Indian Institute of Technology Madras

CH5019: Mathematical Foundations of Data Science

Course Project Report
Department of Chemical Engineering

Automated Descriptive Answer Grader

Group - 6:

Nitin Yadav	AM20M004
Pranjul Dubey	ED21S011
Saravanan M	AM21D010
Sriram A	AM21S013

Project Advisor:

Prof. Raghunathan Rengaswamy

Acknowledgment

I express my deep gratitude to almighty, the supreme guide, for bestowing his blessings up on us in my entire endeavor.

I would like to thank to my guide **Prof. Raghunathan Rengaswamy** for his valuable guidance, keen interest, timely suggestions and thus sparing his valuable time throughout the project work. His encouraging attitude has been immensely valuable in keeping me motivated and helping me in overcoming obstacles in my work. He has been a great teacher, a great guide and above all a great human being. I'm blessed to have him as my project advisor.

I'm grateful to **Karthick Raj** & **Raj Jain**, teaching assistance for being a source of inspiration and providing me constant support, constructive criticism, and invaluable suggestions.

Contents

Introduction

An instructor wants to grade answers to descriptive questions automatically. The instructor has a template best answer that she/he has developed and wants to use the same to grade descriptive answers of students.
To achieve this, we attempt to make a machine learning & natural language processing-based system which would automatically grade descriptive answers based on the given reference answers.

Data Set

Our model for grading answers is applicable to any general descriptive question answer set. But to show how our code works we have taken data set consisting of 10 questions generally asked in a data science interviews along with their template best answers. These questions are taken from this website link.

In our model, we have manually generated sample answer set of five answers from template best answer for each of the ten questions. We have also manually assigned marks out of 10 to each of these sample answers.

There is total 11 data set files used in our code. Each data set is in csv (comma separated values) format.

For build our algorithm we have taken following assumptions about the template best as well as students answers:

- Answer should not contain diagrams, figures, formulae, or any non-alpha numeric character
- Answers should be only written in English language
- Answers should be in typed character form (**NO** scanned images)

Methodology

Our algorithm is an implementation of natural language processing. (NLP). After reading the data set & displaying on screen all the question-answer pairs, user is asked to type the question number out of 10, that he wants to ask the student. After selecting, the chosen question & its corresponding standard answer is displayed on the screen.

Now, the five sample answers to the chosen question-answer pair is loaded by reading from already created data sets (Example: Sample_1.csv for Q1). For the asked question, we have to find the sample answer, closest in meaning to the students answer & display the marks gained by student for that particular question according to the retrieved sample answer.

To compare two answer scripts, we need to have an efficient way of computing semantic similarity between two sentences. To compute semantic similarity between sentences, we will convert each sentence into a vector. We can then use cosine similarity between vectors to come up with a distance measure between sentences that indicates how similar they are in meaning. This was done using NLP, as follows

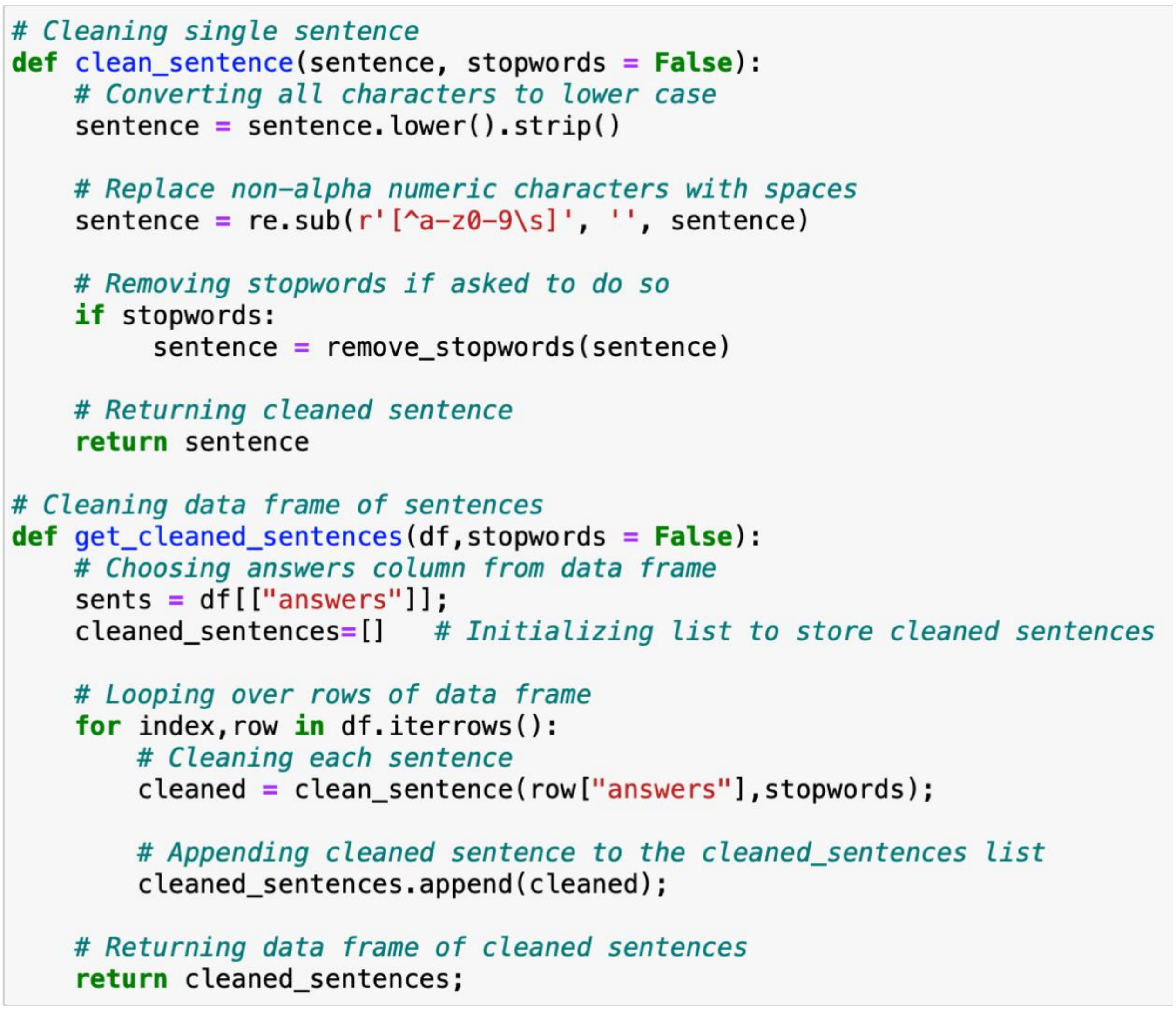
Pre-processing (Data cleaning)

After loading sample answers, the students answer is taken as input. Now, both the sample answers & students answer are pre-processed.

In pre-processing we are doing following operations on the answers:

- Convert all characters to lower case
- Replace non-alpha numeric characters with spaces
- We have also removed stopwords, which are commonly used words such as 'a', 'to', 'in' and so on

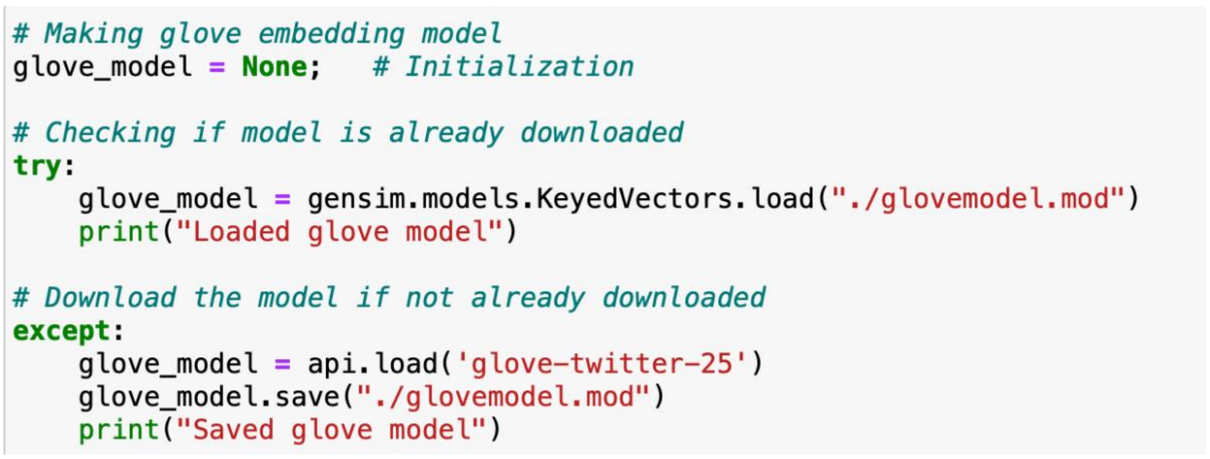
Its python implementation is as follows:



```python
# Cleaning single sentence
def clean_sentence(sentence, stopwords = False):
    # Converting all characters to lower case
    sentence = sentence.lower().strip()

    # Replace non-alpha numeric characters with spaces
    sentence = re.sub(r'[^a-z0-9\s]', '', sentence)

    # Removing stopwords if asked to do so
    if stopwords:
         sentence = remove_stopwords(sentence)

    # Returning cleaned sentence
    return sentence

# Cleaning data frame of sentences
def get_cleaned_sentences(df,stopwords = False):
    # Choosing answers column from data frame
    sents = df[["answers"]];
    cleaned_sentences=[]   # Initializing list to store cleaned sentences

    # Looping over rows of data frame
    for index,row in df.iterrows():
        # Cleaning each sentence
        cleaned = clean_sentence(row["answers"],stopwords);

        # Appending cleaned sentence to the cleaned_sentences list
        cleaned_sentences.append(cleaned);

    # Returning data frame of cleaned sentences
    return cleaned_sentences;
```

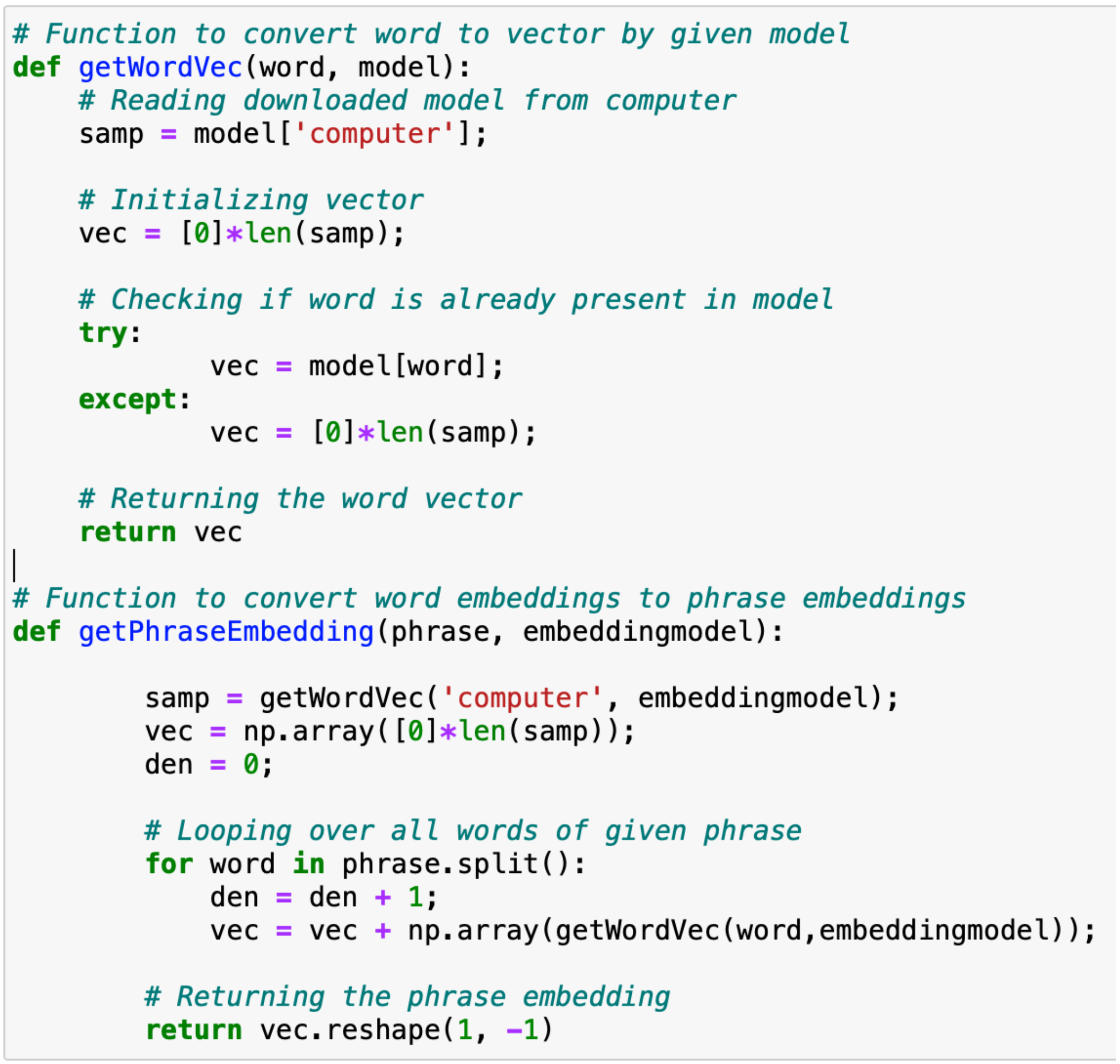
Training Approach

We have trained our model using GloVe Embeddings, which are Global Vectors for Word Representation. GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. Its python implementation is as follows:


```python
# Making glove embedding model
glove_model = None;    # Initialization

# Checking if model is already downloaded
try:
    glove_model = gensim.models.KeyedVectors.load("./glovemodel.mod")
    print("Loaded glove model")

# Download the model if not already downloaded
except:
    glove_model = api.load('glove-twitter-25')
    glove_model.save("./glovemodel.mod")
    print("Saved glove model")
```

<u>Phrase embeddings from word embeddings</u>

To convert word embeddings to phrase embeddings, we have summed up the individual word embeddings in the phrase to get a phrase vector. Its python implementation is as follows:


```python
# Function to convert word to vector by given model
def getWordVec(word, model):
    # Reading downloaded model from computer
    samp = model['computer'];

    # Initializing vector
    vec = [0]*len(samp);

    # Checking if word is already present in model
    try:
            vec = model[word];
    except:
            vec = [0]*len(samp);

    # Returning the word vector
    return vec

# Function to convert word embeddings to phrase embeddings
def getPhraseEmbedding(phrase, embeddingmodel):

        samp = getWordVec('computer', embeddingmodel);
        vec = np.array([0]*len(samp));
        den = 0;

        # Looping over all words of given phrase
        for word in phrase.split():
            den = den + 1;
            vec = vec + np.array(getWordVec(word,embeddingmodel));

        # Returning the phrase embedding
        return vec.reshape(1, -1)
```

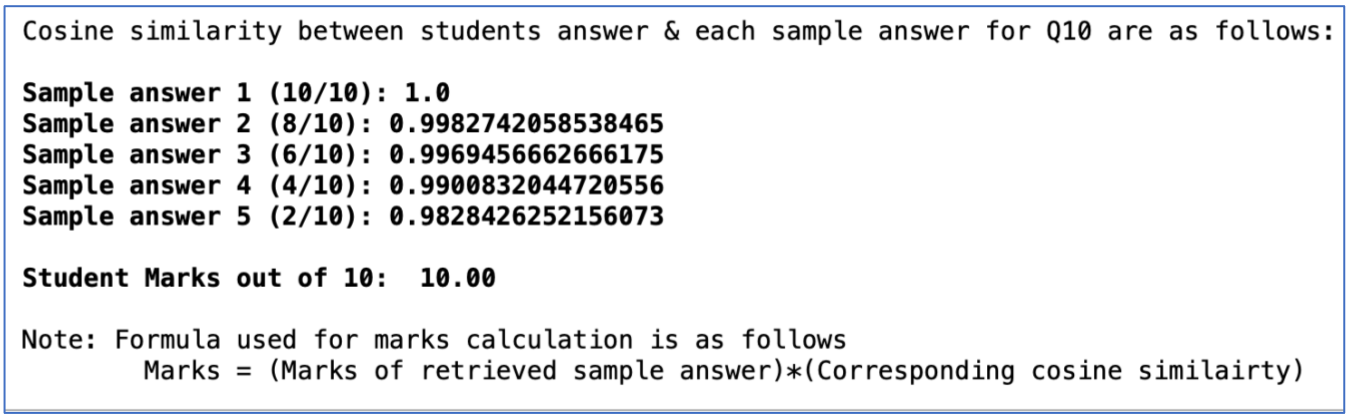
Prediction Approach

Finally, we converted the phrases of sample answers & student answer using our trained model. Then we have used these embeddings to compute cosine similarity between sample answer & student answer. We use these cosine similarities to retrieve the most matching sample answer.

Marks are awarded to the student using below formula

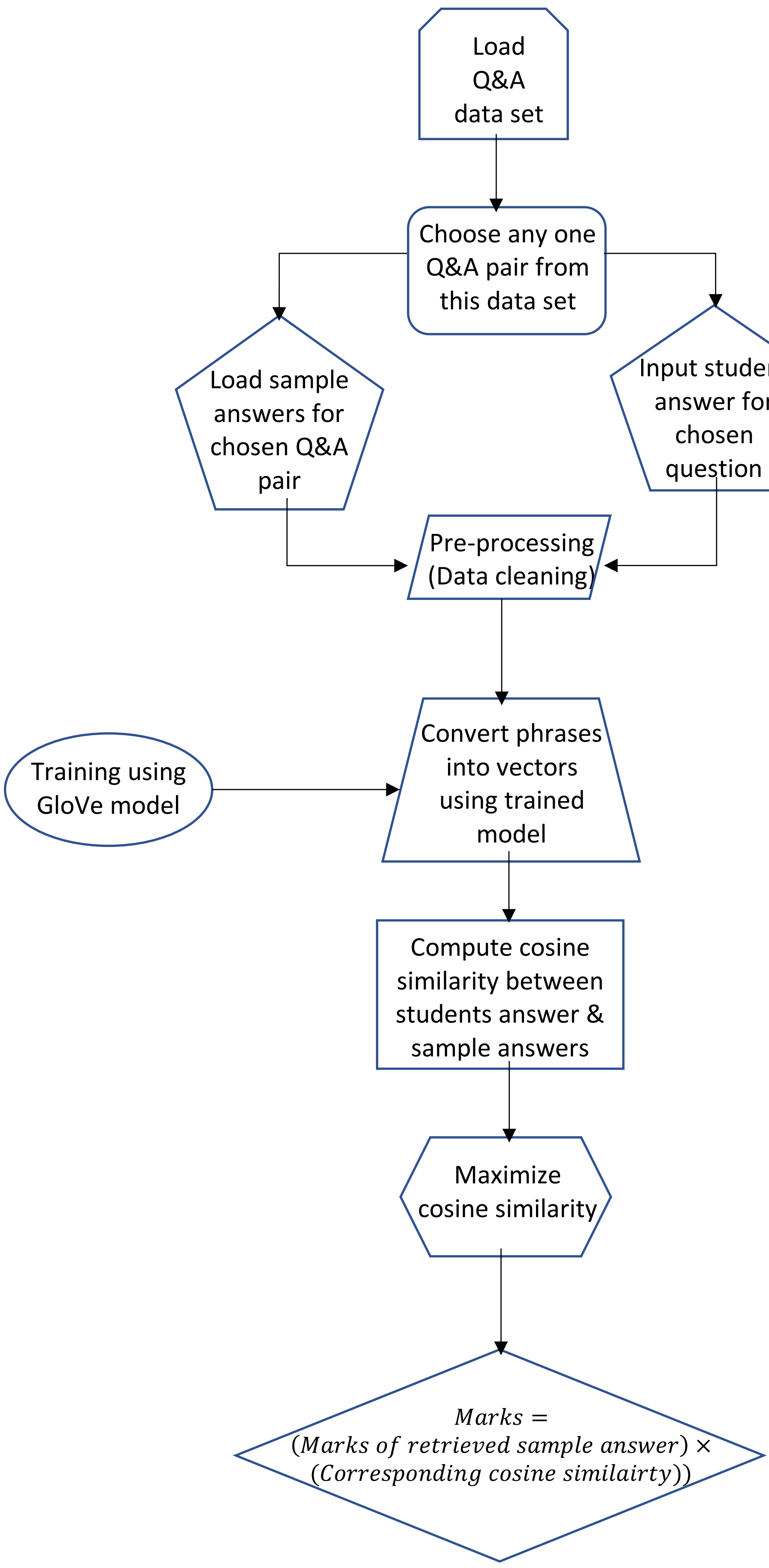
$$\boxed{Marks = (Marks\ of\ retrieved\ sample\ answer) \times (Corresponding\ cosine\ similairty)}$$

Final output screen looks as follows:



```
Cosine similarity between students answer & each sample answer for Q10 are as follows:

Sample answer 1 (10/10): 1.0
Sample answer 2 (8/10): 0.9982742058538465
Sample answer 3 (6/10): 0.9969456662666175
Sample answer 4 (4/10): 0.9900832044720556
Sample answer 5 (2/10): 0.9828426252156073

Student Marks out of 10:  10.00

Note: Formula used for marks calculation is as follows
        Marks = (Marks of retrieved sample answer)*(Corresponding cosine similairty)
```

Algorithm Flowchart


Load Q&A data set
Choose any one Q&A pair from this data set
Load sample answers for chosen Q&A pair
Input student answer for chosen question
Pre-processing (Data cleaning)
Training using GloVe model
Convert phrases into vectors using trained model
Compute cosine similarity between students answer & sample answers
Maximize cosine similarity
Marks = (Marks of retrieved sample answer) × (Corresponding cosine similairty))

Contribution Statement

	Process contribution	***Code contribution***
Nitin Yadav (AM20M004)	Conceptualization	Phrase embeddings from word embeddings, Retrieving sample answer having maximum cosine similarity
Pranjul Dubey (ED21S011)	Methodology	Training using GloVe model, Presenting results
Saravanan M (AM21D010)	Sample answer creation	Reading and presenting Q&A data set, Loading libraries, Reading sample answers for chosen question
Sriram A (AM21S013)	Sample Q&A data collection	Choosing any one Q&A pair out of given data set, Pre-processing data

Appendix

Complete python code, jupyter notebook, data set & project report can be accessed from this link GitHub