TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

KATHMANDU ENGINEERING COLLEGE

DEPARTMENT OF COMPUTER ENGINEERING

Major Project Report

On

QUESTION AND ANSWERING SYSTEM

[Code no: CT 755]

Submitted By:

Anurag Shukla – KAT075BCT013

Ayush Shrestha – KAT075BCT018

Bharat Adhikari – KAT075BCT019

Diwash Adhikari – KAT075BCT030

Kathmandu, Nepal

Baisakh, 2080

TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

KATHMANDU ENGINEERING COLLEGE

DEPARTMENT OF COMPUTER ENGINEERING

QUESTION AND ANSWERING SYSTEM

PROJECT REPORT SUBMITTED TO THE

DEPARTMENT OF COMPUTER ENGINEERING

IN PARTIAL FULFILLMENT OF

REQUIREMENTS IN BACHELOR OF ENGINEERING

[Code no: CT 755]

Submitted By:

Anurag Shukla – KAT075BCT013

Ayush Shrestha – KAT075BCT018

Bharat Adhikari – KAT075BCT019

Diwash Adhikari – KAT075BCT030

Kathmandu, Nepal

Baisakh, 2080

TRIBHUWAN UNIVERSITY
INSTITUTE OF ENGINEERING

KATHMANDU ENGINEERING COLLEGE
DEPARTMENT OF COMPUTER ENGINEERING

CERTIFICATE

The undersigned certify that they have read and recommended to the Department of Computer Engineering, a major project work entitled "Question and Answering System" submitted by Anurag Shukla – 39362, Ayush Shrestha – 39365, Bharat Adhikari – 39366 and Diwash Adhikari – 39377 in partial fulfilment of the requirements for the degree of Bachelor of Engineering.

Er. Sharad Neupane (Project Supervisor) Department of Computer Engineering Kathmandu Engineering College	Dr. Nand Bikram Adhikari (External Examiner) Department of Computer Engineering Pulchowk Engineering Campus
Er. Sharad Chandra Joshi (Project Coordinator) Department of Computer Engineering Kathmandu Engineering College	Er. Sudeep Shakya (Head of Department) Department of Computer Engineering Kathmandu Engineering College

ACKNOWLEDGEMENT

We would like to express our most sincere thanks and gratitude to **Institute of Engineering** for the inclusion of major project as an academic requirement. We would also like to acknowledge with much appreciation the role of **Department of Computer Engineering, Kathmandu Engineering College** in providing necessary support and guidance to augment our individual skills and spirit of team work during the project.

It is our privilege to express our greatest gratitude to project supervisor **Er. Sharad Neupane** and the Head of Department of Computer Engineering, **Er. Sudeep Shakya** for their praiseworthy efforts. At the very beginning, we were confused and little nervous regarding the project. But, with their proper guidance we carried our project forward.

Our thankful regard goes to all our faculty teachers who gave numerous illustrations to help us put our knowledge into practicality. It is their methodology and a rather different approach to teaching that has helped us develop a new perspective to cope with various problems during software development with simplicity and creativity. We also owe thanks to our friends who suggested various ideas.

We have made a diligent attempt to make this report well documented and in accordance to the standard guidance. However, any constructive comments and suggestions are highly welcome and will be greatly appreciated.

ABSTRACT

Question and Answering System is a web application developed using Django that features two machine learning models called semantic similarity model and an extractive QA model. The semantic similarity model trained using GRU RNN is used for checking the database for similar questions and also for preventing duplication. This model is also used for searching, providing a more efficient and effective way to find relevant questions and answers. And, the system's extractive QA model trained with Simple Transformer can extract answers from a given context or passage, providing users with the most relevant response possible which is accessible through a section called “Extractive QA” in the app. Our system also consists of traditional question-and-answer section for new inquiries where users can answer, reply, and vote on responses, creating a collaborative and dynamic community once they log into the system. The Question and Answering System has Admin who has all the rights to the system. They can simply manage the posts that user created and also can manage users. This allows the user to find high quality answers within a short period of time which in turn would result in an improved user experience for writers and readers. The system uses BERT word embedding as input to the deep learning model.

Keywords: *Django, semantic similarity, QA model, Simple Transformer, BERT, GRU*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATION

Acronym	Full Form
AI	Artificial Intelligence
ANN	Artificial Neural Network
BERT	Bidirectional Encoder Representations from Transformers
CNN	Convolution Neural Network
CSS	Cascaded Style Sheet
GloVe	Global Vector
GPU	Graphics Processing Unit
GRU	Gated recurrent Unit
GUI	Graphical User Interface
HTML	Hypertext Markup Language
LSTM	Long-term Short-term Memory
ML	Machine Learning
NLP	Natural Language Processing
QA	Question and Answer
RTE	Recognizing Textual Entailment
RNN	Recurrent Neural Network
SDLC	Software Development Life Cycle
SNLI	Stanford Natural Language Inference
SVM	Support Vector Machine
SQL	Structured Query Language
SQuAD	Stanford Question Answering Dataset
TF-IDF	Term Frequency-Inverse Document Frequency

CHAPTER 1: INTRODUCTION

1.1 Background Theory

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 failed to keep up the content value of their topics and answers due to a lot of junk information posted; thus, their user base declined.

Quora is a social media website where questions are posted by users and answered by experts who provide quality insights. Other users can cooperate by editing questions and suggesting more accurate answers to the submitted questions. According to statistics provided by the Director of Product Management at Quora on 17 September 2018 [4], Quora receives 300 million unique visitors every month, which raises the problem of different users asking similar questions with same intent but in different words. Multiple questions with similar wording can cause readers to spend more time to find the best answer, and make writers answer multiple versions of the same question. Therefore, Quora has an important principle for having a single question thread for logically different questions. For example, questions like ‘‘How can I be a good photographer?’’ and ‘‘What should I do to be a great photographer?’’ are identical because both have the same meaning and should be answered only once. Some questions, like ‘‘How old are you?’’ and ‘‘What is your age?’’ do not have same wording. However, the context remains the same. Therefore, such questions are also considered duplicate. It can be an overhead to have different pages for such questions. Thus, identifying the duplicate questions at Quora and merging them makes knowledge sharing more efficient and effective in many ways. This way, the seekers can get answers to all the questions on a single thread and writers do not need to write the same answer on different locations for the same question. They can get larger number of readers than if the readers are divided in several threads [2].

The purpose of this system is to build a web application that allows user to ask questions and that questions are answered by other users who has quality knowledge to those questions. Other users can also recommend other better answers to those questions by adding comment. The system can identify duplicate questions and recommend previously asked similar questions.

Identification of duplicate questions is a crucial task in Natural Language Processing (NLP) with many applications such as Recognizing Textual Entailment (RTE) classifying text, retrieving information and detecting plagiarism and Paraphrase Recognition. It measures the degree of similarity between two interrogative fragments. If the fragments are semantically similar, they can get the same answer and are considered duplicate. The task of identification of duplicate questions can be a great challenge because the true meaning of sentence cannot be known with certainty due to the ambiguous language and synonymous expressions.

Natural Language Processing

Natural Language Processing (NLP) is a way for computers to analyze, understand, and derive meaning from human language in a smart and useful way. By utilizing NLP, developers can organize and structure knowledge to perform tasks such as automatic summarization, translation, named entity recognition, relationship extraction, sentiment analysis, speech recognition and topic segmentation.

1.2 Problem Statement

"Time is what we want most, but what we use worst" – by William Penn. This statement shows the importance of time and it applies to all of us. But, saving time by utilizing it properly is the most difficult task.

We humans are curious, exploring and ever-learning creature and always ask doubts and queries to others in order to learn more information about something. Nowadays people ask question online. The major problems that most people face these days are:

- Time-consuming process of finding good answers to online questions due to duplication and inefficient keyword-based searching.
- Answering context-based questions related to specific books or paragraphs, which can be difficult for external users and may cause delays in responses.

1.3 Objectives

The objective of the project is to develop an application that:

- Create a web app that uses advanced machine learning models to prevent question duplication and enable semantic searching.
- Provide relevant context-based answers to user's questions.

1.4 Scope of the project

This project is primarily designed to be used in question answering systems where there is a need to identify and eliminate duplicate questions by finding similar questions. The goal of this project is to simplify the process of finding good answers by ensuring that they are not distributed across multiple duplicate questions. By using this project, question answering systems can streamline their operations and improve the quality of the responses provided to users. This makes it easier for users to find the information they are looking for and reduces the likelihood of confusion or frustration due to duplicate or inconsistent responses.

1.5 Applications

Our system can be applied to various use cases like:

- Our system can be tuned to detect plagiarism because it can effectively capture the phrase changes, which is impossible to detect with traditional string-matching systems.
- Information Management systems can use it for grouping of various kinds of data based on semantic similarity.
- Advertising agencies can also use the similarity detection in user interests profiling to show personalized contents.
- Can be integrated in a Chatbot system.

CHAPTER 2: LITERATURE REVIEW

There are some papers which were taken references for making this project. From those papers some conclusions were made and from which we made those decisions what features to take or what features to be used. Some papers were similar to the project to be made while some had the algorithms which were to be implemented, while some had the features which were to be implemented, the others had idea about pre-processing the data and how to clean the data before processing it through the algorithms.

The 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 words in their experiments.

The method proposed in paper [6] 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. 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). The next step was determining the distance measure 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.

In the literature [7], authors have used word ordering and word alignment using a Long-Short-Term-Memory (LSTM) recurrent neural network, 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. Some of these models are Bag of words model, RNN with GRU and LSTM cell, LSTM with attention, Decomposable attention model. 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 [8] achieves better performance. This paper [8] 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. The paper [9] presents SQuAD, a large-scale dataset consisting of 100,000+ questions and answers, designed for training and evaluating machine comprehension of text systems. The dataset was created using crowdsourcing and covers a diverse range of topics and question types. The authors also propose a baseline model for the dataset, which achieves state-of-the-art performance at the time of publication.

CHAPTER 3: METHODOL88OGY

3.1 Process Model

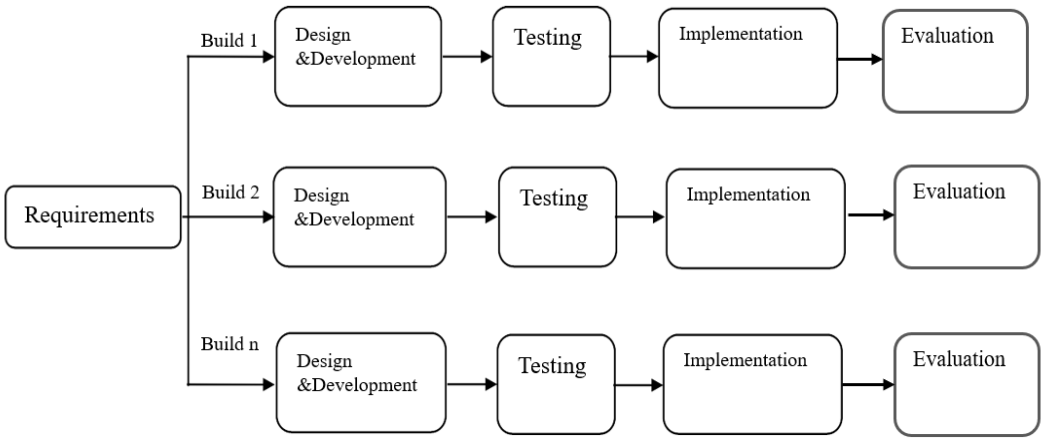

Figure 3.1: Iterative Model

We are going to use iterative model for the development of our project. In this process model, once the initial planning is complete, a handful of phases are repeated again and again, with the completion of each cycle improving the software. The iterative model allows us to access previous cycles which can be reviewed and changed if necessary. The iterative model is cyclical. This SDLC fits our project because we have to review our ML models and change them if higher accuracy is obtained in following ML models**.**

Step 1: The developers sit together to identify software requirement.

Step 2: Design team establishes technical requirements such as language, services etc.
Step 3: The developers start writing their first version of code.

Step 4: Developers test module for potential bugs. If any problems are found, then we go back and work on the second iteration.

Step 5: Once all steps are followed for the iterations, the team will finally evaluate the entire project.

3.2 Block Diagram

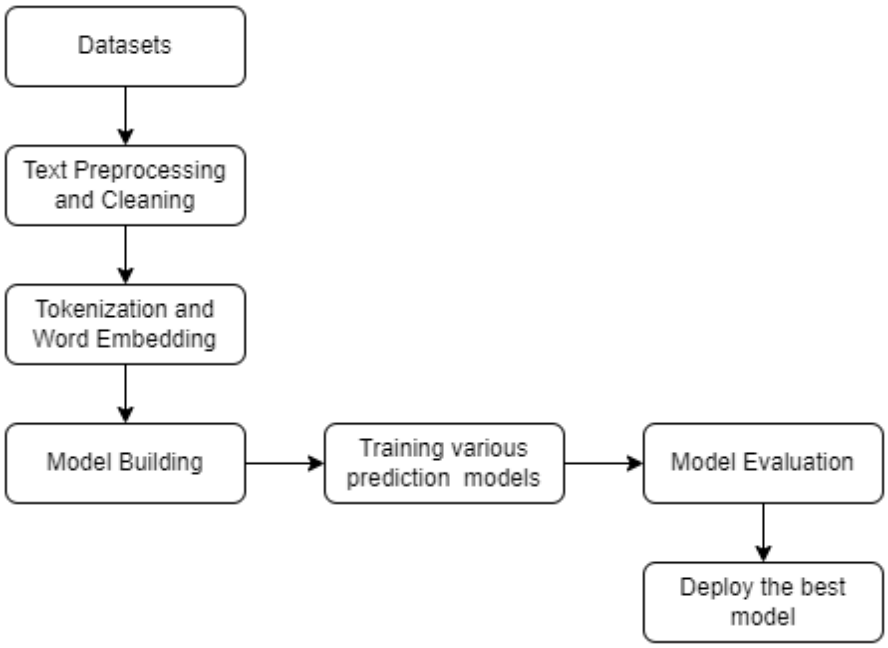

Figure 3.2: Block Diagram

The block diagram consists of several different components. They are:

i) Datasets:

The first step to sole any Machine Learning problem is to collect relevant datasets. In this project we have used two different datasets called SNLI Corpus and SQuAD dataset.

SNLI Corpus

a) Attributes:

- Size: Over 500,000 sentence pairs.
- Source: Derived from image captions and their corresponding texts from the Flickr30k dataset and was developed by the Stanford Natural Language Processing Group.
- Format: Each sentence pair is labeled by human annotators based on their relationship to each other.
- Task: Used for NLP text similarity detection.

b) Column Names:

- Sentence 1: The first text sentence in the pair

- Sentence 2: The second text sentence in the pair
- Label: The label assigned to the sentence pair by human annotators (entailment, contradiction, or neutral) but we used only data points containing entailment and contradiction labels for our project where label 0 means different or opposite and label of 1 represents semantically similar.

sentence1	sentence2	label
a wet dog chasing a white ball	A dog sits sadly.	0.0
A large gray statue in front of a big building.	A statue is front of the building.	1.0
A man with a Mohawk stands in a group of people.	A woman with a Mohawk stands alone.	0.0
A child stand with an elderly women at the 96t...	There is a subway station on 96th Street.	1.0
Two men operating a machine gun, both are wear...	The men use hand guns while wearing a business...	0.0

Figure 3.3: SNLI Corpus sample

SQuAD Dataset

a) Attributes:

- Size: Over 100,000 question-answer pairs.
- Source: Derived from over 500 Wikipedia articles.
- Format: Each question-answer pair is associated with a specific paragraph.
- Task: Used for extracting answer from paragraphs.

b) Key Names:

- ID: Unique identifier for each question-answer pair.
- Context: Paragraph of Wikipedia article, the question-answer is associated with.
- Question: The text of the question asked about the context.
- Answer: The text of the answer to the question.
- Start Position: The character position in the context where the answer begins.
- End Position: The character position in the context where the answer ends.
- Is Impossible: True if there is no answer in the context else False.

```
{
    "context": "the term matter is used throughout physics in a bewildering
variety of contexts for example one refers to condensed matter physics
elementary matter partonic matter dark matter antimatter strange matter and
nuclear matter in discussions of matter and antimatter normal matter has been
referred to by alfven as koinomatter gk common matter it is fair to say that
in physics there is no broad consensus as to a general definition of matter
and the term matter usually is used in conjunction with a specifying
modifier",
    "qas": [
        {
            "id": 390954,
            "is_impossible": true,
            "question": "what field of study has a variety of unusual
contexts?",
            "answers": []
        }
    ]
}
```


Figure 3.4: SQuAD dataset sample with no answer


```
{
    "context": "beyonc\u00e9 giselle knowles-carter (/bi\u02d0\u02c8j\u0252nse
\u026a/ bee-yon-say) (born september 4, 1981) is an american singer,
songwriter, record producer and actress. born and raised in houston, texas,
she performed in various singing and dancing competitions as a child, and rose
to fame in the late 1990s as lead singer of r&b girl-group destiny's child.
managed by her father, mathew knowles, the group became one of the world's
best-selling girl groups of all time. their hiatus saw the release of beyonc
\u00e9's debut album, dangerously in love (2003), which established her as a
solo artist worldwide, earned five grammy awards and featured the billboard
hot 100 number-one singles \"crazy in love\" and \"baby boy\".",
    "qas": [
        {
            "id": 6,
            "is_impossible": false,
            "question": "what areas did beyonce compete in when she was
growing up?",
            "answers": [
                {
                    "text": "singing and dancing",
                    "answer_start": 207
                }
            ]
        }
    ]
}
```


Figure 3.5: SQuAD dataset sample with answer

ii) Text Preprocessing and Cleaning

The aim of this preprocessing step is to clean the messy data so that it is ready for further processing. In NLP, preprocessing consists of several steps such as removing noisy data, null values and converting the text into lowercase. We preprocessed the data using pandas and numpy.

iii) Tokenization and Word Embedding

Tokenization is cutting input data into meaningful parts that can be embedded into a vector space. Tokenization converts a text into a list of integers. Embedding layers map tokens to word vectors (sequence of numbers) called word embedding. Embedding contain semantic information about the word. We used BERT Tokenizer for this step.

iv) Model Building

In this step, we choose the ML models that will be used to train the datasets. In our project, we used Tensorflow's Functional API to build the text semantic similarity model and Simple Transformers to build the extractive QN model.

The hyperparameters used for Text Semantic Similarity model are as follows:

- n_units : 32
- dropout rate : 30
- max_sequence_length : 128
- BERT model selection : bert-base-uncased

The hyperparameters used for Model Building of Extractive QA model are:

- model_name : bert-base-uncased
- max_query _length : 50
- max_answer_length : 20

v) Training the model

The next step after initializing the model is to begin training the model on the train data. The model then learnt all parameters and weights from the train data. We used various hyperparameters to train the model and reviewed the results and re-trained the model and used those hyperparameters which increases accuracy and reduce the loss of the model.

The hyperparameters used for model training of Text Semantic Similarity Model are:

- learning_rate : 1e-4 for bert-freeze training and 1e-5 for fine tuning
- batch_size : 32

- epochs : 3 for bert-freeze training and 3 for fine tuning
- optimizer : adam
- loss : categorical crossentropy

The hyperparameters used for model training of Extractive QA Model are:

- learning_rate : 2e-5
- train_batch_size : 124
- num_train_epochs : 3
- optimizer : adam
- loss : categorical crossentropy

vi) Model Evaluation

Then, we tested the model on a dataset that the model has not yet seen. This helped us to evaluate the performance of the model using different metrics like accuracy, precision, recall.

vii) Deployment

After evaluating the model, we deployed the model by adjusting the prediction threshold as per our need.

3.3 Related Theory

3.3.1 Machine Learning

Machine Learning is the subfield of computer science that gives computer the ability to learn without explicitly programmed. Machine Learning is also related to statistics and focuses on prediction. Machine Learning is broadly classified into three broad categories, depending on the nature of the learning feedback available to a learning system: Supervised Learning, Unsupervised Learning and Reinforcement Learning.

3.3.1.1 Supervised Learning

Supervised Learning is the machine learning task where labeled datasets are used to train the algorithms. Supervised Learning can be classified into two types of problems: Regression and Classification. Classification uses an algorithm to accurately assign test data into specific categories. It recognizes specific entities within the dataset and attempts to draw some conclusions on how those entities should be labeled or defined.

3.3.2 BERT Model

BERT is a method of pretraining language representations that uses multilayer bidirectional transformer encoders for language representations. We can either use these models to extract high quality language features from your text data, or can fine-tune these models on a specific downstream task (classification, entity recognition, question answering, etc.) with your own data to produce state of the art predictions. BERT developers created two main models:

- The BASE: Number of transformer blocks (L): 12, Hidden layer size (H): 768 and self-attention heads(A): 12
- The LARGE: Number of transformer blocks (L): 24, Hidden layer size (H): 1024 and self-attention heads(A): 16

In this project, we will be using BERT-BASE-model. From a very high-level perspective, BERT's architecture looks like this:

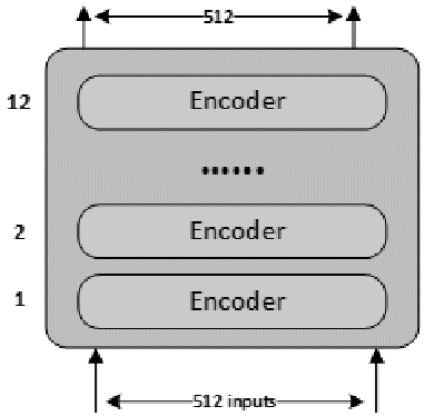

Figure 3.6: BERT Base Model

3.3.2.1 BERT embedding

We can use BERT to extract features, namely word and sentence embedding vectors, from text data. These embedding are useful for keyword/search expansion, semantic search and information retrieval. For example, if you want to match customer questions or searches against already answered questions or well documented searches, these representations will help you accurately retrieve results matching the customer's intent and contextual meaning, even if there's no keyword or phrase overlap. And perhaps more importantly, these vectors are used as high-quality feature inputs to downstream mod-

els. NLP models such as LSTMs or CNNs require inputs in the form of numerical vectors, and this typically means translating features like the vocabulary and parts of speech into numerical representations. In the past, words have been represented either as uniquely indexed values (one-hot encoding), or more helpfully as neural word embedding where vocabulary words are matched against the fixed-length feature embedding that result from models like Word2Vec. BERT offers an advantage over models like Word2Vec, because while each word has a fixed representation under Word2Vec regardless of the context within which the word appears, BERT produces word representations that are dynamically informed by the words around them.

3.3.2.2 Input-Output Format

The whole input to the BERT has to be given a single sequence. BERT uses special tokens [CLS] and [SEP] to understand input properly. [SEP] token has to be inserted at the end of a single input. BERT uses Wordpiece embeddings input for tokens. Along with token embeddings, BERT uses positional embeddings and segment embeddings for each token. Positional embeddings contain information about the position of tokens in sequence. Final Embeddings used by model architecture are the sum of token embedding, positional embedding as well as segment embedding. The final embeddings are then fed into the deep bidirectional layers to get output. The output of the BERT is the hidden state vector of pre-defined hidden size corresponding to each token in the input sequence. These hidden states from the last layer of the BERT are then used for various NLP tasks.

3.3.3 Pooling

Pooling is a process that progressively reduces the spatial size of the representation to reduce the number of parameters and computation in the network, and hence to also control over fitting No learning takes place on the pooling layer. Max pooling is a technique that extracts the maximum value from a set of values. In NLP, this is often applied to feature vectors representing individual words in a sentence, and the resulting maximum values are used to create a compressed representation of the sentence. Average pooling, on the other hand, takes the average of a set of values. This can also be used for feature extraction in NLP, where the average values can represent the overall "meaning" of a sentence or document. Both max pooling and average pooling can be used to

reduce the dimensionality of feature vectors, making them more computationally tractable in downstream tasks like classification

3.3.4 Adam Optimizer

Optimizers are algorithms used to change the attribute of neural network such as weights and learning rate in order to minimize loss function. Adam is also such kind of optimizer which is a combination of momentum and RMSprop and acts upon the gradient component by using v_t (the exponential moving average of gradients) and the learning rate component by dividing the learning rate α by square root of $\hat{v}_t$ (the exponential moving average of squared gradients).

We update the weights by using,

$$w_{t+1} = w_t - \frac{\alpha}{\sqrt{\hat{v}t}+\epsilon} * \hat{m}t \quad ...(i)$$

where the bias corrections are:

$$m_t = \beta_1 * m_{t-1} + (1-\beta_1)*[\frac{\delta L}{\delta wt}] \quad ...(ii)$$

$$v_t = \beta_2 * v_{t-1} + (1-\beta_2)*[\frac{\delta L}{\delta wt}]^2 \quad ...(iii)$$

and,

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t} \quad ...(iv)$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t} \quad ...(v)$$

3.4 Algorithm

3.4.1 Neural Network

An artificial neural network learning algorithm, or neural network, or just neural net, is a computational learning system that uses a network of functions to understand and translate a data input of one form into a desired output, usually in another form. The concept of the artificial neural network was inspired by human biology and the way neurons of the human brain function together to understand inputs from human senses. Neural networks are just one of many tools and approaches used in machine learning algorithms. The neural network itself may be used as a piece in many different machine learning algorithms to process complex data inputs into a space that computers can understand.

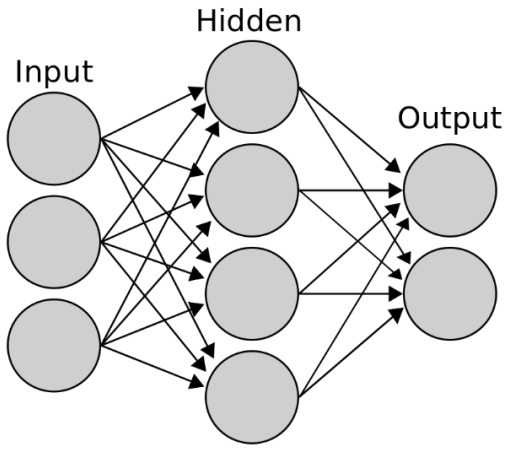

Figure 3.7: A simple feed-forward ANN

3.4.2 RNN

A recurrent neural network (RNN) is a special type of artificial neural network adapted to work for time series data or data that involves sequences. Ordinary feed forward neural networks are only meant for data points that are independent of each other. However, if we have data in a sequence such that one data point depends upon the previous data point, we need to modify the neural network to incorporate the dependencies between these data points. RNNs have the concept of "memory" that helps them store the states or information of previous inputs to generate the next output of the sequence. There are different variations of RNNs that are being applied practically in machine learning problems: LSTM and GRU.

3.4.3 GRU

It is a simplified version of LSTM RNN. There are only 2 sigmoid functions, that means there are only 2 gates, which are called reset gate and update gate. Also, it relies solely on a hidden state for memory transfer between recurrent units, so there is no separate cell state.

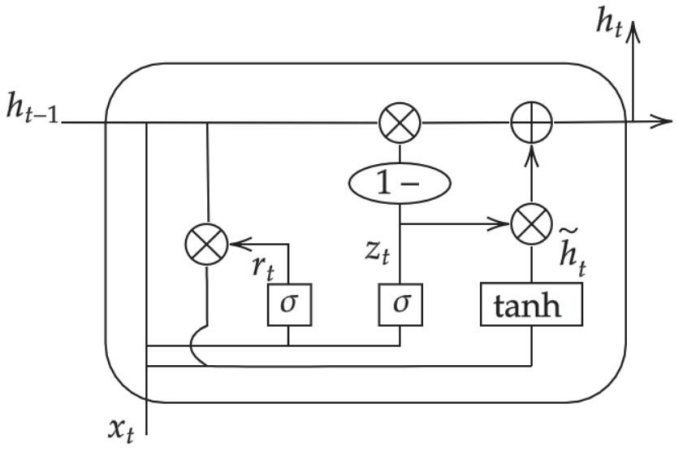

Figure 3.8: A single GRU unit

Reset Gate: Previous hidden state (h_{t-1}) and current input (x_t) are combined (multiplied by their respective weights and bias added) and passed through a reset gate. Since the sigmoid function ranges between 0 and 1, step one sets which values should be discarded (0), remembered (1), or partially retained (between 0 and 1). Step two resets the previous hidden state multiplying it with outputs from step one.

$r_t = \sigma (W_r x_t + U_r h_{t-1})$ ……..(i)

Update gate: The weights and biases used to scale these vectors are different, providing a different sigmoid output. So, after passing a combined vector through a sigmoid function, we subtract it from a vector containing all 1s and multiply it by the previous hidden state. That's one part of updating the hidden state with new information.

$z_t = \sigma (W_z x_t + U_z h_{t-1})$ ……..(ii)

Hidden state candidate: After resetting a previous hidden state in step two, the outputs are combined with new inputs (x_t), multiplying them by their respective weights and adding biases before passing through a tanh activation function. Then the hidden state candidate is multiplied by the results of an update gate and added to previously modified h_{t-1} to form the new hidden state h_t .

$\tilde{h}_t = \tanh(W_z x_t + U_z (r_t . h_{t-1})$ ……..(iii)

Here, "." is an element-wise multiplication.

Output: Lastly, the result from the above operations will be summed with our output from the Update gate in the previous step. This will give us our new and updated hidden state.

$h_t = (1-z_t)h_{t-1} + z_t \tilde{h}_t$ ……..(iv)

We can use this new hidden state as our output for that time step.

3.5 Flowchart

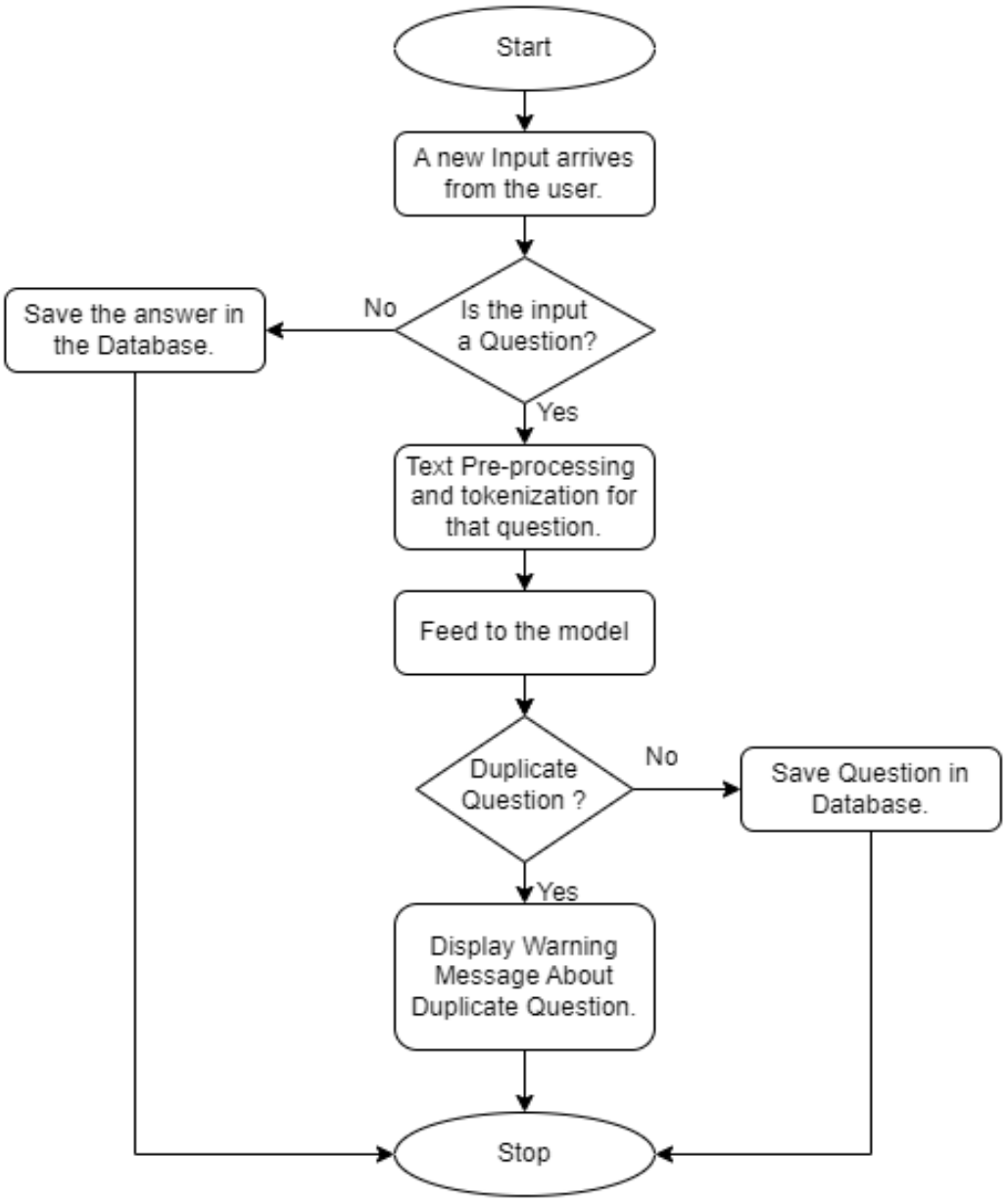

Figure 3.9: Flowchart

3.6 UML Diagrams

3.6.1 Use Case Diagram

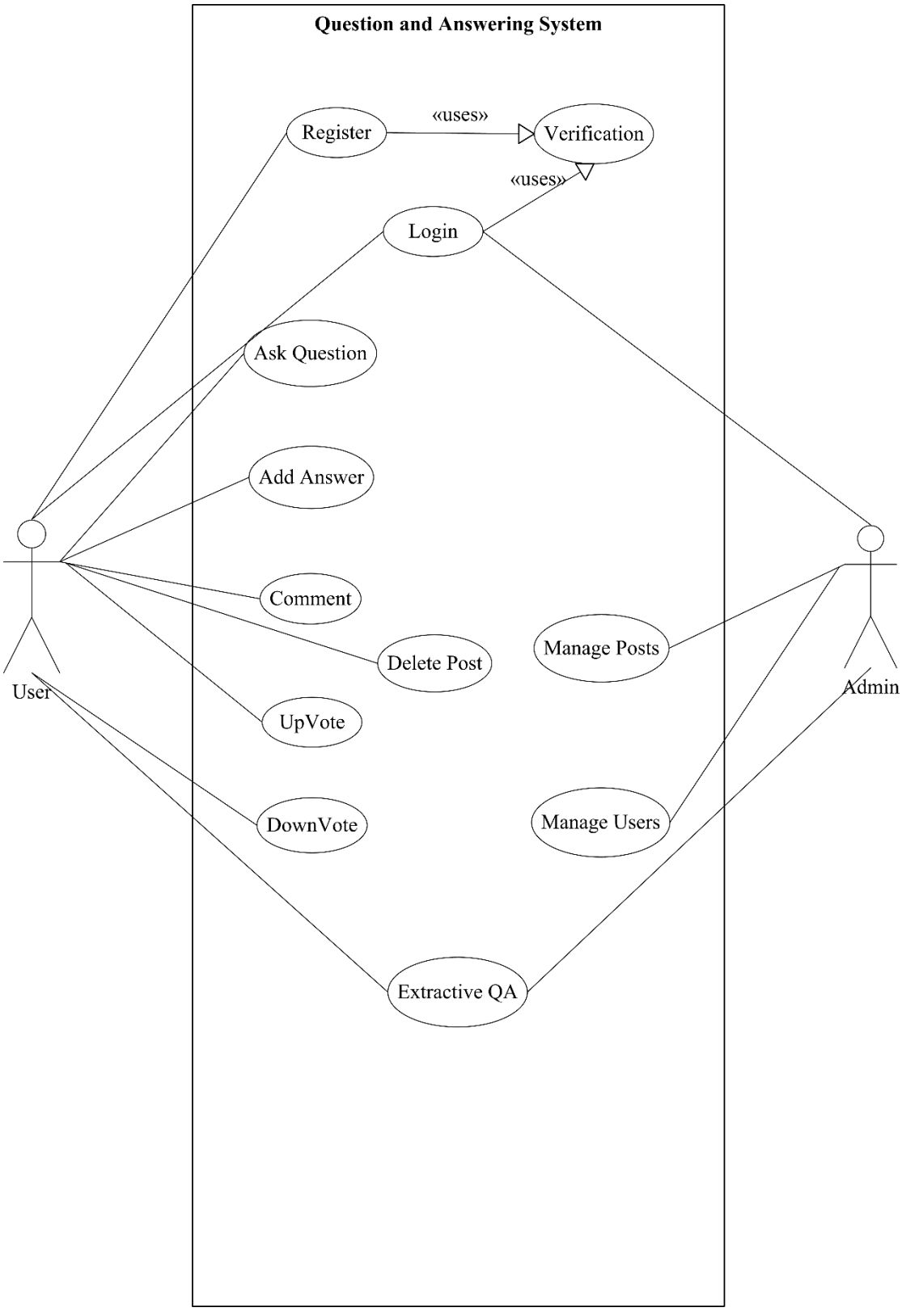

Figure 3.10: Use Case Diagram

3.6.2 Activity Diagram

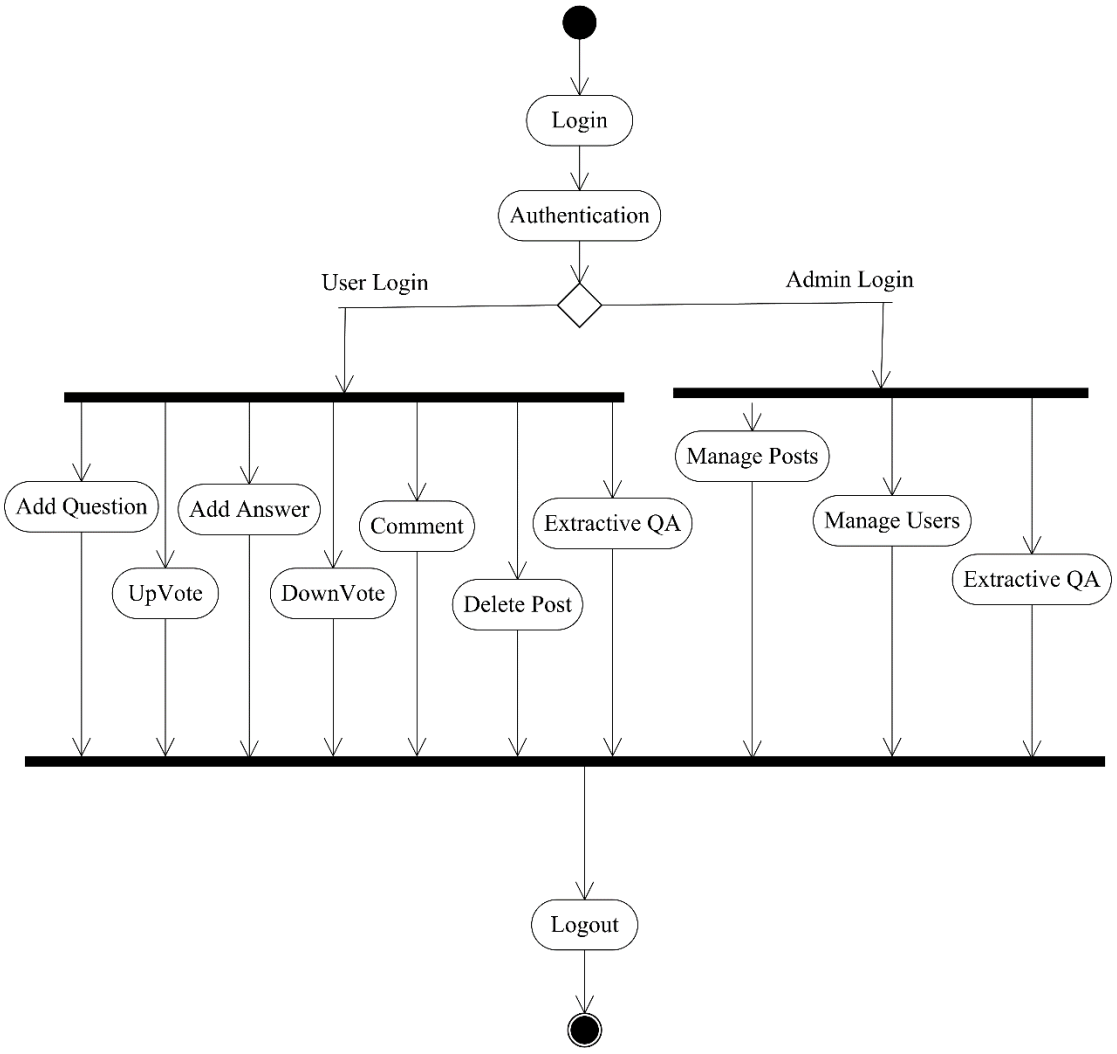

Figure 3.11: Activity Diagram

The above shown figure is the Activity Diagram of Question and Answering System. The system has activity like login, authentication, add question, add answer, comment, upvote, downvote, extractive QA, manage posts and manage users. The system has decision node to decide whether user logged in to system or the admin. The fork node is used to spilt the flow of control into multiple parallel flows whereas join node is used to join multiple parallel flow into single flow.

3.6.3 Data Flow Diagram

3.6.3.1 Level 0 DFD

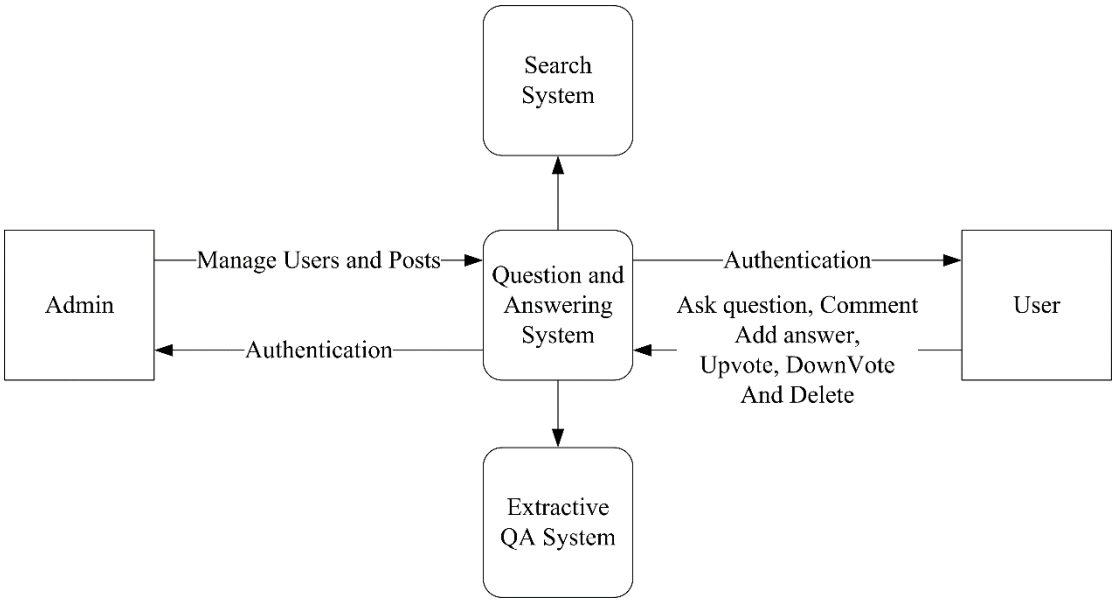

Figure 3.12: Level 0 DFD

The above shown figure is level 0 Data Flow Diagram of Question and Answering System. It shows “Question and Answering System” as process and “Search System” and “Extractive QA System” as sub-processes. “Admin” and “User” are external entities of the system.

The System asks Admin for authentication whereas Admin manage users and posts in the system. The System asks User for authentication whereas User can ask question, add answer, comment, upvote and downvote answers in the system.

3.6.3.2 Level 1 DFD

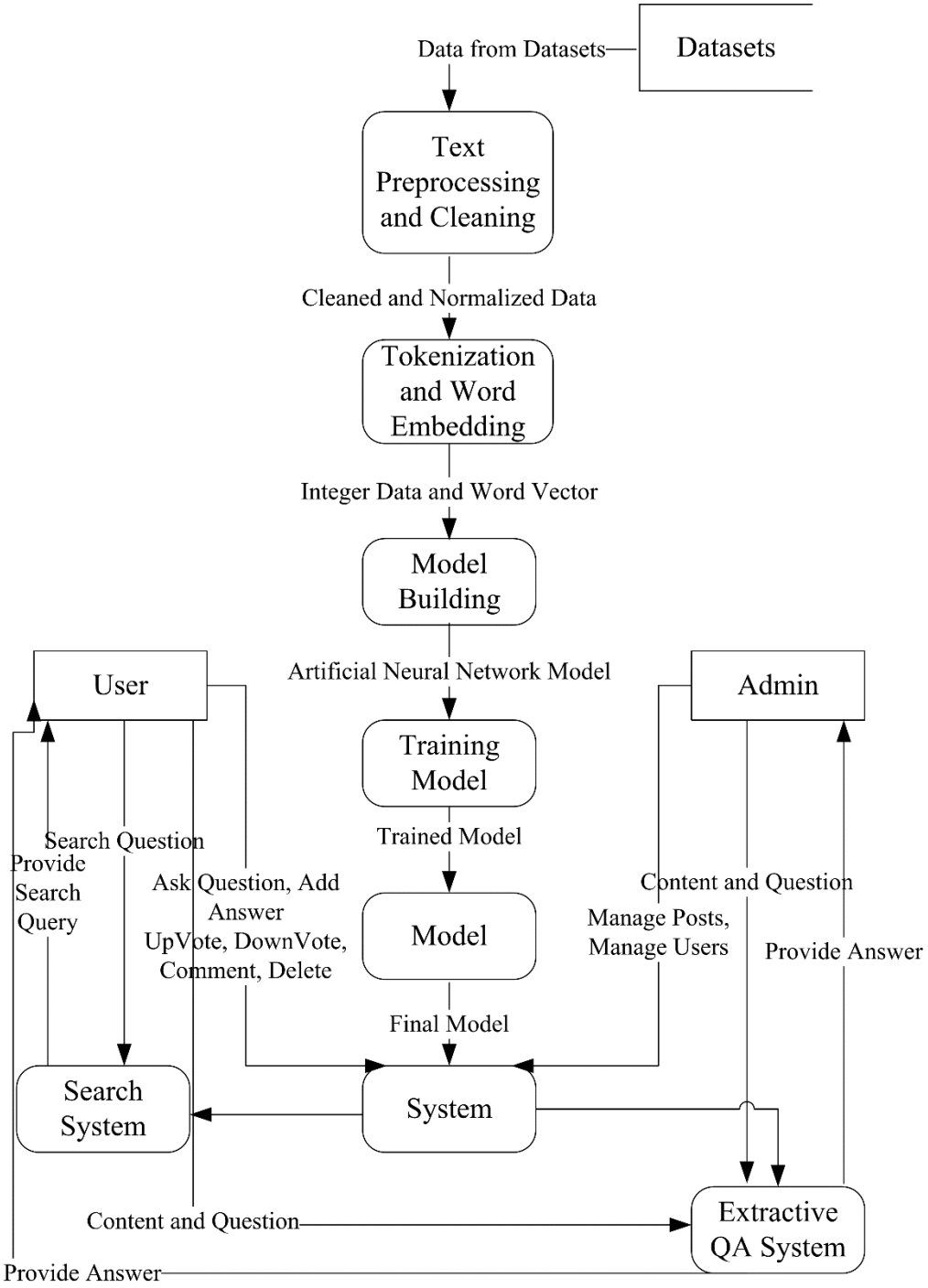

Figure 3.13: Level 1 DFD

3.6.4 Sequence Diagram

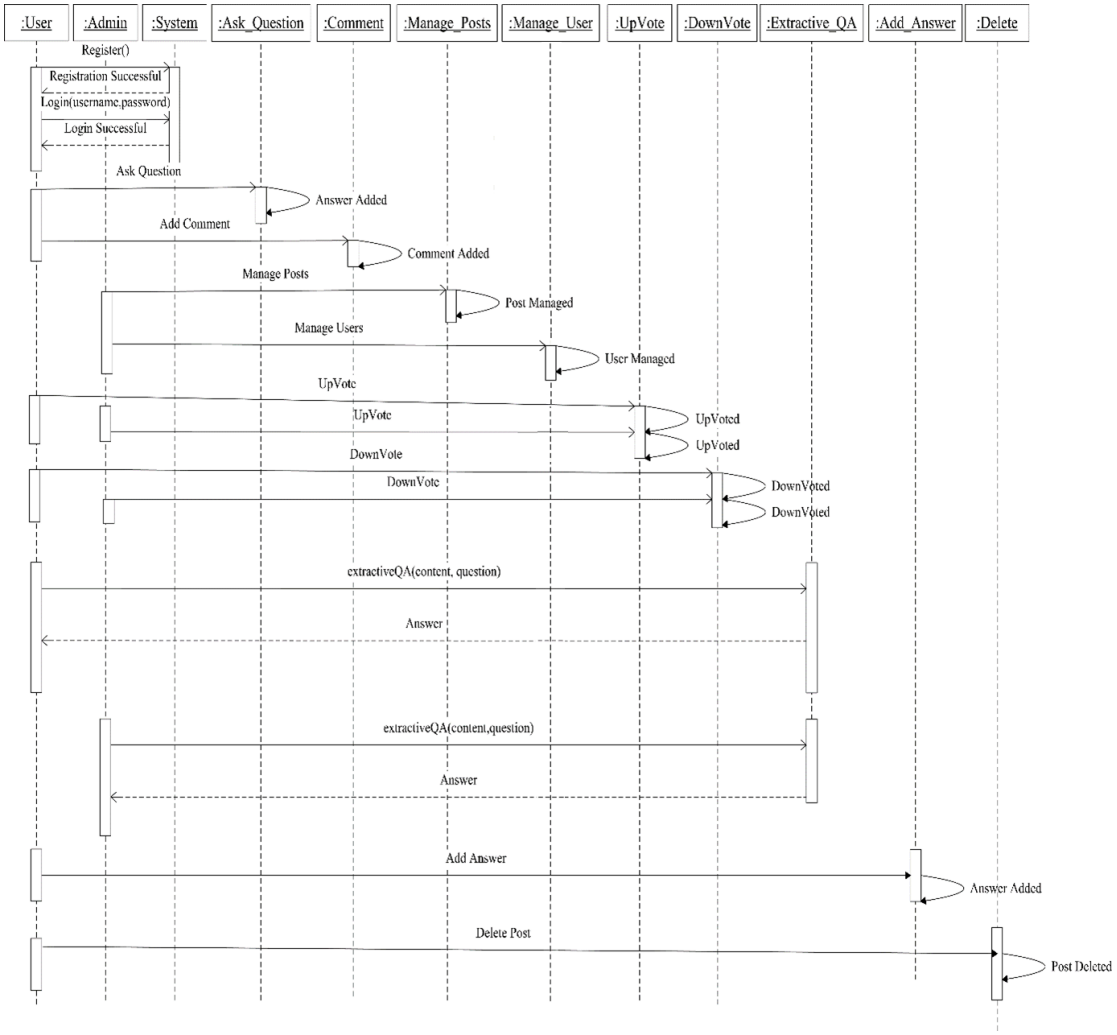

Figure 3.14: Sequence Diagram

The above shown figure is Sequence Diagram of Question and Answering System. A sequence diagram or system sequence diagram (SSD) shows process interactions arranged in time sequence in the field of software engineering. It depicts the processes involved and the sequence of messages exchanged between the processes needed to carry out the functionality.

3.6.5 Class Diagram

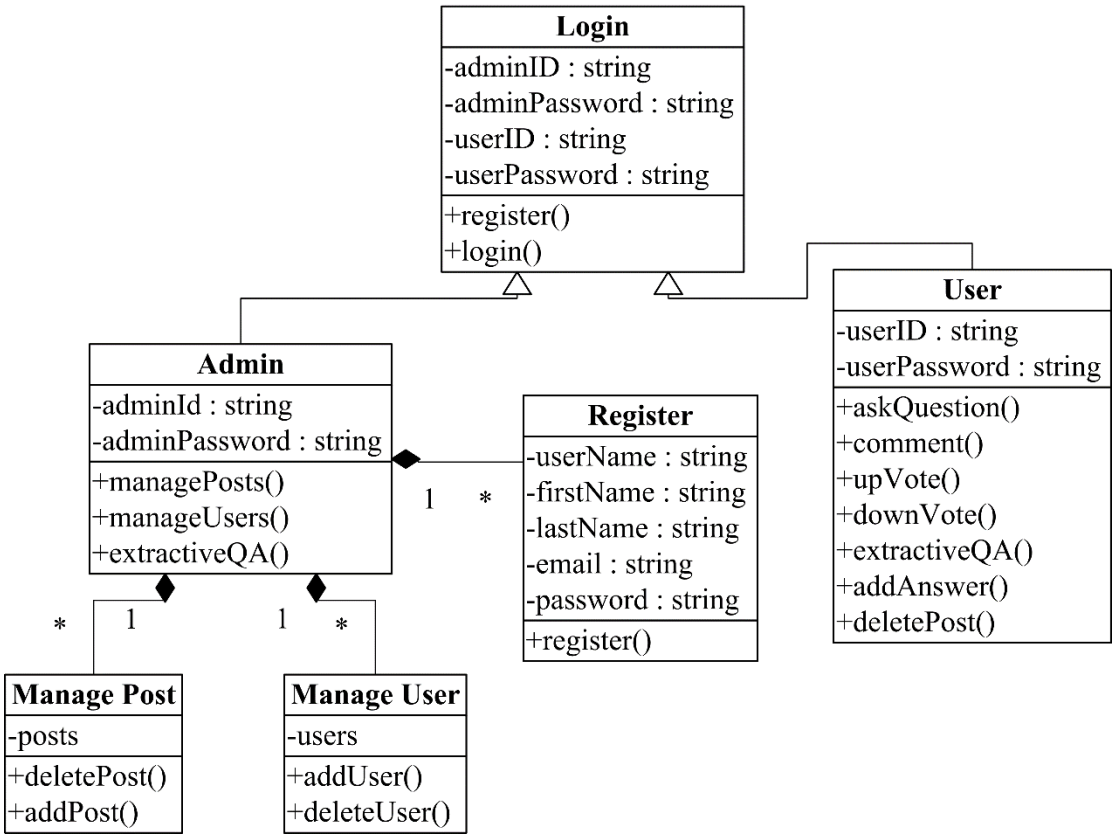

Figure 3.15: Class Diagram

The above shown figure is Class Diagram of Question and Answering System. The system has six classes namely “Login”, “Admin”, “User”, “Register”, “Manage Post” and “Manage User”.

Classes has certain attributes and operations. The class “Login” has “adminID”, “adminPassword”, “userID” and “userPassword” as attributes which are labeled private. Also it has “register()” and “login()” as operations which are labeled public. Class Admin and User are generalized form the parent class Login whereas class Register, Manage Post and Manage User are aggregated form the class Admin.

3.7 System Concept of Semantic Similarity GRU RNN model

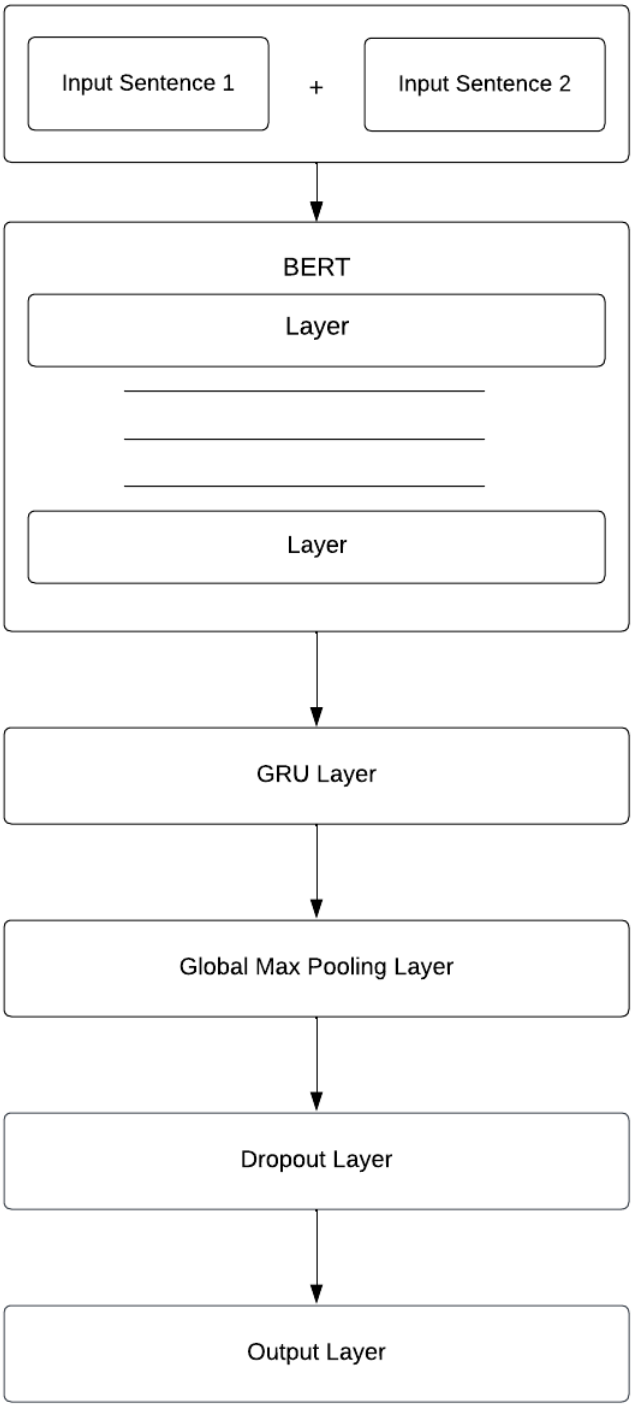

Figure 3.16: System Concept of Semantic Similarity GRU RNN model

3.8 ER Diagram

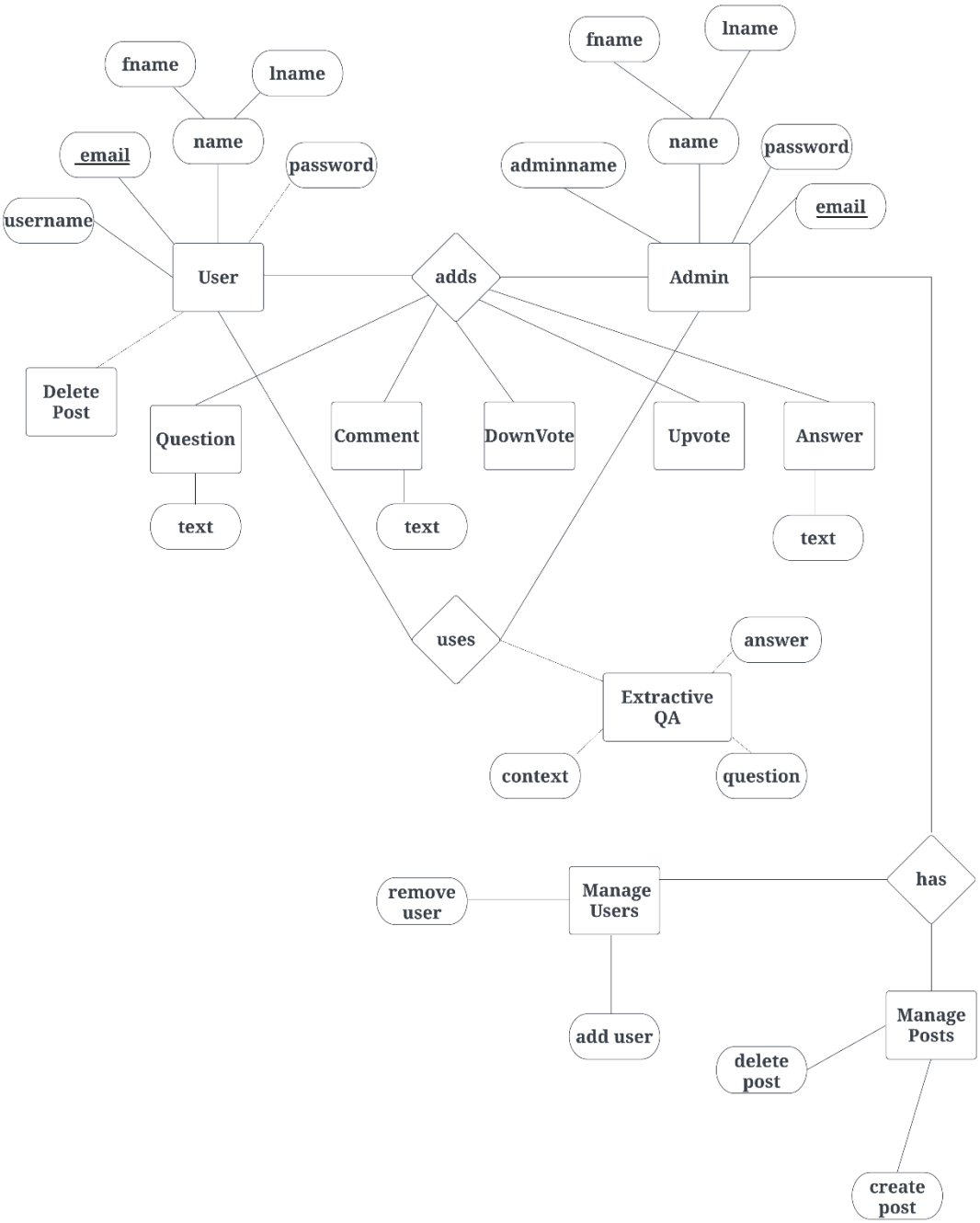

Figure 3.17: Entity Relationship Diagram

3.9 Tools Used

3.9.1 Python

Python is a high-level, general-purpose programming language. In the project, we used Python as the primary programming language for implementing various functionalities, including data preprocessing, feature engineering, model training, and web application development. Python's rich set of libraries and frameworks, such as NumPy, Pandas, Scikit-learn, TensorFlow, and Django, were extensively used to achieve these functionalities. We leveraged Python's functional programming constructs, such as lambda functions and list comprehensions, to write concise and readable code.

3.9.2 Django

Django is a free and open source, full stack web application framework written in Python. We used Django's ORM to define and manage the data models for storing the user input data and the results of the predictions. This made it easier for us to handle database operations without writing SQL queries manually. Additionally, we utilized Django's administrative GUI to manage the data and perform CRUD operations on the database easily.

3.9.3 Numpy

Numpy is the fundamental package for scientific computing in python. We used NumPy arrays to represent the image and text data, which were then processed using various mathematical and statistical operations, such as normalization, scaling, and vectorization. NumPy's broadcasting feature was also utilized to perform element-wise operations on the arrays, which made it easier to manipulate the data in bulk.

3.9.4 Pandas

Pandas is an open-source library that is made mainly for working with relational or labeled data intuitively. We utilized Pandas' data manipulation features, such as data merging, reshaping, splitting, and pivoting, to preprocess and transform the data into a format suitable for machine learning models. Pandas' easy handling of missing data also helped us to deal with missing values in the dataset effectively.

3.9.5 Matplotlib

Matplotlib is a plotting library for the Python programming language. It helps to generate histograms, bar charts and other types of charts and plots with just few lines of code. We utilized Matplotlib's customization features, such as adding titles, labels, and legends to the plots, to make the visualizations more informative and visually appealing.

3.9.6 Google Colab

It is a free Jupyter notebook environment that runs entirely on cloud and doesn't require a setup. Colab's cloud-based architecture provided us with access to a free GPU, which was instrumental in training complex machine learning models quickly and efficiently. This significantly reduced the training time, which allowed us to iterate on the models and experiment with different architectures and hyperparameters.

3.9.7 HTML & CSS

In our project, we utilized HTML and CSS extensively for creating the user interface of the web application. HTML was used to create the basic structure of the web pages, including headings, paragraphs, links, and images. HTML also helped us to add interactivity to the web pages by creating forms, buttons, and other user interface elements.

CSS, on the other hand, was used to design the web pages and make them visually appealing. We utilized CSS to apply styles to various HTML elements, including fonts, colors, backgrounds, borders, and spacing. CSS's powerful features, such as selectors, inheritance, and cascading, enabled us to create complex layouts and designs with ease.

3.9.8 JavaScript

JavaScript is a light-weight object-oriented programming language which is used by several websites for scripting the webpages. One of the key features of JavaScript that we utilized in our project was its ability to make asynchronous requests to web servers and update the web pages in real-time. We utilized JavaScript's AJAX (Asynchronous JavaScript and XML) functionality to fetch data from the server and update the web pages without requiring a page refresh.

3.9.9 SQLite

SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. We utilized the Python library sqlite3 to

interact with the SQLite database and perform various database operations, such as creating tables, inserting data, updating records, and querying data.

3.9.10 Transformers

Hugging Face transformers is a platform that provides the community with APIs to access and use state-of-the-art pre-trained models available from the Hugging Face Hub. We used transformers library's pre-processing and post-processing APIs to tokenize and encode the input text, and used the pre-trained model's forward pass to obtain the predictions.

3.9.11 Tensorflow

Tensorflow is an open-source software library for numerical computation using data flow graphs. TensorFlow's flexibility and scalability made it easy for us to experiment with different architectures and hyperparameters, and to scale our models to handle large datasets and complex tasks.

3.9.12 VS Code

Visual Studio Code (also known as VS Code) is a free open-source text editor by Microsoft. We used VS Code as our primary text editor and integrated development environment (IDE). We found VS Code's built-in code highlighting, auto-completion, and debugging features to be extremely useful for writing and debugging our code.

3.9.13 Simple Transformers

Simple Transformers is a popular open-source library for natural language processing (NLP) tasks, built on top of the powerful PyTorch framework. We leveraged the library's pre-trained models for transfer learning, fine-tuned them on our specific dataset, and achieved state-of-the-art results. The library's easy-to-use interface allowed us to experiment with different architectures and hyperparameters quickly, enabling us to optimize the model's performance efficiently. It also provided easy way to save and store model and load again for inference through the django backend.

3.10 Verification and Validation

Semantic Similarity Model

We used BERT and GRU and implemented a semantic similarity model and verified its correctness. We used a 120,000 data points of sentence pairs and split them in ratio of 60:20:20 where 60% dataset was used for training and 20% each dataset was used for model validation and testing, and for each pair, we labeled the degree of similarity between the sentences. We collected this dataset from SNLI corpus. To ensure the correctness of the model, we performed the following verification steps:

Data preprocessing and Feature Engineering: We carefully preprocessed the input data to ensure that it was in the correct format for the model to process. We performed cleaning data by removing unnecessary noise and null values and also did label encoding to ensure that the input was consistent. We used pre-trained BERT embedding to capture the meaning of the sentences and fed them as inputs to the GRU.

Model training: We trained the model on the preprocessed data and tuned the hyper parameters to achieve the best performance on the validation set. We used categorical cross entropy loss and Adam optimizer to optimize the model. We also tried different hyperparamters like such as learning rate, batch size, number of epochs, and regularization techniques to improve the model's performance. During training, we monitored the model's performance on both the training and validation sets to detect overfitting. So we used dropout regularization to prevent overfitting and improve generalization.

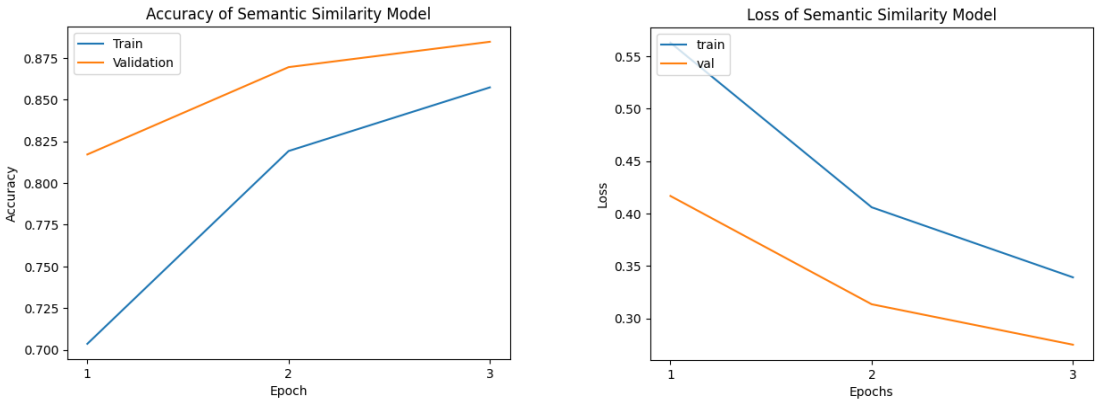

Figure 3.18: Train, validation Accuracy and Loss of Semantic Similarity model by Freezing BERT Layers

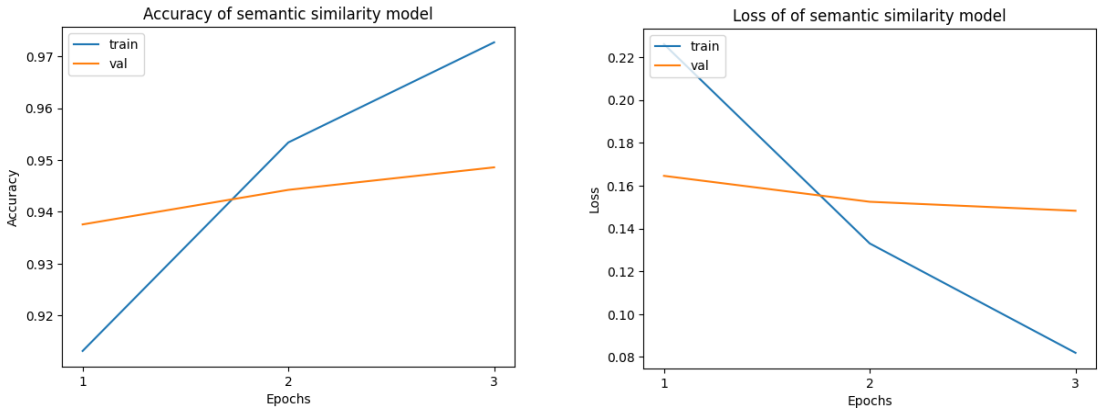

Figure 3.19: Train, validation Accuracy and Loss of Semantic Similarity model by Unfreezing BERT Layers and Fine tuning

Model evaluation: We evaluated the model's performance on the test set to ensure that it was approximately 94% accurate. The precision of model is around 93% which means that it can also be used for duplicate question detection and rarely make false positive prediction. We calculated the values for recall, and F1 score and classification report:

	precision	recall	f1-score	support
0.0	0.960	0.937	0.949	10002
1.0	0.939	0.961	0.950	10000
accuracy			0.949	20002
macro avg	0.949	0.949	0.949	20002
weighted avg	0.949	0.949	0.949	20002

Figure 3.20: Classification Report of Semantic Similarity model

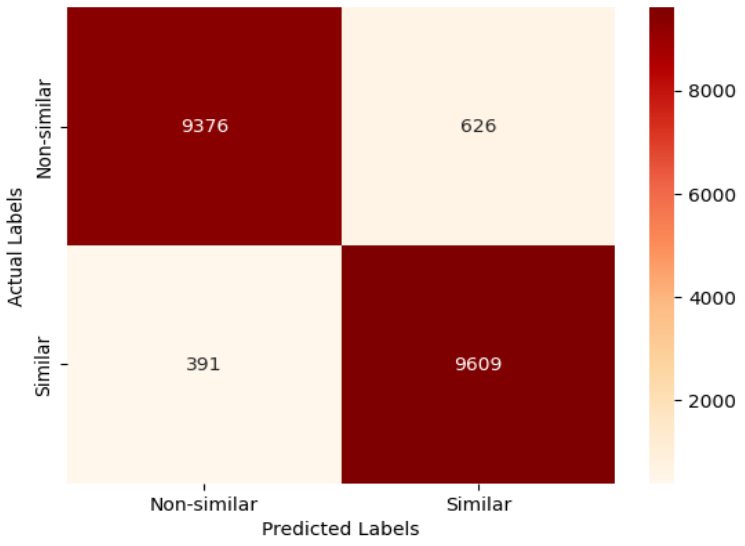

Figure 3.21: Confusion Matrix of Semantic Similarity model

The macro-averaged F1 score, precision, and recall are also provided, which are the average of the metrics for each class. The weighted average of these metrics is also given, where the weighting takes into account the number of instances in each class. Overall, this classification report indicates that the model performed well in predicting both classes with high precision and recall scores.

Based on our verification steps, we are confident in the correctness of our Semantic Similarity model.

Extractive QA model

Here, we used Simple Transformers and implemented an extractive QA model. We used 100,000 data points context, question and answers and split them in ratio of 70:15:15 where 70% dataset was used for training and 15% each dataset were used for validation and testing. To ensure the correctness of the model, we performed the following verification steps:

Data preprocessing: We carefully preprocessed the input data to ensure that it was in the correct format for the model to process. We converted the JSON SQuAD dataset which was not in correct format, to the list of Python dictionary which is the correct format of training as required by the Simple Transformers.

Model training: We trained the model on the preprocessed data and tuned the hyper parameters like learning rate as well as other parameters to achieve the best performance on the validation set. We used a pre-trained language model as a starting point and fine-tuned it on the task of question answering.

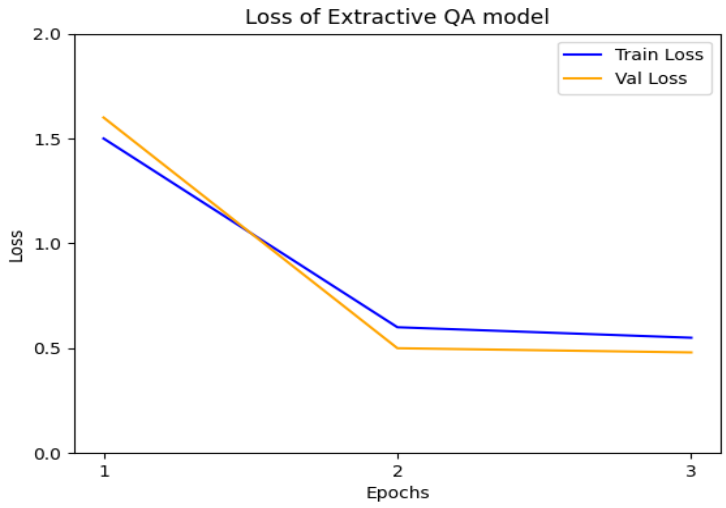

Figure 3.22: Train, eval loss of Extractive QA model

Model evaluation: We evaluated the model's performance on the test set to ensure that it was accurate and reliable by observing its loss on training and validation set during training. While the model answered approximate 10000 answers with Exact Match (EM) it also answered 4,000 questions with a Substring Match and around 1000 questions incorrectly. These results indicate that there is still room for improvement in the model's performance, and we plan to explore ways to increase its accuracy and reduce the number of similar and wrong answers.

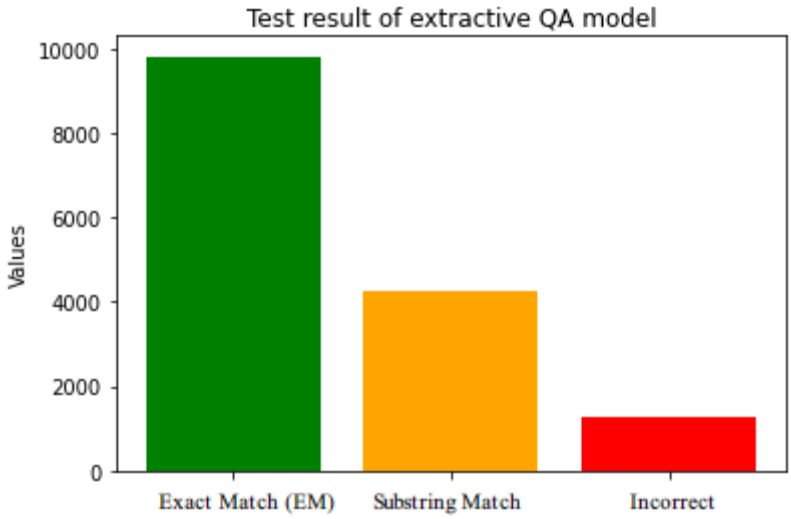

Figure 3.23: Exact Match (EM) and Substring Match count of Extractive QA model

Based on our verification steps, we are confident in the correctness of our extractive QA model using Simple Transformers.

CHAPTER 4: EPILOGUE

4.1 Result

The text semantic similarity model achieved an accuracy of 94% in the test set. The precision was also approximately 94%. During manual testing across different data, we found that the model was consistent. Initially the model was over fitting, but when we reduced the GRU n-units to 32 and added a regularization layer with Dropdown of 30% the model started to generalize well for unseen data.

Similarly, the Extractive QA model gave Exact Match (EM) score of 66.67% and sub-string match 26.67 % and incorrect score of 5.6%. In this model we improved the EM score after we observed that the length of answers were short and assigned max-answer-length hyperparameter as 20.

Finally, we developed a web application and used these models where the Semantic similarity model is used for Duplicate Question Detection and searching, while the Extractive QA model is used to find answer by providing paragraph and context.

4.2 Conclusion

We were able to train a text semantic similarity model using GRU and BERT and achieve good score in test set. The evaluation metrics such as accuracy and precision indicated that the model performed well in predicting the degree of similarity between sentences. The model can be used in various natural language processing applications such as duplicate question detection and information retrieval. The extractive QA model also gave satisfactory results but still there's a room for improvement to address the issue of incorrect answers. Overall, this project provides a valuable example of how to build an intelligent Question and Answering system using modern web development and machine learning techniques, and can serve as a starting point for further exploration and development of intelligent applications.

4.3 Future Enhancement

Some possible future enhancements that could be done are:

Fine-tuning the model on specific domain: The pre-trained model could be fine-tuned on specific domains or topics, to further improve its accuracy and relevance for specific use cases or industries.

Multilingual and large document support: The system could be expanded to support multiple languages, either by training separate models for different languages or by incorporating language detection and translation capabilities and also support large documents.

BIBLIOGRAPHY

1. Mall, Rajib. Fundamentals of Software Engineering. 4th ed., PHI Learning Pvt. Ltd., 2017.
2. Larman, Craig. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. 3rd ed., Pearson Education Asia, 2008.
3. Tan, Ning, Steinbach, Michael, and Kumar, Vipin. Introduction to Data Mining. Addison-Wesley, 2005.
4. Thakur, Ayush. “Simple Transformers: Transformers Made Easy.” Weights & Biases, 30 Nov. 2022
5. Briggs, James. “TensorFlow and Transformers” Towardsdatascience, 20 Nov. 2020

REFERENCES

[1] Ansari, Navedanjum, and Rajesh Sharma. "Identifying semantically duplicate questions using data science approach: A Quora case study." *arXiv preprint arXiv:2004.11694* (2020).

[2] Imtiaz, Zainab, et al. "Duplicate questions pair detection using Siamese Malstm." *IEEE Access* 8 (2020): 21932-21942.

[3] Patel, Uday, Amol Dattu, Pritam Patil, Renuka Khot, and Sujit Tilak. "Quora Question Duplication Problem." (2020).

[4] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, ‘‘Signature verification using a ‘siamese’ time delay neural network,’’ in Proc. 6th Int. Conf. Neural Inf. Process. Syst. (NIPS), San Francisco, CA, USA, 1993, pp. 737–744.

[5] Travis Addair. 2017. Duplicate question pair detection with deep learning. Stanf. Univ. J (2017).

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

[7] A Tung and E Xu. 2017. Determining Entailment of Questions in the Quora Dataset., 8 pages.

[8] A Parikh, O Tckstrm, D Das, and J Uszkoreit. 2016. A Decomposable Attention Model for Natural Language Inference. Proc. 2016 Conf. Empir. Methods Nat. Lang. Process (2016), 2249–2255.

[9] Rajpurkar, P., Zhang, J., Lopyrev, K., & Liang, P. (2016). SQuAD: 100,000+ questions for machine comprehension of text. arXiv preprint arXiv:1606.05250.

SCREENSHOTS

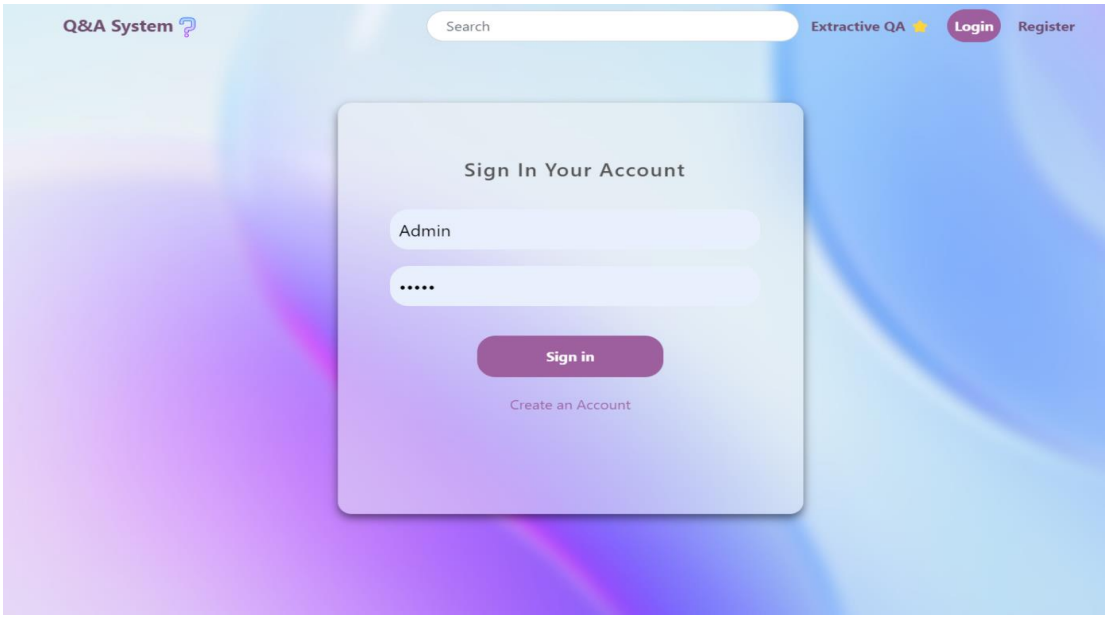

Figure: Sign-in/ Login page of Webapp

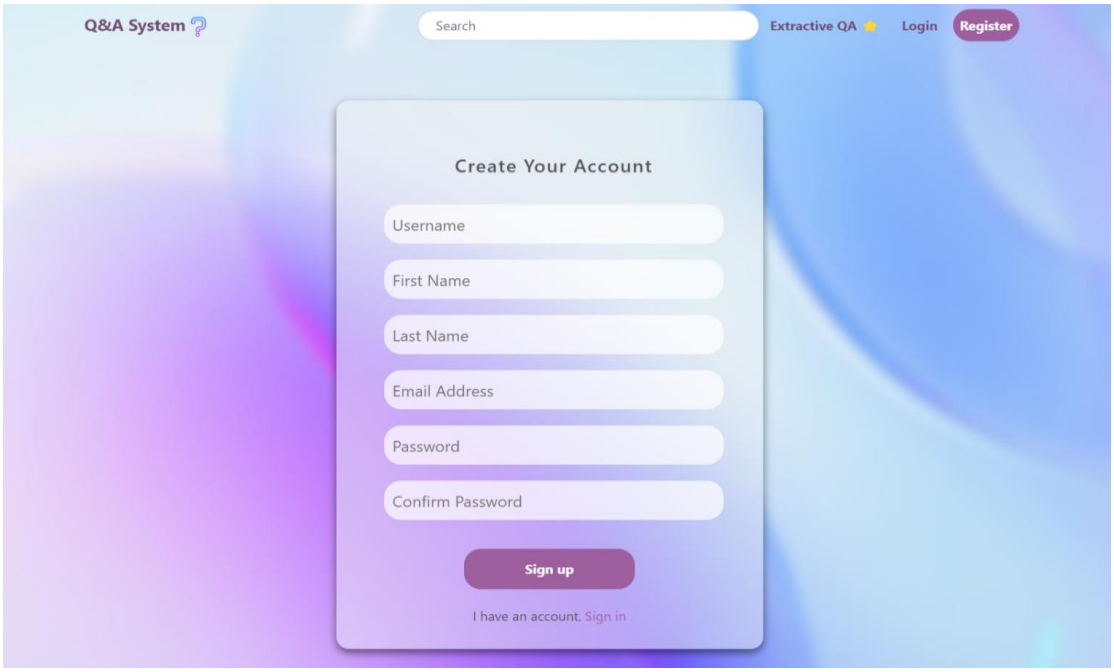

Figure : Sign-up Page

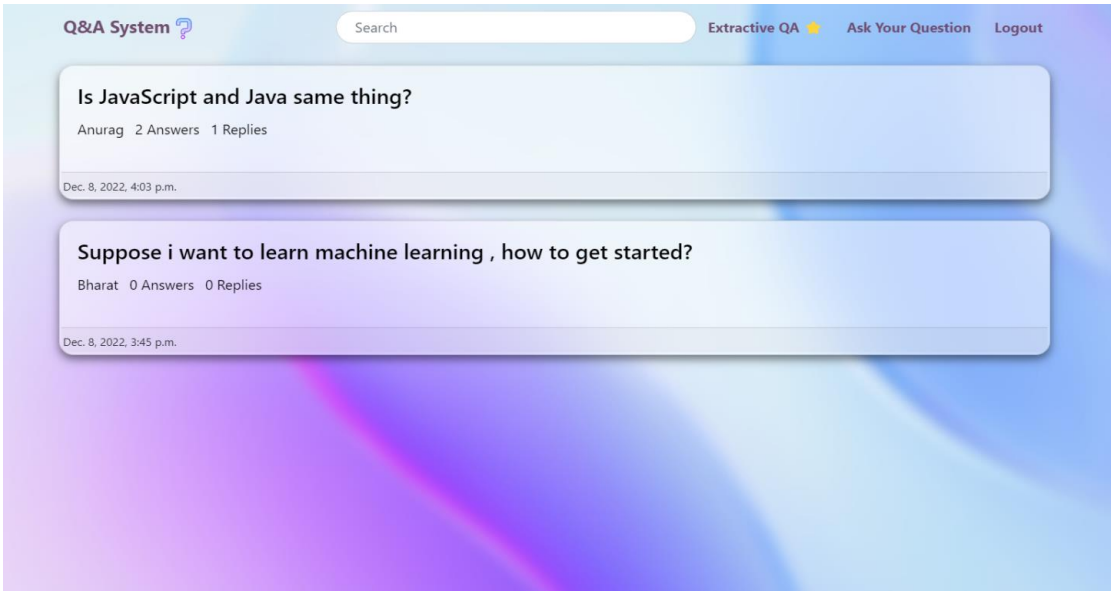

Figure : Homepage with questions asked by different users

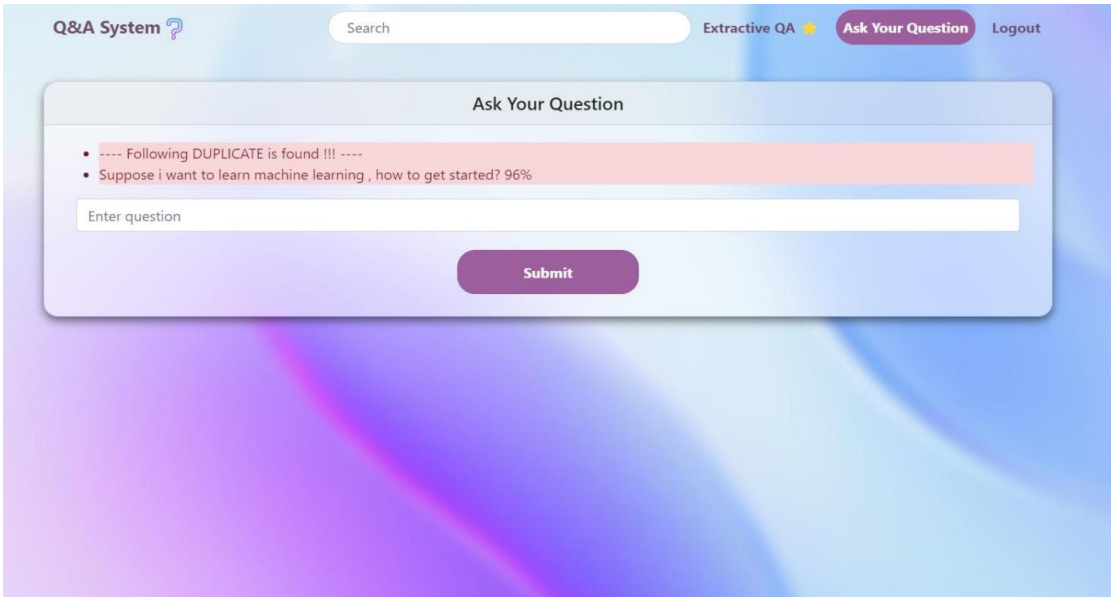

Figure : Duplicate Question Detection using Semantic Similarity model, while adding New Question

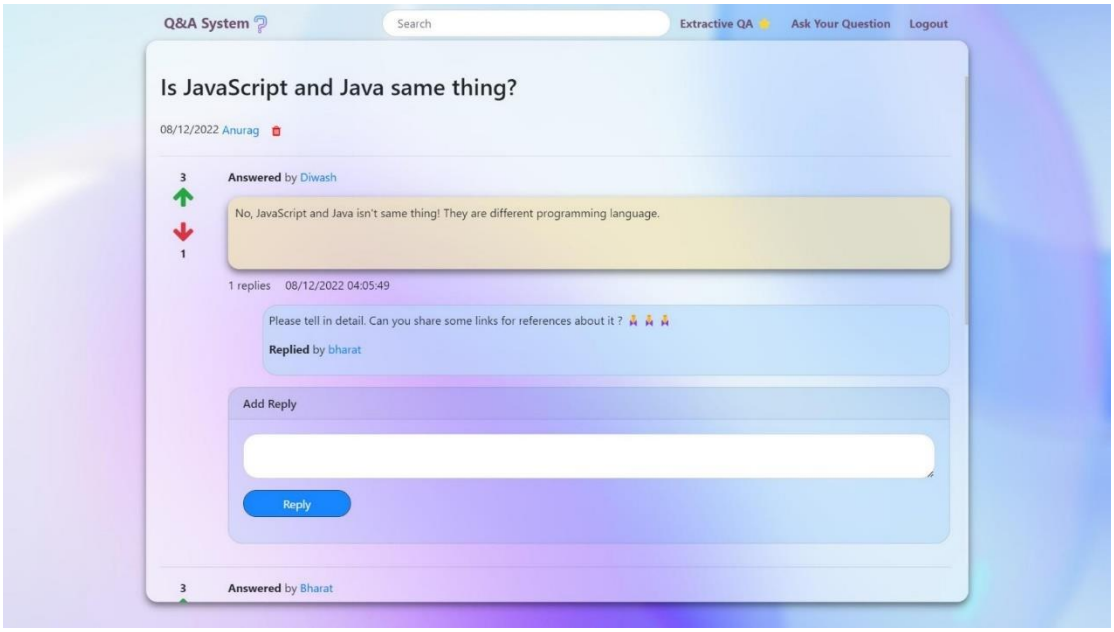

Figure : Detail Page of Question

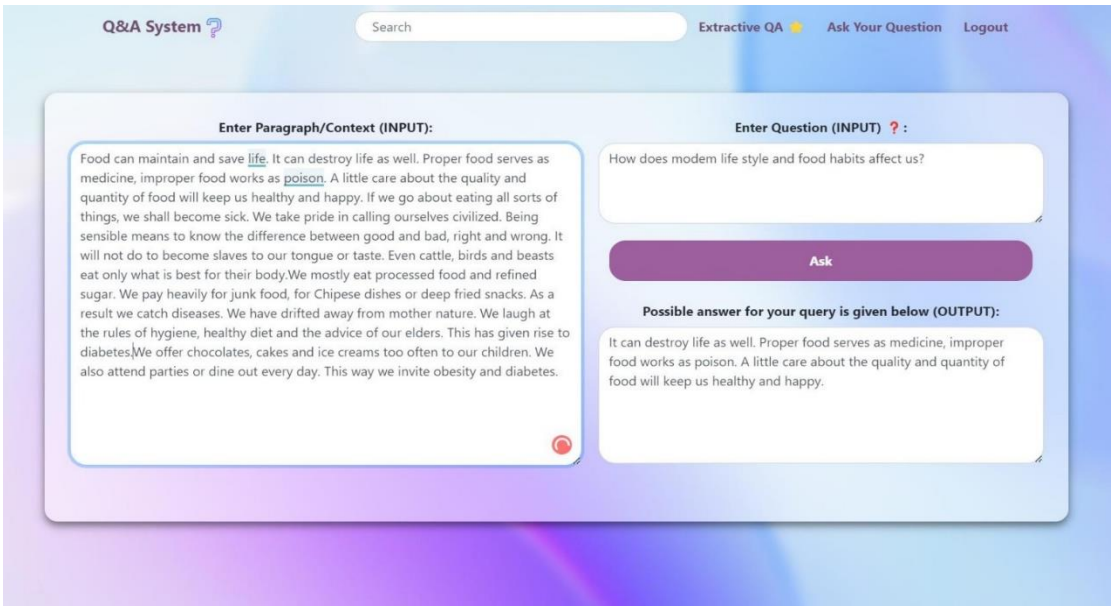

Figure : Answering by Extractive QA model when Context and Question are passed

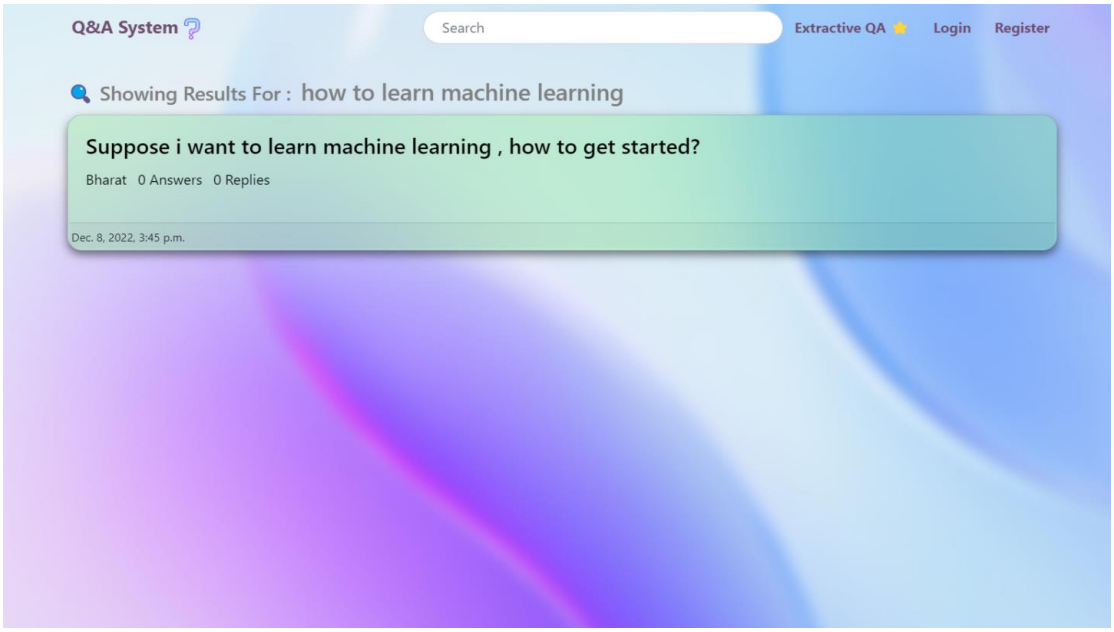

Figure : Searching using semantic similarity model