ML&NLP Project
Chatbot

Members:-

Aniket Choudhary	18803002	B13
Vaibhav Gupta	18803006	B13
Sanyam Goel	18803031	B13

Submitted To:-

Dr. K Vimal

CSE Department, JIIT Noida, Sector - 62

main.py

```
import chatbot
from flask import *

app = Flask(__name__)


@app.route("/", methods=["POST", "GET"])
def lnrs():
    if request.method == 'GET':
        return render_template('index.html', otp="")
    elif request.method == 'POST':
        query = request.form["query"]
        a=chatbot.querries(query)
        return render_template('index.html', otp=a)


if __name__ == '__main__':
    app.run(debug=True)
```

Chatbot.py

```
import bs4 as bs
import urllib.request
```

```
import re
import nltk
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import TfidfVectorizer


# function to answer a query
def querries(query):
    gett_data =
urllib.request.urlopen("https://en.wikipedia.org/wiki/India").read()
    gett_data_paragraphs = bs.BeautifulSoup(gett_data,
'lxml').find_all('p')
    gett_text = ''
    for p in gett_data_paragraphs:
        gett_text += p.text.lower()
    # print(gett_text)

    gett_text = re.sub(r'\s+', ' ', re.sub(r'\[[0-9]*]', ' ', gett_text))

    nltk.download('punkt')
    gett_sentences = nltk.sent_tokenize(gett_text)



    def chatbot_answer(user_query):
        # adding the input query at the last position of sentence vector
        gett_sentences.append(user_query)
        # creation of vectorizer based on tfidf metric
        vectorizer = TfidfVectorizer()
        # getting tfidf values for all sentence vectors
        sentences_vectors = vectorizer.fit_transform(gett_sentences)

        # calculation of cosine similarity metric for all sentences w.r.t.
input query sentence
        vector_values = cosine_similarity(sentences_vectors[-1],
sentences_vectors)
        # Extracting the sentence with second highest cosine similarity
value
        # All the sentences are sorted in increasing order of cosine
similarity
```

```
        # As the last sentence is the input query itself, hence it has
highest cosine similarity
        # Therefore, we are choosing the sentence with second highest
cosine similarity
        answer = gett_sentences[vector_values.argsort()[0][-2]]
        # getting the actual sentence from the senctence vector
        input_check = vector_values.flatten()
        # Here the sorting is happening
        input_check.sort()

        # checking if the sentence is valid or not
        if input_check[-2] == 0:
            return "Query not valid. Kindly Try again"
        else:
            return answer

    # functionality to exit the program
    while True:
        # if user input is one of the below keywords --> Then, exit the
program with a message.
        if query not in ['bye', 'good bye', 'take care']:
            lmnop = chatbot_answer(query)
            gett_sentences.remove(query)
            return lmnop
        else:
            return "Thanks for coming, Visit Again!!!"
            break
```

templates/index.html

```
<html>
    <head>
        <title>
            Chatbot
        </title>
    </head>
    <body>
```

```
        <h1>Ask us anything</h1>
        <form method="post" action="/">
            Input:
            <input type="text" name="query">
            <button type="submit">Submit</button>
            <p>Output: {{otp}}</p>
        </form>
    </body>
</html>
```