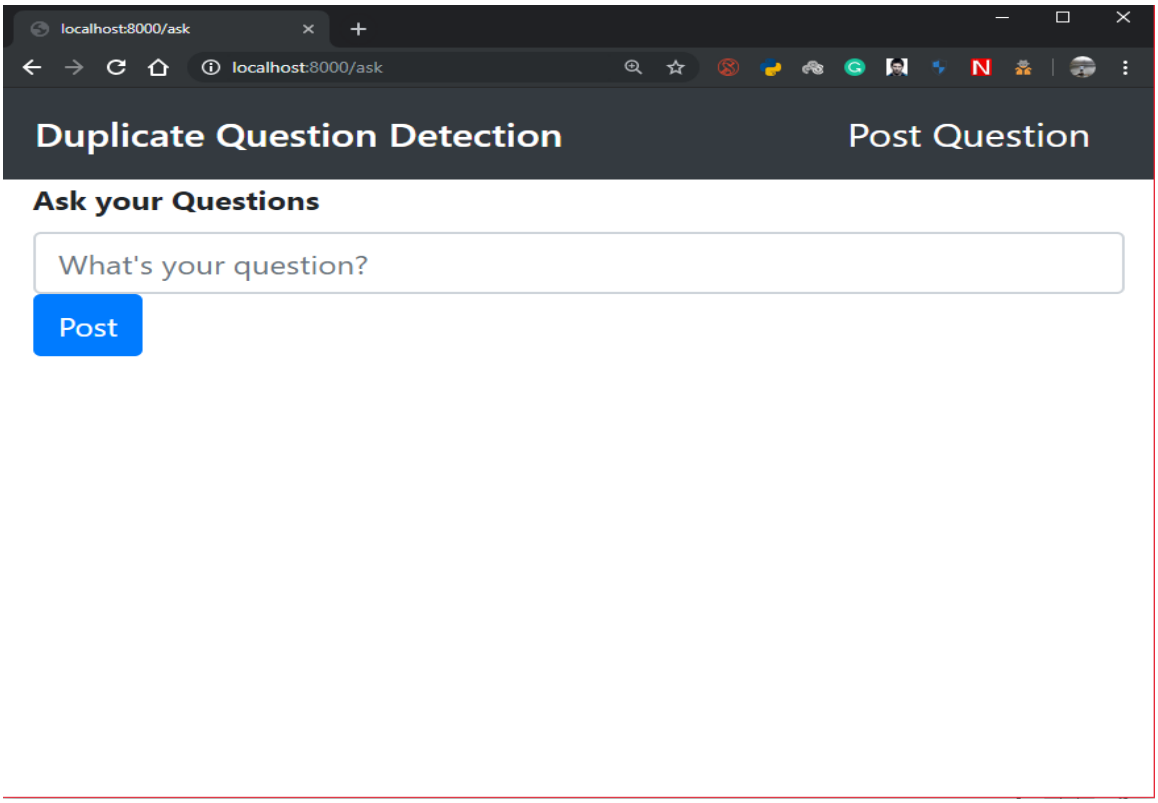


Fig 3.2.2: Interface User Input Page

3.3 Input Output Design

Data to our system is given in a string format, which is simply a question. The question is feed to model for the prediction. Observing the predictions, if any duplicate questions are found from the stored set of questions, the system will show those duplicate questions the user. In case there is no duplicate, the user question is added to our existing set of questions.

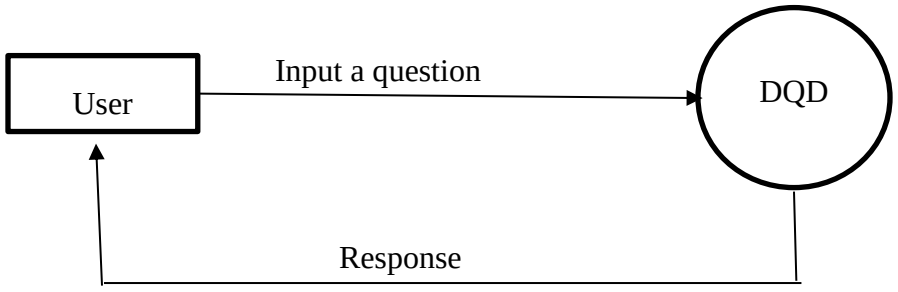


Fig3.3: Input Output Design

3.4 Process Design

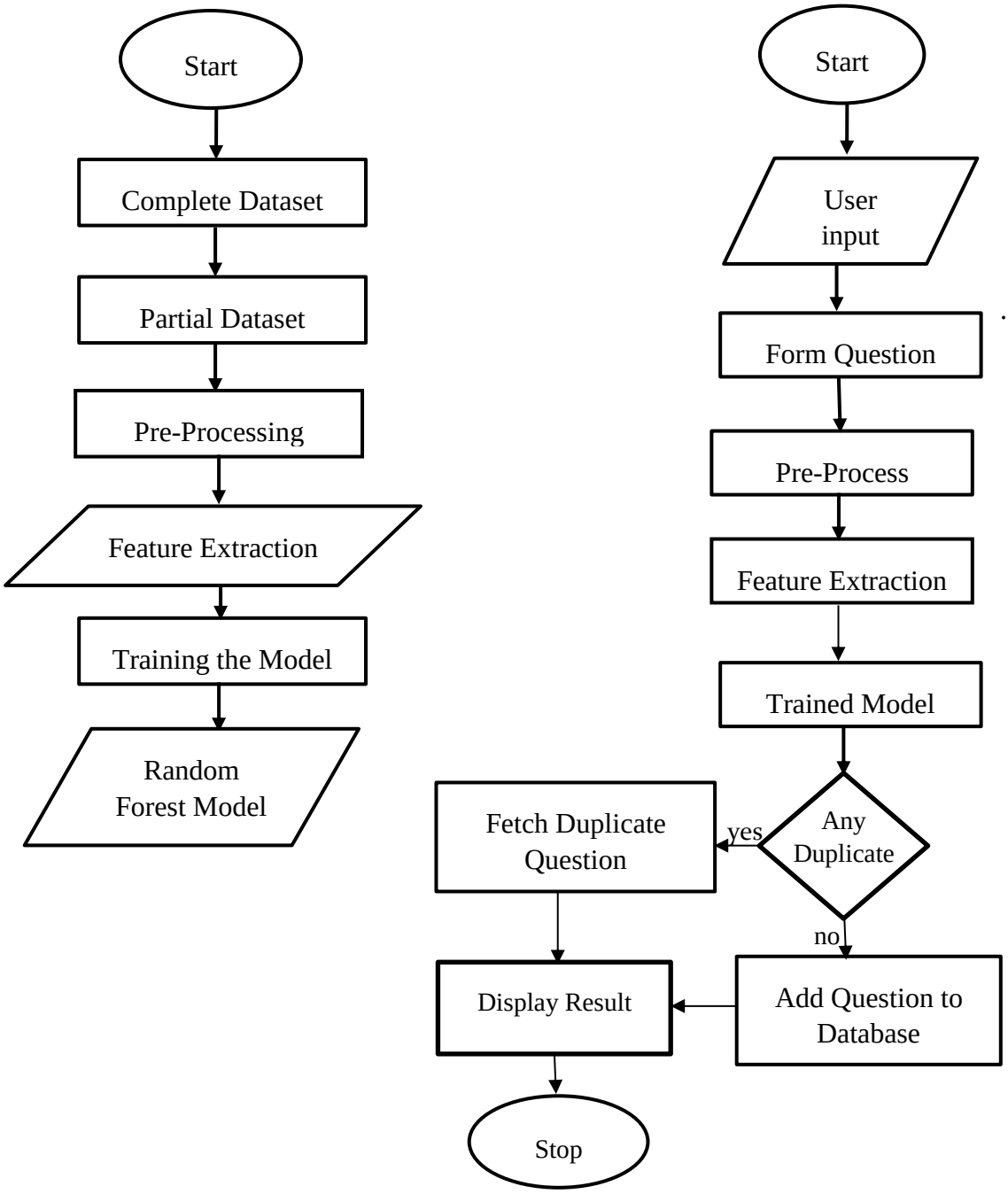


Fig 3.4: Flow Chart of Duplicate Question Detection

CHAPTER FOUR
IMPLEMENTATION AND TESTING

4.1 Implementation

User input was taken as single questions. Then the question was preprocessed tokenize (d and converted into vector, using google wordtovec [10] pre-trained model). After completing this, we extracted the features between available questions in system and the user input. Then the features were used in calculating similarity between them using Random Forest Algorithm. Finally, the result was displayed to the user accordingly.

4.1.1 Description of Algorithm used

Since our project can be considered as a binary classification problem where we classify our input in either of the two categories or classes. Duplicate or Not Duplicate. For our problem, we have numbers of algorithms to perform classification. However, we chose to go with the Random Forest due to its simplicity and ability to solve our problem. Brief discussion about the algorithm is given below.

Random Forest is extended form of decision trees. It is a CART algorithm short for classification and regression trees. Meaning that it can be used for both types of problems. The algorithm uses a tree to make a prediction or generate the result.

Each non-leaf node in this tree is basically a decision maker. These nodes are called decision nodes. Each node has one attribute and a value which is used to make decision on where to go next. Depending on the test, we either move to the left branch or the right branch of that node. Since we can constructing a classifier, each leaf node of our decision tree in the random forest represents a class.

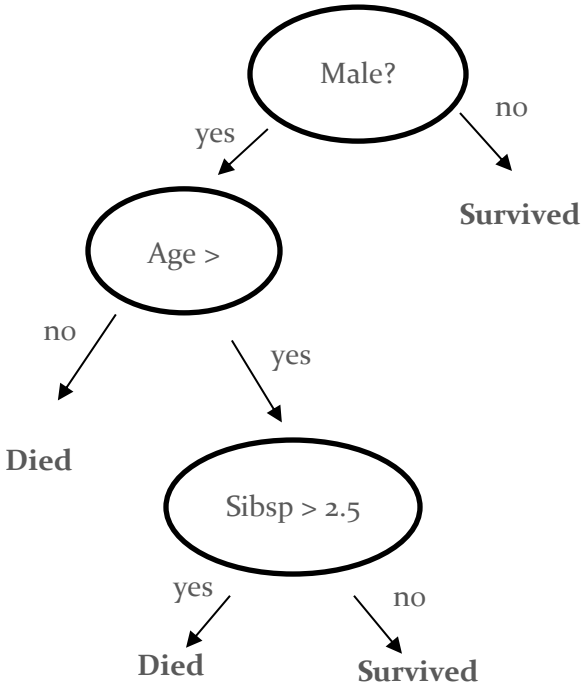


Fig 4.1: Decision Tree Example

One major issue is how we can construct an optimal tree. What should be considered before selecting an attribute based on which decisions are to be made at each node. There are numbers of approaches to do this. There are four major approaches:

I. Gini Index
II. Chi-Square
III. Information Gain
IV. Reduction in variance

Among these, our random forest uses the Gini index to split the dataset at each node. It is also called as cost function that evaluates our split. A Gini score gives us the idea of how good our split is by telling how mixed the classes are in the two groups that is created by our split. Less the Gini value, better will our split gets. The split is considered worst if the

results in the 50/50 classes in each group formed by that split. Gini Score can be calculated using the given formula.

$$\textbf{Gini(t)} = 1 - \sum_{i=0}^{c-1} \left[p\left(\frac{i}{t}\right) \right]^2$$

Where,

c = Number of classes

i = Number of data in split belonging to i^{th} class

t = total number of data in the split

	parent
C0	6
C1	6

Suppose there are two possible splits. A and B

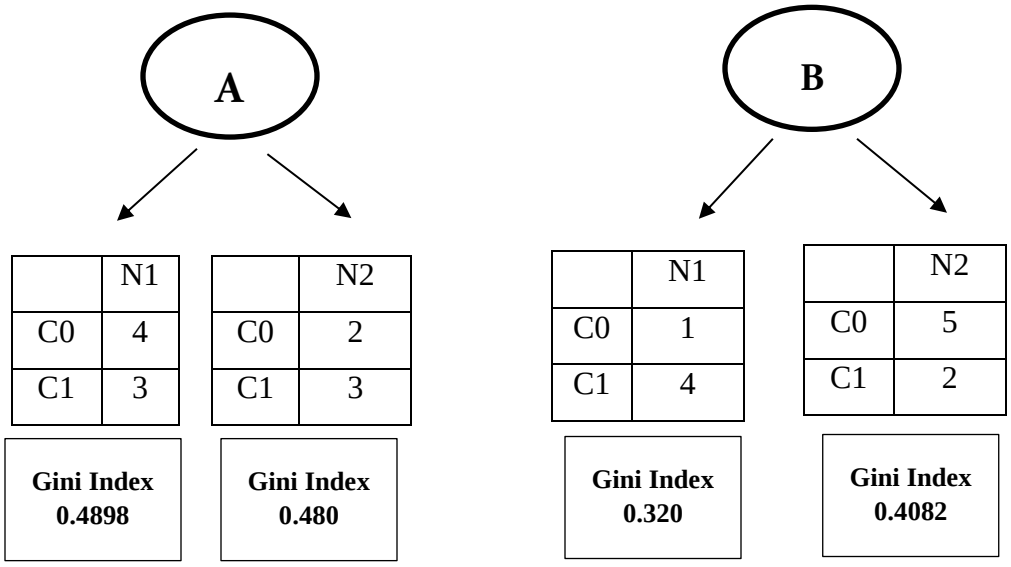


Fig 4.2: GINI index calculation

Computing the weighted average of the Gini index of both attribute, we select the one, which gives less GINI value. At each node, we calculate the Gini index for each attribute and finally construct the decision tree.

Finally, computing multiple decision trees, we create a forest of trees known as random forest. The prefix random comes in the sense that, every time the decision tree is to be trained, instead of taking the whole dataset, only fraction of dataset is taken randomly and used for that particular decision tree to train. Likewise, every tree uses only some of the features at once, most often the square root of total available features. Hence this reduces the chance of overfitting our decision tree.

Working of algorithm

- Calculate Gini Index for each attribute
- Take attribute with the lowest gini index as root node
- For splitting through nodes,
    - Take the attribute with the lowest gini index as splitting node
- Repeat the steps until Gini Index = 0.

Step 1: Calculate the total positive and negative results on the split on a attribute.
Example: For fuzz_ratio

Positive = 1504
Negative = 2294
Total samples = 3798

Step 2: Calculate the gini index for that attribute
Gini index (fuzz_ratio) = 0.473
Step 3: Compare the gini index for each attribute, select the gini index with the lowest value, and split on that node. In this particular case, we have the attribute, fuzz ratio, with lowest gini index of 0.473. So split on that attribute.

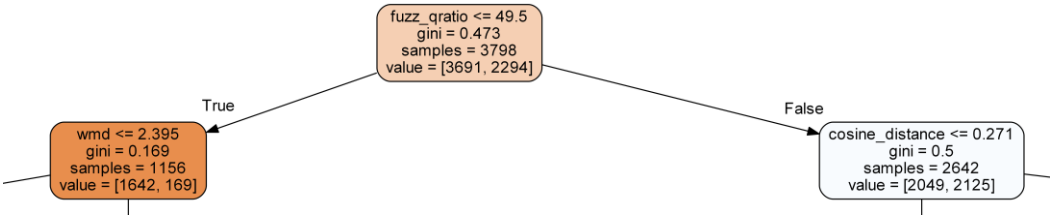


Fig 4.3.1: Root Node Split

Step 4: Repeat above process with respect to root node attributes. With respect to word mover distance and corresponding value 2.395,

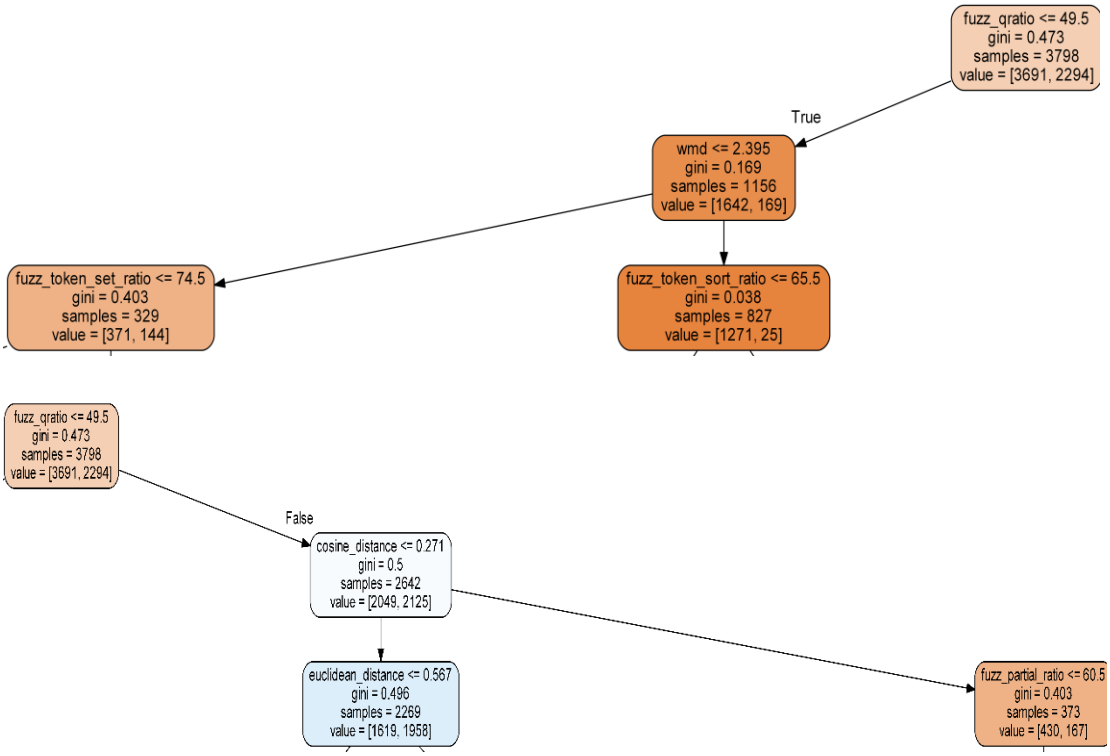


Fig 4.3.2: Child Node Split (Level 1)

In this way, we construct the decision trees. Finally, we created a forest of trees.

4.2 PHASES IN DUPLICATE QUESTION DETECTION

The final prediction is not generated form a single step. The system has to undergo through several phases in order to function as expected. The system includes the following phases.

4.2.1 Training Phase

- Data Collection: Quora has provides us dataset [3] with more than 400k question pairs. Using all of them is quite not feasible for us operationally and technically. So we selected only 8000 pairs of questions in random to train our random forest.
- Data pre-processing: The datasets provided are noisy. Hence, some cleaning needed to be done. We cleaned the data to some extent using regular expression. For example we replaced isn't with is not, we 'are with we are, US with America, e.g. with example, removed hyphens (-) and so on. In addition, we removed the stop words from our text except for words like what, why, where, how, when, has, had, have etc. since they might be crucial while representing a question instead of simple statements.
- Feature Extraction: The cleaned text then was used to extract the features. Altogether, we extracted 22 features. To extract them we used some APIs provided by the python like nltk, scipy, genism etc. We used word2vec [10] tool with 3M word list in Google's pre-trained model as a corpus, for vectorising our data. Using the vector representation, we calculated some distance based features.

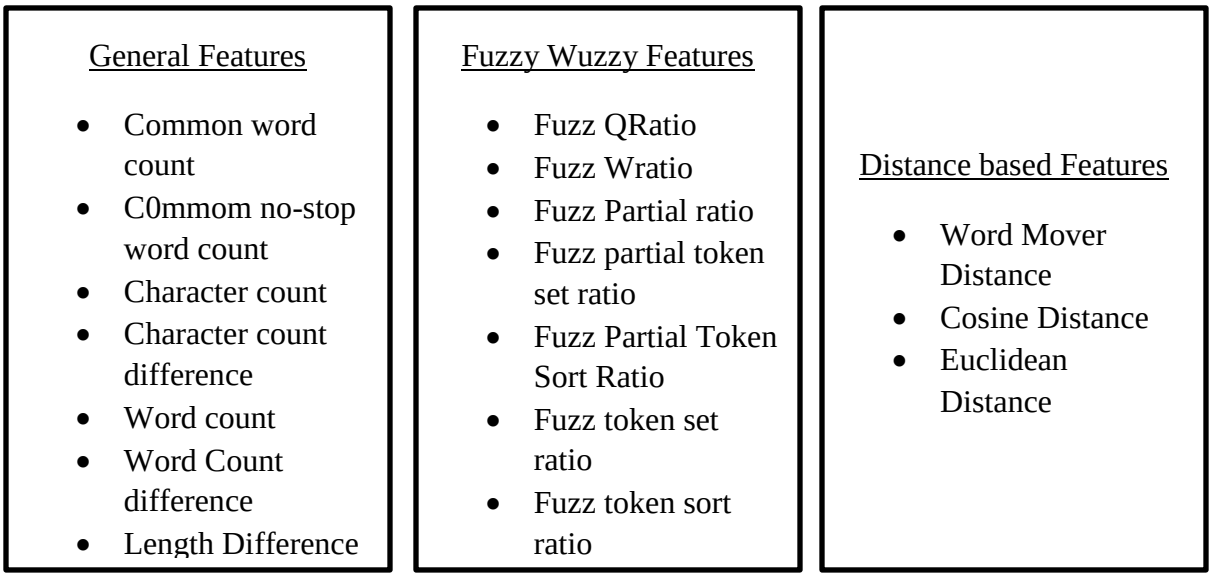


Fig 4.4 List of Features

- Training Random Forest: Finally, we used those features to construct the random forest. In our forest we set our important parameters as:
    - Number of features : $\sqrt{Number\ of\ features}$
    - Number of trees : 15
    - Maximum depth : 5

4.2.2 Prediction phase

- Input: Input from the user is a question that is read in string format
    - Input Question: Where can I find a good place for digital marketing in Kathmandu?
- Pairing the questions: The user input question is paired with every other questions existing in the database so that the similarity between each pair could be calculated based on several features.
- Pre-processing: In order to feed our input to trained model it must be first preprocessed to extract exactly those features, which were used while training that model. The preprocessing includes:

    - Lower Casing every words in the sentence
        - Converted: where can i find a good place for digital marketing in kathmandu?
    - Stop Words Removal:
        - Stop words: [where, can, a, for, in]
        - After removal: i find good place digital marketing kathmandu
    - Tokenization
        - Tokens: [i, find, good, place, digital, marketing, kathmandu]
    - Lemmatization
        - Marketing => market
    - Vectorization
        - The input question were converted into vectors using a google word to vec pre trained model.

- The preprocessed question pairs are then used to extract the features. We calculate 17 different features as,

	common_word_cnt	common_nonstop_word_cnt	char_cnt_1	char_cnt_2	char_cnt_diff	word_cnt_1	word_cnt_2	word_cnt_diff	len_q1	len_q2
0	4	3	57	54	9	9	9	0	57	54
1	1	1	33	54	441	6	9	9	33	54
2	1	0	40	54	196	8	9	1	40	54
3	4	3	57	54	9	9	9	0	57	54
4	0	0	25	54	841	4	9	25	25	54

fuzz_partial_ratio	fuzz_partial_token_set_ratio	fuzz_partial_token_sort_ratio	fuzz_token_set_ratio	fuzz_token_sort_ratio	wmd	cosine_distance
65	100	68	75	66	1.756	0.309
48	100	47	52	47	3.716	0.681
45	100	38	39	35	3.965	0.731
65	100	68	75	66	1.756	0.309
48	38	38	34	34	3.481	0.641

Fig 4.5: Extracted Features

- Feeding the data to trained model: The data with its features are then feed into our random forest model. We read the prediction from each tree.
- Final Prediction: Finally, the mode of the output of each tree is taken and it is our final prediction.

4.2.3 Tools Used

4.2.3.1 Analysis and Design Tools

Several tools has been used to create the diagrams. Visual Paradigm was used to create Activity diagram, Sequence diagram and Use Case diagram.

4.2.3.2 Implementation Tools

- Front End:- HTML, CSS, Bootstrap
- Back End:- SQLITE3, Python, Python Framework(Django)

4.3 Testing

4.3.1 Unit Testing

Unit test was done by feeding the system with several test cases. We performed unit test by giving two different inputs as an input and observed our output. System worked fine. However, the system gave some wrong predictions as well.

4.3.2 System Testing

The system included two parts: One for the web application and the other for making the prediction. We tested our system for the both. First, we created our prediction model. On next phase, we developed a simple web application interface and integrated them together to form a complete system.

Input Question (Test Cases)	**Output**	**Expected Output**	**Result**
Question: Can anybody suggest from where one should start Machine Learning?	Duplicate	Duplicate	Duplicate to: Where should I start to learn Machine Learning?
Question: Do you think that aliens exist?	Not Duplicate	Not Duplicate	Added to the database
Question: Where can I find a good place for digital marketing in Kathmandu?	Not Duplicate	Duplicate	Added to the database

Fig 4.6: System testing table

4.4 Accuracy Measurement

While training, our algorithm uses only few fraction of the total training data at once and repeat the same process with different fraction of sub samples. Since we have provides the number of folds to be five, the whole training data will be divided into five subsamples. Then on training phase, we pick one fold as a test data set and the remaining four folds as a training data set. This process is repeated, each time taking a non-repeating fold as testing and remaining as training set.

Total data = 8000

Number of Folds = 5

Each subsample size = $\frac{Total\ Data\ size}{Nuber\ of\ folds} = \frac{8000}{5} = 1600$

Iteration	Total Predicted	Correct Prediction	Accuracy
1	1600	1148	71.75
2	1600	1101	68.8125
3	1600	1106	69.125
4	1600	1107	69.1875
5	1600	1101	68.8125
			Mean Accuracy = 69.5375

Fig 4.7: Accuracy Measurement

CHAPTER FIVE
CONCLUSION AND RECOMMENDATION

5.1 Conclusion

The major intention of this project is to develop a system where different users can ask numbers of questions. However, the system would detect if any question were repeated. Doing so, the system would decrease the size of its database in the case of large numbers of users and every user asking so many question. In addition, the answerers do not have to keep answering the same question repeatedly.

Since the start of the project, we did research on several methodologies on how can we analyze the similarity between the two sentences, or to be more concise, two questions. We planned to go with the simple machine learning approach that included some basics of NLP. As per our plan, we developed a system that could operate as we have expected.

However, we did not get the accuracy as much as we have wanted. Our system predicted correctly at some places but in some cases failed to detect the duplication even when two questions meant exactly the same but composed of different words. Other major issue is that, our system takes lots of time to process the input and give the prediction. This obviously has downgraded the overall appearance of the project since time is one major factor.

5.2 Recommendation

This application can always be improved no matter how hard we have tried to make it as we have expected. In our application, we have only allowed a user to ask the question without any authorization, and any other user can answer it. To improve further, we can add an authentication system for the user.

In addition to this, we have only focused on the question being asked but not the answer. What we can do is, we can implement the same duplication detection techniques for the

answers in case the answerer has copied the answer from some other source and taken the credit by him/herself.

Furthermore, we can use some classification algorithm to classify the questions into different sections and user being able to visit their desired sections. This makes the system more modular and manageable.

5.3 Future Works

The foundation established by this project can be extended to several other platforms as well. In our case, we have focused on a platform where user can ask a question. However, it can be further implemented on other tasks where the duplication is prohibited. Some of them can be listed as:

- **Plagiarism**: To detect how similar the documents are. Instead of just a sentence level analysis, we can implement it on a document to make the prediction.
- **Chat bots:** In case of the automated chat bots or robots, they can save their response in case they are asked the similar questions.
- **Community Discussion Forum:** When a person open a topic for discussion, then there would be no point establishing a totally new topic if the same topic has been discussed in the past. So, in such case DQD can be implemented and integrated into the system.

REFERENCES

[1] Adversarial Domain Adaptation for Duplicate Question Detection Darsh J Shah,Tao Lei,Alessandro Moschitti, Salvatore Romeo,Preslav Nakov

[2] https://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/

[3] https://www.kaggle.com/quora/question-pairs-dataset

[4] Identifying Duplicate Questions: A Machine Learning Case Study: Siri Surabathula

[5] https://homes.cs.washington.edu/~tqchen/pdf/BoostedTree.pdf

[6]https://scikitlearn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression

[7] https://github.com/RahulVamusani/Quora-Question-Pair-Similarity

[8] https://dataaspirant.com/2017/05/22/random-forest-algorithm-machine-learing/

[9] Siamese Neural Networks with Random Forest for detecting duplicate question pairs Ameya Godbole ,Aman Dalmia, Sunil Kumar Sahu, Department of Electronics and Communication Engineering Department of Computer Science and Engineering Indian Institute of Technology Guwahati, Assam, India

[10] https://code.google.com/archive/p/word2vec/

BIBLIOGRAPHY

- Duplicate Question Pair Detection with Deep Learning: Travis Addair Department of Computer Science Stanford University

- Sentence Similarity Based on Semantic Nets and Corpus Statistics: Yuhua Li, David McLean, Zuhair A. Bandar, James D. O'Shea, and Keeley Crockett

- Quora Question Pairs: Zihan Chen, Hongbo Zhang, Xiaoji Zhang, Leqi Zhao University of Waterloo

- Real and Misflagged Duplicate Question Detection in Community Question-Answering: thesis presented by Doris Hoogeveen

- Identifying Quora question pairs having the same intent: Shashi Shankar, Aniket Shenoy

- How Well Sentence Embeddings Capture Meaning, Lyndon White, Robert Togneri, Wei Liu

APPENDICES

Appendix-1

// RandomForest.py

```
import pickle
from random import seed
from random import randrange
from csv import reader
from math import sqrt
# Load a CSV file
def load_csv(filename):
    dataset = list()
    with open(filename, 'r') as file:
        csv_reader = reader(file)
        for row in csv_reader:
            if not row:
                continue
            dataset.append(row)
    return dataset
# Split a dataset into k folds
def cross_validation_split(dataset, n_folds):
    dataset_split = list()
    dataset_copy = list(dataset)
    fold_size = int(len(dataset) / n_folds)
    for i in range(n_folds):
        fold = list()
        while len(fold) < fold_size:
            index = randrange(len(dataset_copy))
            fold.append(dataset_copy.pop(index))
        dataset_split.append(fold)
    return dataset_split
```

```
# Calculate accuracy percentage
def accuracy_metric(actual, predicted):
    correct = 0
    for i in range(len(actual)):
        if actual[i] == predicted[i]:
            correct += 1
    with open('correct.txt', 'w') as f:
        for a, p in zip(actual, predicted):
            f.write("actual:            " + str(a))
            f.write("Predicted:      " + str(p))
            f.write("\n")
            f.write("Total Correct Predicted:        " + str(correct))
        return correct / float(len(actual)) * 100.0

# Evaluate an algorithm using a cross validation split
def evaluate_algorithm(dataset, algorithm, n_folds, *args):
    folds = cross_validation_split(dataset, n_folds)
    scores = list()
    for fold in folds:
        train_set = list(folds)
        train_set.remove(fold)
        train_set = sum(train_set, [])
        test_set = list()
        for row in fold:
            row_copy = list(row)
            test_set.append(row_copy)
            row_copy[-1] = None
        predicted = algorithm(train_set, test_set, *args)
        actual = [row[-1] for row in fold]
```

```
def evaluate_algorithm(dataset, algorithm, n_folds, *args):
    folds = cross_validation_split(dataset, n_folds)
    scores = list()
    for fold in folds:
        train_set = list(folds)
        train_set.remove(fold)
        train_set = sum(train_set, [])
        test_set = list()
        for row in fold:
            row_copy = list(row)
            test_set.append(row_copy)
            row_copy[-1] = None
        predicted = algorithm(train_set, test_set, *args)
        actual = [row[-1] for row in fold]
        accuracy = accuracy_metric(actual, predicted)
        scores.append(accuracy)
    return scores
# Split a dataset based on an attribute and an attribute value
def test_split(index, value, dataset):
    left, right = list(), list()
    for row in dataset:
        if row[index] < value:
            left.append(row)
        else:
            right.append(row)
    return left, right
```

```
def gini_index(groups, classes):
    # count all samples at split point
    n_instances = float(sum([len(group) for group in groups]))
    # sum weighted Gini index for each group
    gini = 0.0
    for group in groups:
        size = float(len(group))
        # avoid divide by zero
        if size == 0:
            continue
        score = 0.0
        # score the group based on the score for each class
        for class_val in classes:
            p = [row[-1] for row in group].count(class_val) / size
            score += p * p
        # weight the group score by its relative size
        gini += (1.0 - score) * (size / n_instances)
    return gini
# Create a terminal node value
def to_terminal(group):
    outcomes = [row[-1] for row in group]
    return max(set(outcomes), key=outcomes.count)
# Build a decision tree
def build_tree(train, max_depth, min_size, n_features):
    root = get_split(train, n_features)
    split(root, max_depth, min_size, n_features, 1)
    return root
```

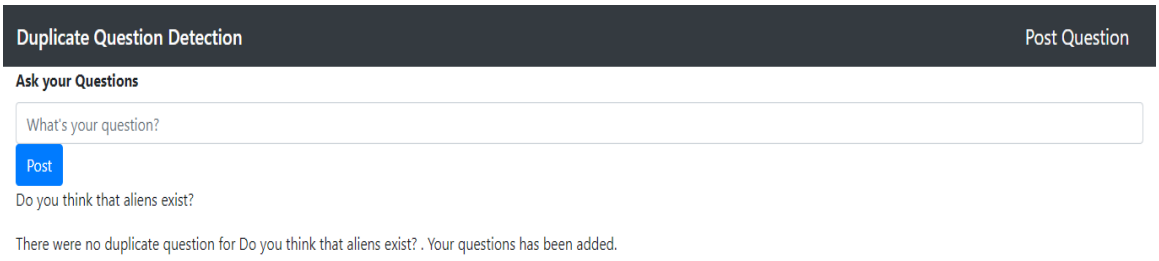
```
# Select the best split point for a dataset
def get_split(dataset, n_features):
    gini_values = dict()
    class_values = list(set(row[-1] for row in dataset))
    b_index, b_value, b_score, b_groups = 999, 999, 999, None
    features = list()
    while len(features) < n_features:
        index = randrange(len(dataset[0]) - 1)
        if index not in features:
            features.append(index)
    for index in features:
        for row in dataset:
            groups = test_split(index, row[index], dataset)
            gini = gini_index(groups, class_values)
            gini_values[dataset[0][index]] = gini
            if gini < b_score:
                b_index, b_value, b_score, b_groups = index, row[index], gini, groups
    return {'index': b_index, 'value': b_value, 'groups': b_groups}
def predict(node, row):
    if row[node['index']] < node['value']:
        if isinstance(node['left'], dict):
            return predict(node['left'], row)
        else:
            return node['left']
    else:
        if isinstance(node['right'], dict):
            return predict(node['right'], row)
        else:
            return node['right']
```

```
def split(node, max_depth, min_size, n_features, depth):
    left, right = node['groups']
    del (node['groups'])
    # check for a no split
    if not left or not right:
        node['left'] = node['right'] = to_terminal(left + right)
        return
    # check for max depth
    if depth >= max_depth:
        node['left'], node['right'] = to_terminal(left), to_terminal(right)
        return
    # process left child
    if len(left) <= min_size:
        node['left'] = to_terminal(left)
    else:
        node['left'] = get_split(left, n_features)
        split(node['left'], max_depth, min_size, n_features, depth + 1)
    # process right child
    if len(right) <= min_size:
        node['right'] = to_terminal(right)
    else:
        node['right'] = get_split(right, n_features)
        split(node['right'], max_depth, min_size, n_features, depth + 1)
def subsample(dataset, ratio):
    sample = list()
    n_sample = round(len(dataset) * ratio)
    while len(sample) < n_sample:
        index = randrange(len(dataset))
        sample.append(dataset[index])
    return sample
```

```
# Make a prediction with a list of bagged trees
def bagging_predict(trees, row):
    predictions = [predict(tree, row) for tree in trees]
    return max(set(predictions), key=predictions.count)
 # Random Forest Algorithm
def random_forest(train, test, max_depth, min_size, sample_size, n_trees, n_features):
    trees = list()
    for i in range(n_trees):
        sample = subsample(train, sample_size)
        tree = build_tree(sample, max_depth, min_size, n_features)
        trees.append(tree)
    with open('trees_40000.pickle', 'wb') as f:
        pickle.dump(trees, f)
    predictions = [bagging_predict(trees, row) for row in test]
    return (predictions)
seed(2)
filename = 'final_training_20k.csv'
dataset = load_csv(filename)
for i in range(len(dataset)):
    dataset[i].append(dataset[i][0])
    dataset[i].remove(dataset[i][0])
print(dataset[0])
n_folds = 5
max_depth = 5
min_size = 1
sample_size = 1.0
n_features = int(sqrt(len(dataset[0]) - 1))
```

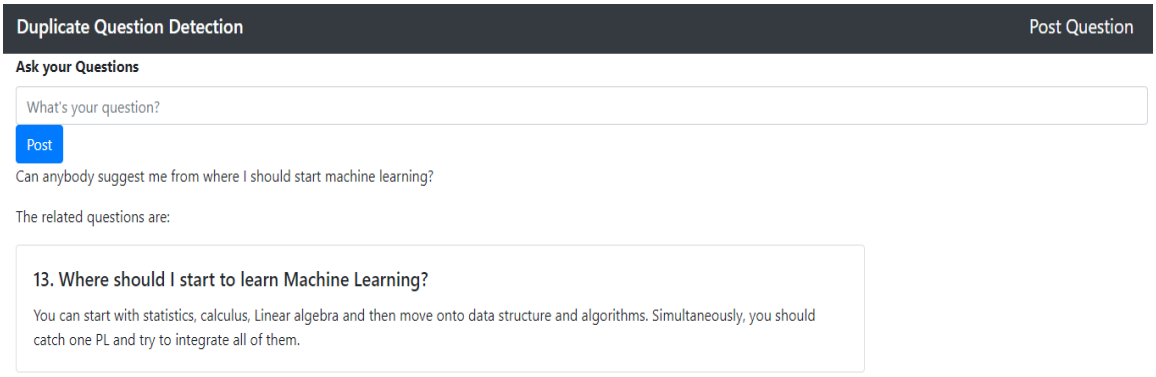
```
for n_trees in [15]:
    scores = evaluate_algorithm(dataset, random_forest, n_folds, max_depth, min_size, sample_size, n_trees, n_features)
    print('Trees: %d' % n_trees)
    print('Scores: %s' % scores)
    print('Mean Accuracy: %.3f%%' % (sum(scores) / float(len(scores))))
    with open('accuracy.txt', 'a') as f:
        for s in scores:
            f.write(str(s))
        f.write(str(sum(scores) / float(len(scores))))
```

Appendix-2



0	How can I increase the numbers of instagram followers?	Do you think that aliens exist?	0
1	How can I be good at programming?	Do you think that aliens exist?	0
2	What are the top 5 comedy movies?	Do you think that aliens exist?	0
3	What is the best/most memorable thing you've ever eaten and why?	Do you think that aliens exist?	0
4	What are some outfit ideas to wear to a party?	Do you think that aliens exist?	0
5	How did you learn to use a computer?	Do you think that aliens exist?	0
6	Which is the best digital marketing institution in Kathmandu??	Do you think that aliens exist?	0
7	Why should I major in math?	Do you think that aliens exist?	0
8	How hard is it to play the ukulele?	Do you think that aliens exist?	0
9	Where can I find a good place for digital marketing in kathmandu?	Do you think that aliens exist?	0
10	Is majoring in math a good choice?	Do you think that aliens exist?	0
11	Where should I start to learn Machine Learning?	Do you think that aliens exist?	0

Fig 6.1: Detected as not duplicate



0	How can I increase the numbers of instagram followers?	Can anybody suggest me from where I should start machine learning?	0
1	How can I be good at programming?	Can anybody suggest me from where I should start machine learning?	0
2	What are the top 5 comedy movies?	Can anybody suggest me from where I should start machine learning?	0
3	What is the best/most memorable thing you've ever eaten and why?	Can anybody suggest me from where I should start machine learning?	0
4	What are some outfit ideas to wear to a party?	Can anybody suggest me from where I should start machine learning?	0
5	How did you learn to use a computer?	Can anybody suggest me from where I should start machine learning?	0
6	Which is the best digital marketing institution in Kathmandu??	Can anybody suggest me from where I should start machine learning?	0
7	Why should I major in math?	Can anybody suggest me from where I should start machine learning?	0
8	How hard is it to play the ukulele?	Can anybody suggest me from where I should start machine learning?	0
9	Where can I find a good place for digital marketing in kathmandu?	Can anybody suggest me from where I should start machine learning?	0
10	Is majoring in math a good choice?	Can anybody suggest me from where I should start machine learning?	0
11	Where should I start to learn Machine Learning?	Can anybody suggest me from where I should start machine learning?	1

Fig 6.2: Detected as duplicate