AI based Chatbot to answer FAQs

A Project Report

Submitted in the partial fulfillment for the award of the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE WITH SPECIALIZATION IN

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

Submitted by:

20BCS8812 Avanish Rai

20BCS6896 Priyanshu Singh

Under the Supervision of:

Mrs. Anita

CHANDIGARH UNIVERSITY, GHARUAN, MOHALI - 140413, PUNJAB

FEB-MAY, 2023

BONAFIDE CERTIFICATE

Certified that this project report **"AI Based ChatBot To Answer FAQ"** is the Bonafide work of **AVANISH RAI and PRIYANSHU SINGH** who carried out the project work under my supervision.

SIGNATURE OF THE HOD	**SIGNATURE OF THE SUPERVISOR**
MR. AMAN KOUSHIK	**MRS. ANITA**
(HEAD OF DEPARTMENT)	(SUPERVISOR)
CSE- AIML	(AIT-CSE)

Submitted for the project viva- voce examination held on

INTERNAL EXAMINER	**EXTERNAL EXAMINER**

Abstract

As a result of the rapid technological development and the development of the chatbot concept and the time and effort it can save. Many specialized frameworks have emerged to undertake chatbot creation and development. By relying on artificial intelligence, the chatbot has integrated machine learning within it, and it has become more comprehensive and wider for various technological fields. Therefore, we will create a chatbot for the University's Admission and Registration, the project aims to build a chatbot to facilitate the process of accessing information related to students' inquiries towards admissions, Registration and the university itself. As a conclusion, it lies in answering frequent and common questions by people and providing the answer to these questions at any time the person wants.

What is natural language processing (NLP)?

Natural language processing, which evolved from computational linguistics, uses methods from various disciplines, such as computer science, artificial intelligence, linguistics, and data science, to enable computers to understand human language in both written and verbal forms.

What is natural language understanding (NLU)?

Natural language understanding is a subset of natural language processing, which uses syntactic and semantic analysis of text and speech to determine the meaning of a sentence.

Table of Contents

ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany a successful completion of any task would be incomplete without the mention of people who made it possible. Success is the epitome of hard work and perseverance, but steadfast of all is encouraging guidance.

So, it is with gratitude that we acknowledge all those whose guidance and encouragement served as beacon of light and crowned our effort with success.

We consider it a privilege and honor to express our sincere gratitude to our internal guide Mrs. Anita, Professor, Department of Computer Science & Engineering for her valuable guidance throughout the tenure of this project work.

We would also like to thank all the faculty members who have always been very Co- operative and generous. Conclusively, we also thank all the non-teaching staff and all others who have done immense help directly or indirectly during our project.

Avanish Rai- 20BCS8812

Priyanshu Singh- 20BCS6869

1. INTRODUCTION

A text-based user interface is commonly used by chatbots, enabling the user to type commands and get text and textual content-to-speech responses. Chatbots are often stateful systems that remem?ber prior commands to deliver functionality. When chatbot technology is connected with major web services, an even bigger audience can use it safely. A chatbot is a user-interactive agent that uses simple language to converse with users. Several chatbot applications, such as Customer Service, contact centers, and so on, employ AI terminology to converse with the user. One of the primary aims of chatbots is to resemble an intelligent human and make it difficult for the recipient of the discussion to understand the significance of collaborating with various architecture and capabilities for his or her use. Chatbots employ machine learning to achieve artificial intelligence, allowing them to understand the user's query and offer a suitable response. To converse or engage with the user, the chatbot is built using the help of Jupiter notebook using Natural Language Processing (NLP). The proposed system could be an internet application that responds to the question posed via way of means of the user. Students only need to ask a question through the chatbot. Students can chat in any format they want; there is no set format that the user must adhere to. The replies to the user's request are relevant. The technology allows the user to query any college-related activities. To ask in person, the user does not need to travel to campus. The system will analyze the question and reply to the user. The system responds to the question as though the person replied. The system responds with an efficient graphical user interface, which implies that a real person speaks to the user. This application allows the user to inquire about college-related activities online. The user can search for college-related events such as placements, admissions, club information, departmental achievements and activities, and other cultural activities. This system keeps students up to date on.

1.1 Problem Definition

At the start of each academic semester, registration opens for those wishing to join the university in various disciplines, and telephone calls for admission and registration abound. This leads to an increase in the loads and work for the employees of the Deanship of Admission and Registration as a result of the constant pressure of those wishing to register and their families by flocking to the Deanship, so the employees are not able to answer the phone calls and social media. This often leads to many students who wish to register to be ignored.

1.2 Problem Overview

The goal of this project is to develop an AI-based chatbot that can answer frequently asked questions (FAQs) for a college or university. The chatbot will use natural language processing (NLP) and machine learning algorithms to understand student queries and provide accurate responses.

The chatbot will be integrated into the college's website or mobile app, allowing students to access the bot and get their queries resolved without having to interact with a human representative. The chatbot will answer various questions such as admission process, course details, fees structure, campus facilities, scholarships, and much more.

Key Features:

Natural Language Processing: The chatbot will be designed to understand and respond to student queries in natural language, allowing for a seamless and intuitive user experience.

Machine Learning: The chatbot will use machine learning algorithms to improve its understanding of student queries over time, making it more accurate and efficient.

Multi-Platform Integration: The chatbot will be integrated into various platforms such as the college's website, mobile app, and social media channels, providing students with a consistent and accessible support experience

1.3 Hardware Specification

- 8 GB RAM
- Processor - 1.5–4.5x
- Monitor – 15.6”
- Keyboard - 2.4GHz USB wireless receiver 1.4

1.4 Software Specification

- Python Compiler with required Libraries and Modules
    - · Language: Python
    - · Operating System: Windows 7/8/10/11
    - · Libraries like:
        - Tensorflow
        - nltk
        - tflearn
        - json
        - pickle
        - random
        - numpy
    - Software specs
        - PyCharm or Jupyter notebook
        - Conda Prompt (for running streamlit server)
        - Google Chrome (mostly latest version)
        - GPU environment

2. LITERATURE SURVEY

2.1 Existing System

The first and perhaps the easiest bots are legal-based chatbot, also known as bot-tree bots. These bots are very common, and many of us may be contacting one through Live Chat features, on e-commerce sites, or through a social media platform. communication with users. When communicating with users, written bots recognize keywords and deliver them in the right way to achieve their goals, such as information about the best deals currently, and more. Such a chatbot has a limited set of skills. However, you can use it for simple tasks like: Customer support agents who provide customers with automated responses. Engagement bots inform customers about special discounts. A legal-based chatbot is able to hold basic conversations based on the "if/when" concept. This chatbot does not understand the context or objectives. Human agents create a chat map with a flowchart, anticipate what the customer might ask, and plan how the chatbot should respond. We use the following logical steps and clear call-to-action buttons to create legal chatbot conversations. Companies create a legal chatbot to answer simple questions and often bring web visitors to a live agent to further the conversation. They are not designed to read and be intelligent over time. We can build a legal chatbot with very simple or complex rules. However, they cannot answer any questions other than the stated rules. Legal-based chatbot does not learn collaboratively and performs and operates only within the context in which they are trained.

2.2 Proposed System

Unlike rule-based chatbot ,AI-Powered Chatbot is a complex chatbot, usually powered by native language processing (NLP) and machine learning algorithms (ML). The user submits questions to the chatbot asking for information to move from the source to the location. User input is then compared to the character unit in the target file that includes the chatbot website. Powerful AI bots can respond to the user with pre-defined responses, and ML helps them learn from each user interaction. Public transport companies have a heavy responsibility: to provide affordable transportation options, maintain multiple transport networks, and keep passengers informed at all times when cost controls. Bot is an AI-enabled chatbot that can improve the performance of public transport networks based on ingenuity. Chatbot is ready to assist with route planning, provide traffic updates, handle custom alerts, and answer any frequently asked questions. Chatbot allows passengers to find schedules, arrival times, routes, and more within the chat.

3. PROBLEM FORMULATION

One of the primary problems facing colleges today is the high volume of frequently asked questions (FAQ) that are received from students regarding admissions, courses, fees, and other related topics. Handling these inquiries can be a time-consuming and resource-intensive process for college staff members, taking them away from other important tasks. An AI-based chatbot has the potential to address this problem by automating the handling of common inquiries. By using natural language processing (NLP) and machine learning algorithms, the chatbot can understand and respond to student queries in real-time, providing quick and accurate answers to their questions.

However, there are several challenges that need to be addressed in developing an effective chatbot for college enquiries. One challenge is ensuring that the chatbot is able to understand and respond to a wide variety of student inquiries, including those that may be phrased in different ways or use different terminology.

Another challenge is ensuring that the chatbot is able to escalate inquiries to human staff members when necessary. This requires the development of a robust escalation process that ensures students receive the appropriate level of support.

4. OBJECTIVES

AI-powered chatbot that can efficiently and accurately answer frequently asked questions (FAQ) related to college enquiries. The chatbot will be designed to use natural language processing (NLP) and machine learning algorithms to understand and respond to student queries regarding admissions, courses, fees, and other related topics.

The chatbot will be integrated into the college's website and other digital channels, providing students with 24/7 access to information. By automating the handling of common inquiries, the chatbot will help reduce the workload on college staff, allowing them to focus on more complex issues. Additionally, the chatbot will continuously learn from student interactions, improving its performance over time. Our goal is to improve the student experience by providing a fast and convenient way for students to find the information they need. By providing quick and accurate answers to student queries, we aim to reduce student frustration and increase their satisfaction. The chatbot will also be able to escalate inquiries to human staff members when necessary, ensuring that students receive the appropriate level of support.

Ultimately, the implementation of this AI-powered chatbot will enable the college to provide a higher level of service to students while reducing operational costs. The chatbot will help improve student retention rates and free up staff time for more meaningful interactions with students.

5. METHODOLOGY

A. UI development

Bot is an open-source application that provides a user-friendly interface to deal with the content of government bus services. It supports all three : (A) Modify knowledge base, (B) Data Processing, (C) Model Testing.

B. Intent Classification

(1) Dataset Preparation

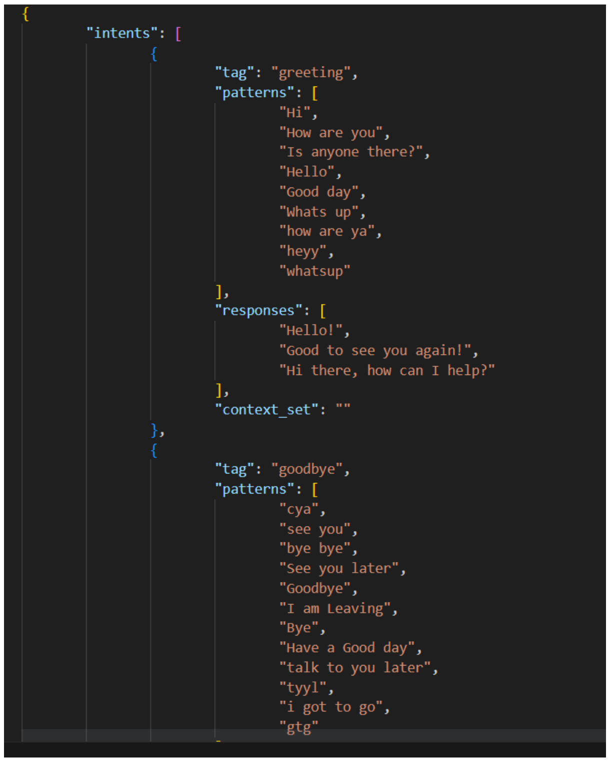
• <u>Import Dataset</u>

In this module the admin uploads a dataset (CSV) file. This will be used to train your chatbot

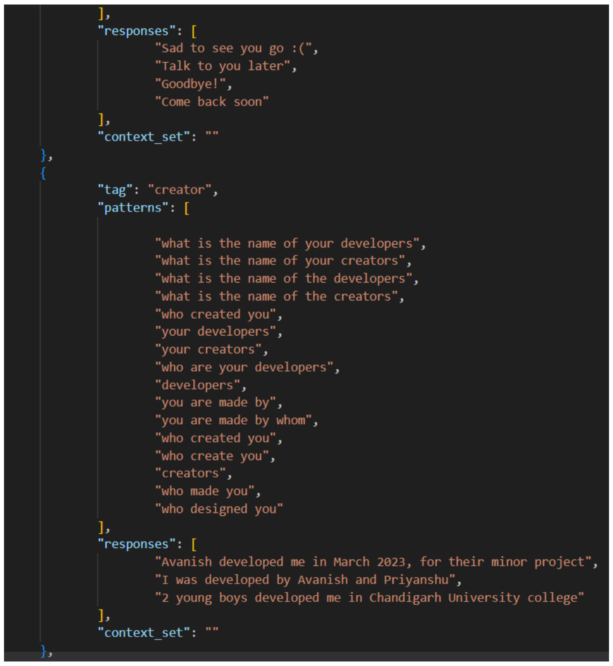
• <u>Read Dataset</u>

The chatbot reads dataset to output the purpose for the user intent •<u>Explore Dataset</u>

EDA Data visualization tool that brings the entirety of data together into a striking and easy-to-follow view.



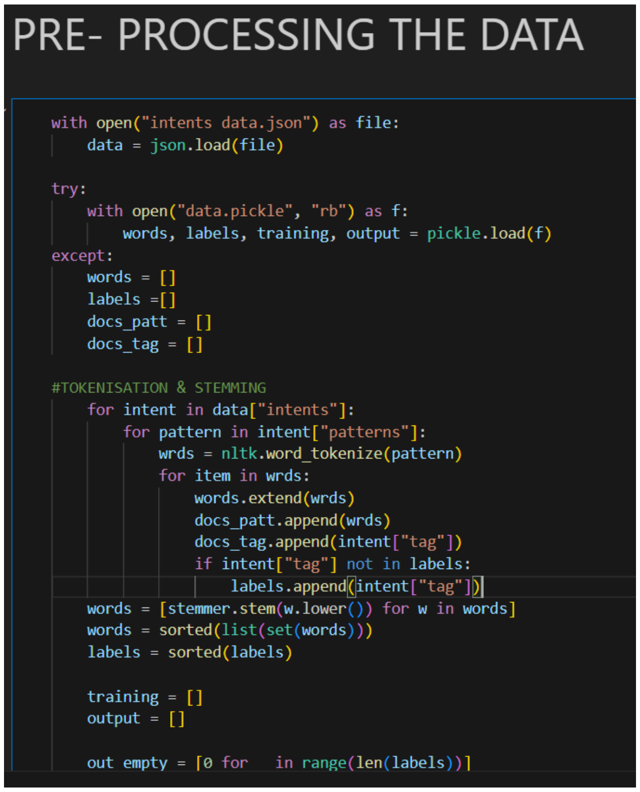
```
{
        "intents": [
                {
                        "tag": "greeting",
                        "patterns": [
                                "Hi",
                                "How are you",
                                "Is anyone there?",
                                "Hello",
                                "Good day",
                                "Whats up",
                                "how are ya",
                                "heyy",
                                "whatsup"
                        ],
                        "responses": [
                                "Hello!",
                                "Good to see you again!",
                                "Hi there, how can I help?"
                        ],
                        "context_set": ""
                },
                {
                        "tag": "goodbye",
                        "patterns": [
                                "cya",
                                "see you",
                                "bye bye",
                                "See you later",
                                "Goodbye",
                                "I am Leaving",
                                "Bye",
                                "Have a Good day",
                                "talk to you later",
                                "tyyl",
                                "i got to go",
                                "gtg"
```



```
                        ],
                        "responses": [
                                "Sad to see you go :(",
                                "Talk to you later",
                                "Goodbye!",
                                "Come back soon"
                        ],
                        "context_set": ""
                },
                {
                        "tag": "creator",
                        "patterns": [

                                "what is the name of your developers",
                                "what is the name of your creators",
                                "what is the name of the developers",
                                "what is the name of the creators",
                                "who created you",
                                "your developers",
                                "your creators",
                                "who are your developers",
                                "developers",
                                "you are made by",
                                "you are made by whom",
                                "who created you",
                                "who create you",
                                "creators",
                                "who made you",
                                "who designed you"
                        ],
                        "responses": [
                                "Avanish developed me in March 2023, for their minor project",
                                "I was developed by Avanish and Priyanshu",
                                "2 young boys developed me in Chandigarh University college"
                        ],
                        "context_set": ""
                },
```

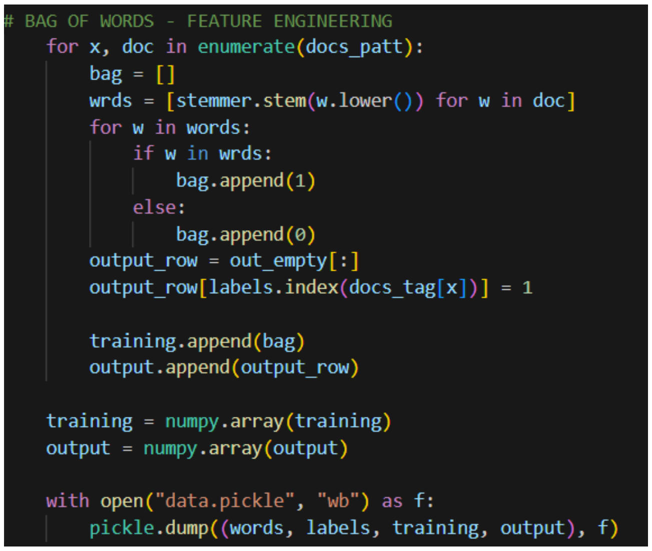
(2) Data Pre-processing

The data pre-processing phase consists of natural language-based steps that standardize the text and prepare it for analysis.



PRE- PROCESSING THE DATA

```
with open("intents data.json") as file:
    data = json.load(file)

try:
    with open("data.pickle", "rb") as f:
        words, labels, training, output = pickle.load(f)
except:
    words = []
    labels =[]
    docs_patt = []
    docs_tag = []

#TOKENISATION & STEMMING
    for intent in data["intents"]:
        for pattern in intent["patterns"]:
            wrds = nltk.word_tokenize(pattern)
            for item in wrds:
                words.extend(wrds)
                docs_patt.append(wrds)
                docs_tag.append(intent["tag"])
                if intent["tag"] not in labels:
                    labels.append(intent["tag"])
    words = [stemmer.stem(w.lower()) for w in words]
    words = sorted(list(set(words)))
    labels = sorted(labels)

    training = []
    output = []

    out_empty = [0 for _ in range(len(labels))]
```



```
# BAG OF WORDS - FEATURE ENGINEERING
    for x, doc in enumerate(docs_patt):
        bag = []
        wrds = [stemmer.stem(w.lower()) for w in doc]
        for w in words:
            if w in wrds:
                bag.append(1)
            else:
                bag.append(0)
        output_row = out_empty[:]
        output_row[labels.index(docs_tag[x])] = 1

        training.append(bag)
        output.append(output_row)

    training = numpy.array(training)
    output = numpy.array(output)

    with open("data.pickle", "wb") as f:
        pickle.dump((words, labels, training, output), f)
```

- **Tokenization**

Breaking up the original text into component pieces is the tokenization step in natural language processing. There are predefined rules for tokenization of the documents into words. The tokenization step is performed in Python by using the Spacy library. Convert a sentence [i.e. a collection of words] into single words. Sentence Tokens

- **Normalization**

These are the steps needed for translating text from human language (English) to machine-readable format for further processing. The process starts with: changing all alphabets to lower or upper case, expanding abbreviations, excluding numbers or changing those to words ,removing white spaces, punctuation, inflection marks, and other circumflexes.

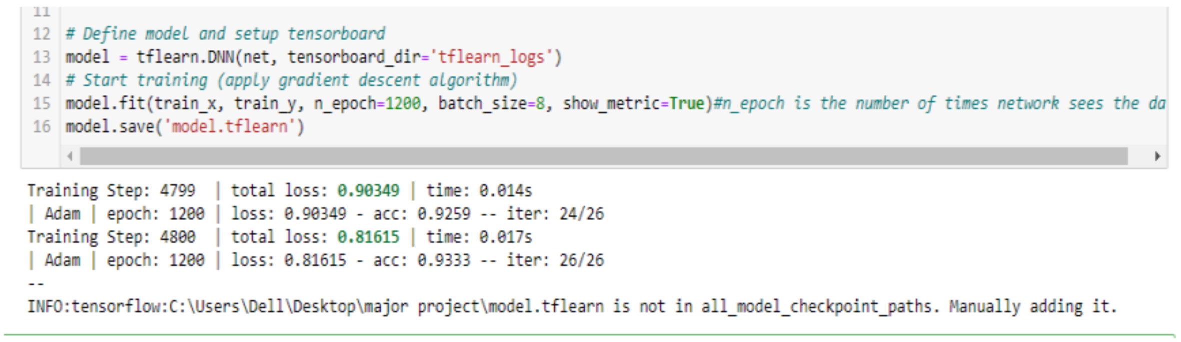
- **Stemming**

It is a process to find similarities between words with the same root words. This will help us to reduce the bag of words by associating similar words with their corresponding root words.

(3) Building and Training the neural network model

The data related to CHANDIGARH UNIVERSITY College is very diverse. Therefore, there is a need for a neural net?work which uses many layers of nodes to derive high end functionalities. Therefore, deep neural networks (DNN) fit the best in this use case scenario. The neural network is also very efficient even if the inflow of data is exponentially increasing. The college has numerous day to day activities which in turn lead to huge data which the neural network can accommodate. The bag of words model has converted the textual data to a document vector containing 0's and 1's. The length of the vector will be equal to vocabulary size. We initialize 1 when a word from the curent pattern is found in the given position X (pattern converted into array [0,1,0,1,0, 1...,0]), Y (training data) (intents converted to an array [0,0,1,0,0,0,0], there will be only single 1 for intents array as it responds to the match found). Layers of interconnected nodes make up a Neural Net?work. In Neural Networks, there are two key components. The Neuron and the Network are their names. A neuron is a data-storage node. The network is a mathematical relationship between nodes that is achieved through the use of functions. Different Layers are also present in the network. The Input Data Layer, Multiple Intermediate Hidden Data Layers, and Output Data Layers are the three layers. We have constructed a deep neural network. The DNN is constructed with 2 hidden layers for mapping comforts. Optimization function: When you train a model you try to solve the optimization problem. While you are trying to optimize the weights. Each connection between the neuron has an arbitrary weight assigned to it.

During the training, these weights will be constantly updated and attempted to reach optimal values. In terms of how they are being optimized it depends on the optimization algorithm. In our project, we have considered the optimization algorithm to be GD (Gradient De?scent). The no of epochs we used for our model is 1200. We have obtained it from trial and error method?ology where we tested for a wide number of ranges and this range was giving desirable results. We achieved 93 percent accuracy on our model after training it for 1200 epochs.



```
# Define model and setup tensorboard
model = tflearn.DNN(net, tensorboard_dir='tflearn_logs')
# Start training (apply gradient descent algorithm)
model.fit(train_x, train_y, n_epoch=1200, batch_size=8, show_metric=True)#n_epoch is the number of times network sees the da
model.save('model.tflearn')
```

```
Training Step: 4799  | total loss: 0.90349 | time: 0.014s
| Adam | epoch: 1200 | loss: 0.90349 - acc: 0.9259 -- iter: 24/26
Training Step: 4800  | total loss: 0.81615 | time: 0.017s
| Adam | epoch: 1200 | loss: 0.81615 - acc: 0.9333 -- iter: 26/26
--
INFO:tensorflow:C:\Users\Dell\Desktop\major project\model.tflearn is not in all_model_checkpoint_paths. Manually adding it.
```

Fig. 3. Training Accuracy and Loss

(4) Feature Extraction

After eliminating irrelevant information, the elaborated list of words is converted into numbers. The TF-IDF method is applied to accomplish this task. Term Frequency is several occurrences of a word in a document, and IDF is the ratio of a total number of documents and the number of documents containing the term. A popular and straightforward method of feature extraction with text data is called the bag-of-words model of text. A bag-of-words model for short, is a way of extracting features from the text for use in modelling, such as machine learning algorithms. A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things (1) A vocabulary of known words, (2) A measure of the presence of known words. We extract features on the basis of Equations Here tf represents term frequency and df represents document frequency. Eq. 1-4 Feature extraction in DL with the context of words is also essential. The technique used for this purpose is word2vec neural network-based algorithm. Equation 5 given below shows how word2vec manages the word-context with the help of probability measures. The D represents the pair-wise illustration of a set of words, and (w; c) is the word-context pair drawn from the large set D. Eq.5 The multi-word context is also a variant of word2vec, as shown in Equation 6. The variable-length context is also controlled by the given below mathematics. Eq.6

(5) Speech recognition

Speech recognition is one of the most common features that find its application almost everywhere. It makes the environment more user-friendly and eases the process of querying at the user's end. Our project has utilized the speech recognizer and pyttsx3 modules for building speech recognition functions.

a)Input from speech

The system's inbuilt microphone is used to take the speech input from the user and that input is converted to a wave file (audio file). That audio file is being analyzed by the speech recognizer module of python and generates the textual data out of the audio. Now that textual data is being sent for pre-processing and bag of words algorithm is applied. Finally sent to DNN model for the response.

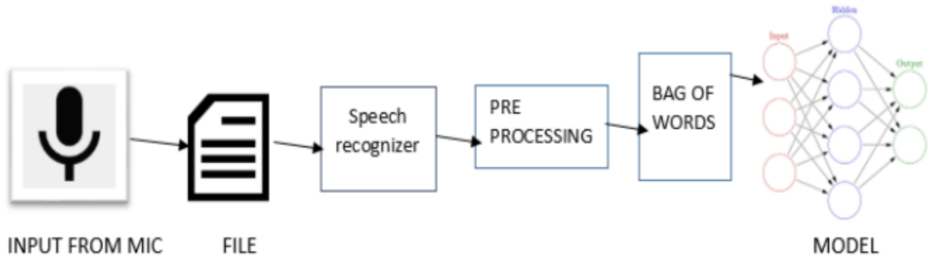


Fig. 4. Flowchart for Speech Recognition Input

b)Output as speech

The pyttsx3 initializes an engine that is responsible for loading speech, changing the pitch of the speech (voices female or male). After the engine is initialized, it is run and it gives you a simple speech output of the response that responds to the given query.

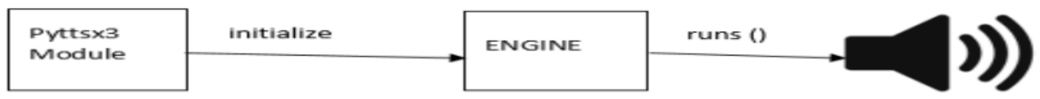
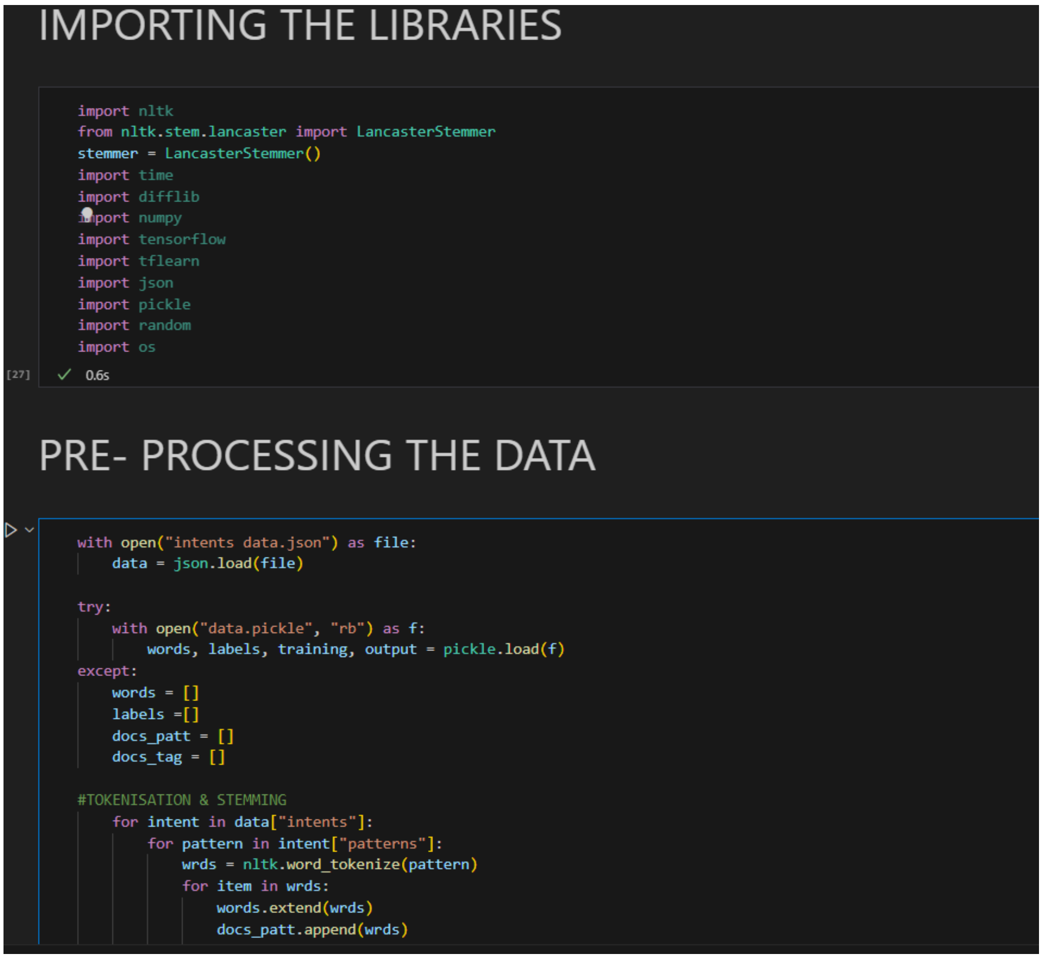


Fig. 5. Flowchart for Speech Recognition Output

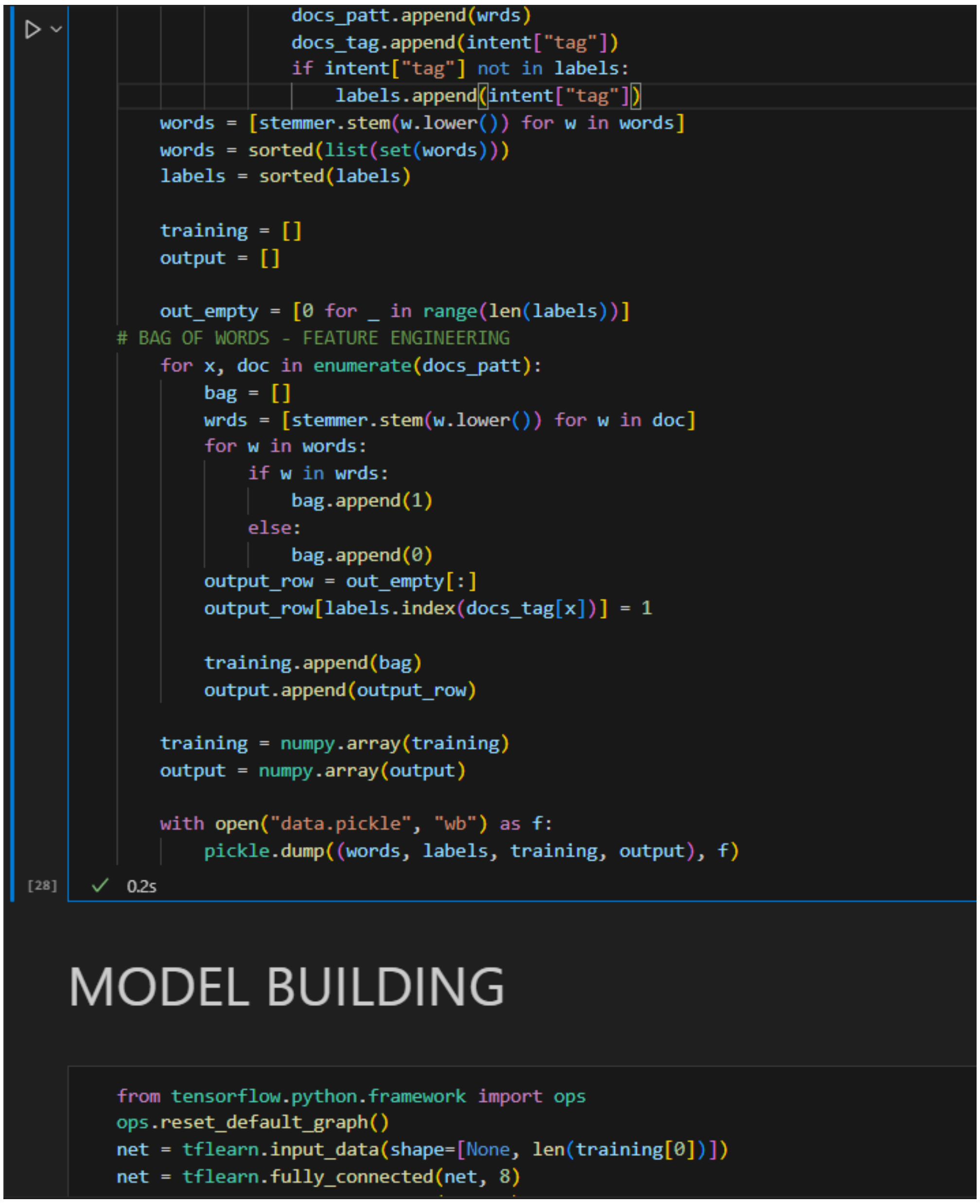
6.SNAPSHOTS OF IMPLEMENTATION


IMPORTING THE LIBRARIES

```
import nltk
from nltk.stem.lancaster import LancasterStemmer
stemmer = LancasterStemmer()
import time
import difflib
import numpy
import tensorflow
import tflearn
import json
import pickle
import random
import os
```
[27] ✓ 0.6s

PRE- PROCESSING THE DATA

```
with open("intents data.json") as file:
    data = json.load(file)

try:
    with open("data.pickle", "rb") as f:
        words, labels, training, output = pickle.load(f)
except:
    words = []
    labels =[]
    docs_patt = []
    docs_tag = []

#TOKENISATION & STEMMING
    for intent in data["intents"]:
        for pattern in intent["patterns"]:
            wrds = nltk.word_tokenize(pattern)
            for item in wrds:
                words.extend(wrds)
                docs_patt.append(wrds)
```

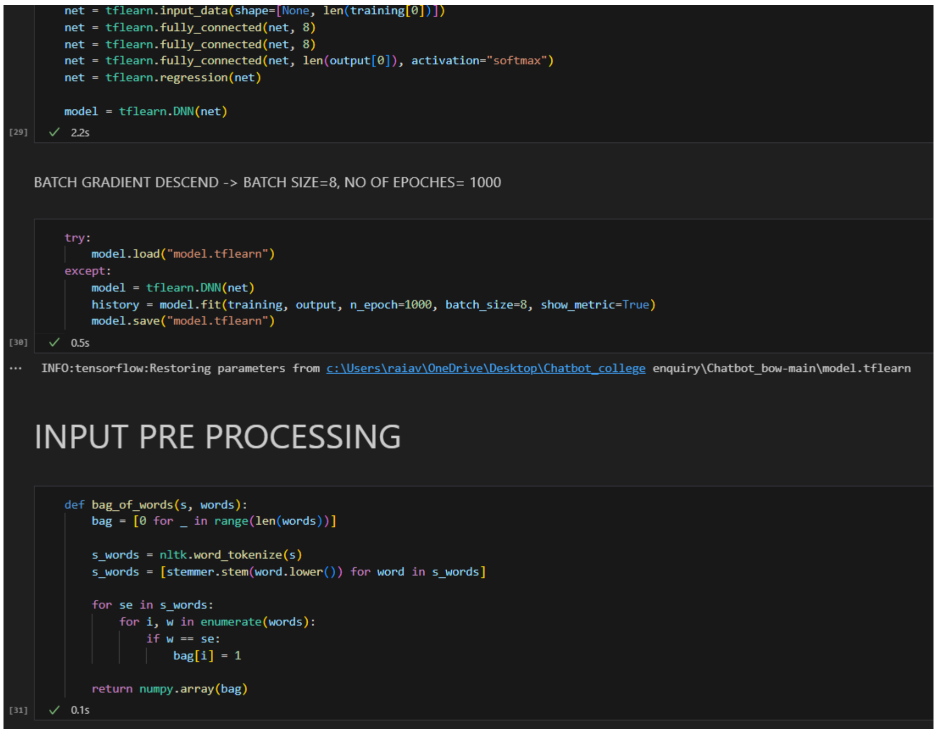


```
                docs_patt.append(wrds)
                docs_tag.append(intent["tag"])
                if intent["tag"] not in labels:
                    labels.append(intent["tag"])
    words = [stemmer.stem(w.lower()) for w in words]
    words = sorted(list(set(words)))
    labels = sorted(labels)

    training = []
    output = []

    out_empty = [0 for _ in range(len(labels))]
# BAG OF WORDS - FEATURE ENGINEERING
    for x, doc in enumerate(docs_patt):
        bag = []
        wrds = [stemmer.stem(w.lower()) for w in doc]
        for w in words:
            if w in wrds:
                bag.append(1)
            else:
                bag.append(0)
        output_row = out_empty[:]
        output_row[labels.index(docs_tag[x])] = 1

        training.append(bag)
        output.append(output_row)

    training = numpy.array(training)
    output = numpy.array(output)

    with open("data.pickle", "wb") as f:
        pickle.dump((words, labels, training, output), f)
```

[28] ✓ 0.2s

MODEL BUILDING

```
from tensorflow.python.framework import ops
ops.reset_default_graph()
net = tflearn.input_data(shape=[None, len(training[0])])
net = tflearn.fully_connected(net, 8)
```



```
net = tflearn.input_data(shape=[None, len(training[0])])
net = tflearn.fully_connected(net, 8)
net = tflearn.fully_connected(net, 8)
net = tflearn.fully_connected(net, len(output[0]), activation="softmax")
net = tflearn.regression(net)

model = tflearn.DNN(net)
```
[29] ✓ 2.2s

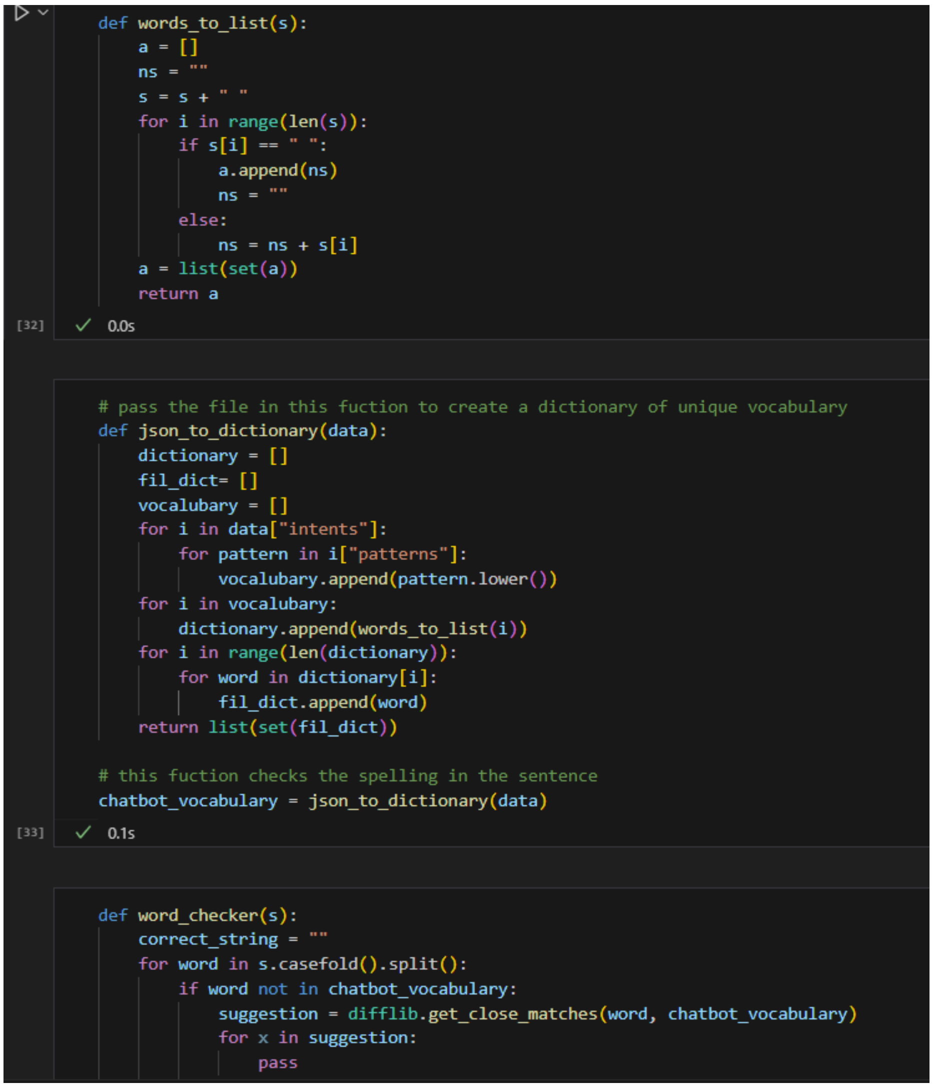
BATCH GRADIENT DESCEND -> BATCH SIZE=8, NO OF EPOCHES= 1000

```
try:
    model.load("model.tflearn")
except:
    model = tflearn.DNN(net)
    history = model.fit(training, output, n_epoch=1000, batch_size=8, show_metric=True)
    model.save("model.tflearn")
```
[30] ✓ 0.5s

```
INFO:tensorflow:Restoring parameters from c:\Users\rajav\OneDrive\Desktop\Chatbot_college enquiry\Chatbot_bow-main\model.tflearn
```

INPUT PRE PROCESSING

```
def bag_of_words(s, words):
    bag = [0 for _ in range(len(words))]

    s_words = nltk.word_tokenize(s)
    s_words = [stemmer.stem(word.lower()) for word in s_words]

    for se in s_words:
        for i, w in enumerate(words):
            if w == se:
                bag[i] = 1

    return numpy.array(bag)
```
[31] ✓ 0.1s



```
def words_to_list(s):
    a = []
    ns = ""
    s = s + " "
    for i in range(len(s)):
        if s[i] == " ":
            a.append(ns)
            ns = ""
        else:
            ns = ns + s[i]
    a = list(set(a))
    return a
```
[32] ✓ 0.0s

```
# pass the file in this fuction to create a dictionary of unique vocabulary
def json_to_dictionary(data):
    dictionary = []
    fil_dict= []
    vocalubary = []
    for i in data["intents"]:
        for pattern in i["patterns"]:
            vocalubary.append(pattern.lower())
    for i in vocalubary:
        dictionary.append(words_to_list(i))
    for i in range(len(dictionary)):
        for word in dictionary[i]:
            fil_dict.append(word)
    return list(set(fil_dict))

# this fuction checks the spelling in the sentence
chatbot_vocabulary = json_to_dictionary(data)
```
[33] ✓ 0.1s

```
def word_checker(s):
    correct_string = ""
    for word in s.casefold().split():
        if word not in chatbot_vocabulary:
            suggestion = difflib.get_close_matches(word, chatbot_vocabulary)
            for x in suggestion:
                pass
```

This chapter will provide information about the subject system and tools used for evaluation of proposed method. This will include the requirements of the project and the most suitable solution for it.

7.CONCLUSION

The main objective of the college inquiry Chatbot project is to employ algorithms to read user ques?tions and understand user messages. The internet offers a variety of methods to obtain information and has dramatically altered how people communicate. There is a need for technology that address?es the queries of users 24/7. One such technology is a chatbot. A chatbot is just a man-made pro?gram that easily connects with users to assist and solve their queries. The services that a chatbot can deliver are quite distinct and diverse as they could assist from daily essential queries to big in?dustrial needs. Chatbots can reach a large range of audience and be more practical than humans. We can incorporate speech-based inquiries and replies into our project in the future. The applica?tion may be integrated into the college's website. Sentiment analysis analyses the user's input text or speech to assist understand their feelings and state of mind. Chatbots can use this data to better prepare discussions and give the appropriate replies. There is room for more data to be added in the future, making the chatbot more authentic and accurate.

8. TENTATIVE CHAPTER PLAN FOR THE PROPOSED WORK

While building this intelligent college enquiry bot system, we have followed the below mentioned steps.

A. Building front-end

B. Building back-end

C. Creating the proposed chatbot model

D. Connecting the front-end to back-end

The below is a brief discussion on each of the steps followed in the mentioned methodology:

A. Building Front End

We used ReactJS to build the front-end and Python as a back-end. The ML-Model for chatbot was build and incorporated into the backend. Coming to the front-end the components mentioned in the hierarchy below were used to build the UI for the chatbot. Moreover, the UI is also

responsive for the different screen sizes.

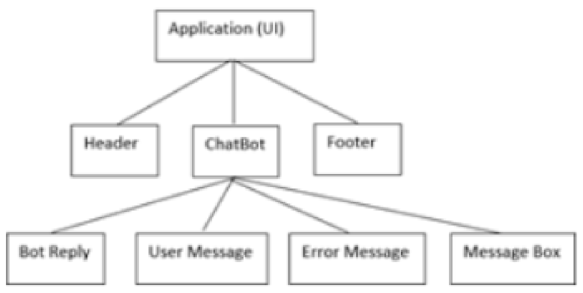


Fig.1 Frontend Modules Hierarchy

As shown in the above “Fig. 1”, there are three main components for the React App: a) Header Component – which displays the college’s name and logo as a header b) Footer Component – which displays a footer for the webpage which usually has a copyright c) Chatbot Component – which is the main component for this project. It is further subdivided into 4 other components as mentioned below. This component is responsible for the rendering of the chatbot’s UI for the end-users. Its sub-components are: i. UserMessage Component – which is responsible for displayingthe user queries that are entered by the user in the message box provided after sending to the chatbot ii. BotReplyComponent – which is responsible for displaying the replies of the chatbot for the queries made by the user after getting a response from the backend iii. MessageBox Component – which provides an input box where the user can write their queries and send the message to the chatbot by clicking a send button provided by the side of it. iv. ErrorMessage Component – whichdisplays the various error messages captured in different circumstances and notifies the user about it by error text

B. Building Back -end Then speaking of the backend, we have written several methods for several specific functionalities. The below are a few modules and their descriptions we had used in our backend.

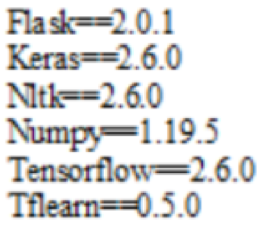
a) Intents Module – which is the data file containing all the predefined patterns and their corresponding responses. It is stored in the form of a JSON (JavaScript Object Notation) file.

b) Preprocessing Module – which consists of the preprocessing methods in order to clean the sentences and perform NLP techniques like tokenization and lemmatization.

c) Chatbot Training Module – which is responsible for using preprocessing module methods to clean the data and split the data into train and test sets. It also involves creating, training and saving the deep learning model

d) Model Related Module – which is basically a wrapper to help the chatbot training module to deal with the deep learning model and perform operations on it. In it there is contained the information regarding the model and also the weights of the neurons.

e) Chatbot Module – It can be called the main module that is responsible for connecting the chatbot to the Flask app. It serves as a backend which receives user query from UI and sends back the bot's responses through REST APIs. C. Creating the proposed chatbot model It is basically a retrieval-based chatbot making use of technologies like NLTK, Keras, Python, etc. In order to compile the model we used the stochastic gradient descent (SGD) along with nesterov accelerated gradient as an optimizer and also it gave good results for this model. The below are the five steps followed in order to create a chatbot in Python: a) Importing and loading libraries The first and foremost step in the process of creating the chatbot is importing the necessary packages and initializing necessary variables. Below mentioned are a few important required packages:



```
Flask==2.0.1
Keras==2.6.0
Nltk==2.6.0
Numpy==1.19.5
Tensorflow==2.6.0
Tflearn==0.5.0
```

b) Preprocess data

Pre-processing the data is yet an important step for an accurate response from the chatbot. When we are working with textdata, we should use certain pre-processing techniques and perform them on the data before building a machine earning or deep learning model. A few pre-processing techniques used are the available list of responses corresponding to the input pattern. Also here we import "words.pkl" and "Classes.pkl" pickle files that were created during the model training. In the below "Fig. 3", there is the architecture of the system design.

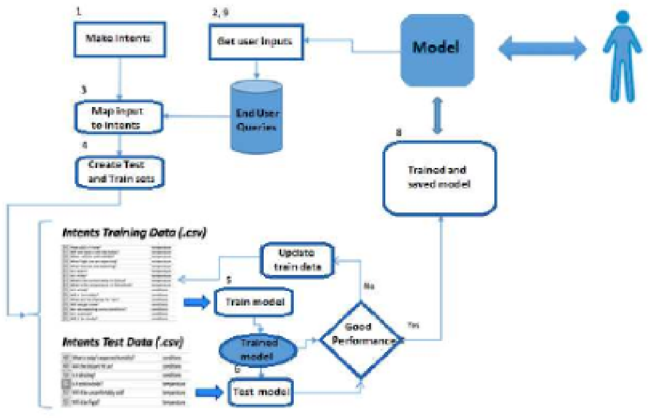
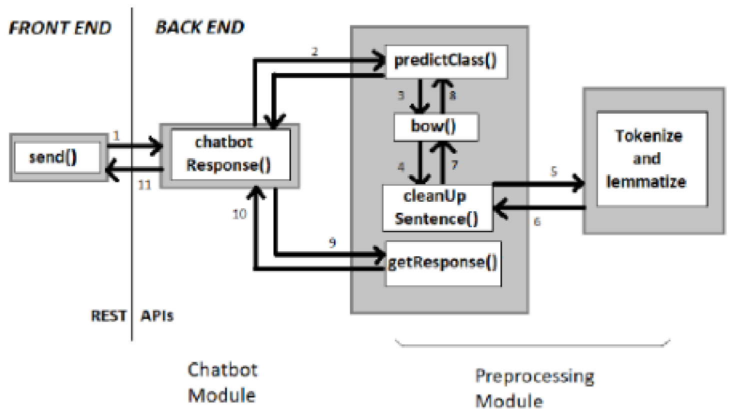


Fig. 3 System architecture

D. Connecting front-end to the back-end

This is again a crucial part of the chatbot application. Whenever a user enters a query through the message box provided in the UI, we send a POST request to a backend API. We provide the user query as a payload in the header contents of the POST request. At the backend, whenever we receive a user query at the endpoint through a POST API request, we first extract the query from the payload in the header part of the request received. Then, with the help of a trained deep learning model, we get the response for the corresponding user query. This process involves some pre-processing of the received payload and then predicting the class label. Finally, we get an appropriate response, which is then appended to a chats list at the backend. Then, the chats list is sent back to the frontend as a JSON object. At last, in the frontend, we try to capture any possible errors or exceptions if any, and send the received chats list to the UI in order to update the chats. This is how the frontend and backend are connected. The workflow of the chatbot is depicted in the below figure.



CHAPTER 2: FUTURE SCOPE

The goal of our proposed system is to help the students to get information about their college activities and to post their admission-related queries on the go from anywhere, even outside the college. Another main motive is to reduce the workload on the college staff and reduce the response time for a user queries. For this, we have proposed a web-based chatbot system with the combination of Deep Leaning and NLP based techniques. It had almost 99% accuracy score in giving appropriate responses to the users for their queries. The performance as well as accuracy is very considerable for our chatbot system along with very small response time. In future, using AI/ML/DL, chatbots can provide students with learning material in an interactive manner on any topic, help them learn quicker through visuals, speech or video and evaluate their responses to gauge their learning. They can be used for – a. Admission process – To assist through documentation guidelines, enrollment procedures, campus info, generate more inquiries, etc. b. Recommending courses – A personalized assistance to students on courses offered and resolving queries on curriculum, credits, internship opportunities, etc. c. On boarding students – FAQ resolution on orientations, campus visits,events, etc. d. Student/Faculty Support point – to offload the staff's work of solving repetitive queries. Student/Faculty Feedback - Collect students' feedback through conversations on learning sessions, campus environment, course improvement, etc.

CHAPTER 4: METHODOLOGIES

A chatbot is software that responds

to user questions and provides information from a knowledge base. The purpose

of this project is to create a chatbot for CHANDIGARH UNIVERSITY that will answer queries raised

about fests, departmental activities, events, clubs, infrastructure, placement data,

admission procedure, and others. The proposed methodology consists of a chatbot

built using Deep Neural Networks and speech recognition capabilities. The information is delivered in both speech and text modes using the proposed methodology.

Data is collected and formatted in JSON format initially. The prepared data is preprocessed and then the bag of words algorithm is applied to it. The bag of words

algorithm is most influential method for object categorization. The key aspect of

using this algorithm is for converting the word vector to a numerical data set for

machine to do a deeper analysis. A deep neural network is created using tensor

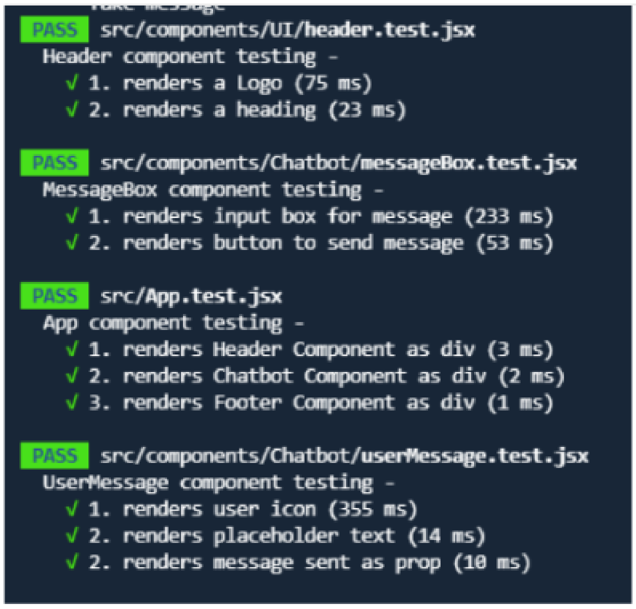
flow API, and the speech recognition function is defined for the input query and

output response. Finally, chatbot function is defined and utilized for generating

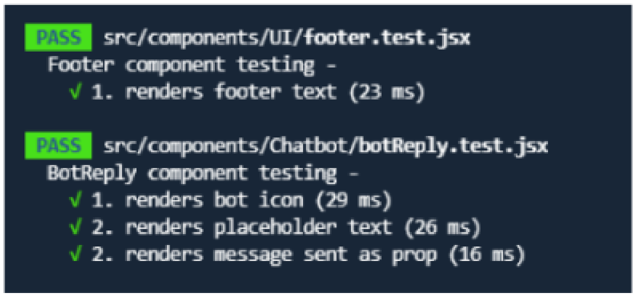
responses for any given query.

Keywords: Bag of words, Deep Neural Networks, Batch gradient descent algorithm,

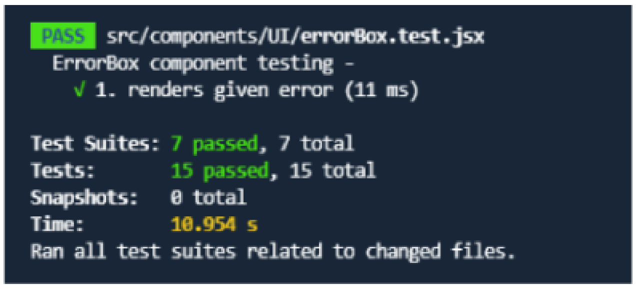
NLP, JSON.



```
 PASS  src/components/UI/header.test.jsx
  Header component testing -
    √ 1. renders a Logo (75 ms)
    √ 2. renders a heading (23 ms)

 PASS  src/components/Chatbot/messageBox.test.jsx
  MessageBox component testing -
    √ 1. renders input box for message (233 ms)
    √ 2. renders button to send message (53 ms)

 PASS  src/App.test.jsx
  App component testing -
    √ 1. renders Header Component as div (3 ms)
    √ 2. renders Chatbot Component as div (2 ms)
    √ 3. renders Footer Component as div (1 ms)

 PASS  src/components/Chatbot/userMessage.test.jsx
  UserMessage component testing -
    √ 1. renders user icon (355 ms)
    √ 2. renders placeholder text (14 ms)
    √ 2. renders message sent as prop (10 ms)
```



```
 PASS  src/components/UI/footer.test.jsx
  Footer component testing -
    √ 1. renders footer text (23 ms)

 PASS  src/components/Chatbot/botReply.test.jsx
  BotReply component testing -
    √ 1. renders bot icon (29 ms)
    √ 2. renders placeholder text (26 ms)
    √ 2. renders message sent as prop (16 ms)
```

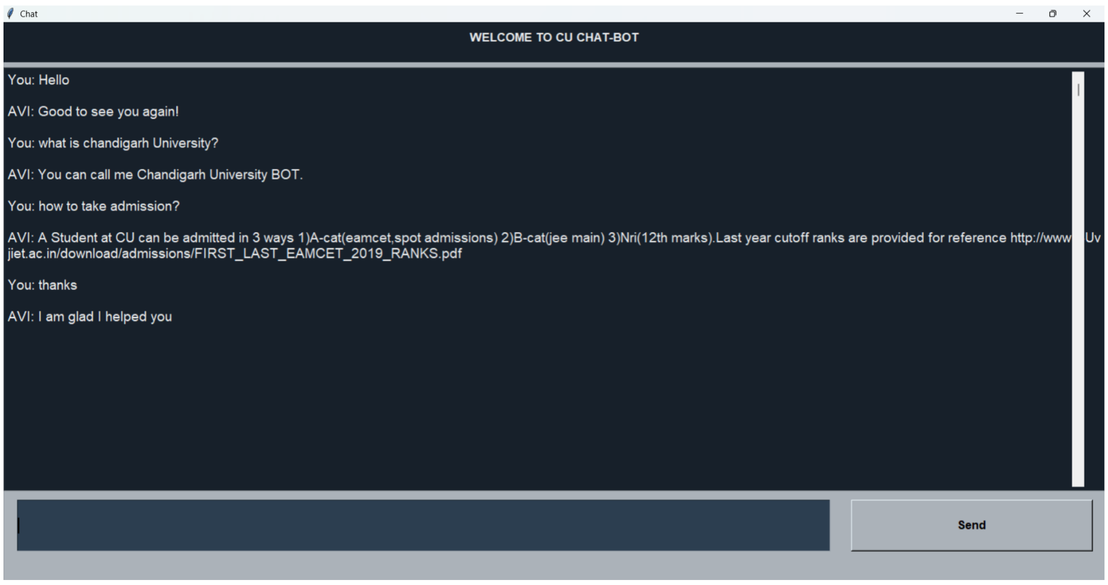


```
 PASS  src/components/UI/errorBox.test.jsx
  ErrorBox component testing -
    √ 1. renders given error (11 ms)

Test Suites: 7 passed, 7 total
Tests:       15 passed, 15 total
Snapshots:   0 total
Time:        10.954 s
Ran all test suites related to changed files.
```

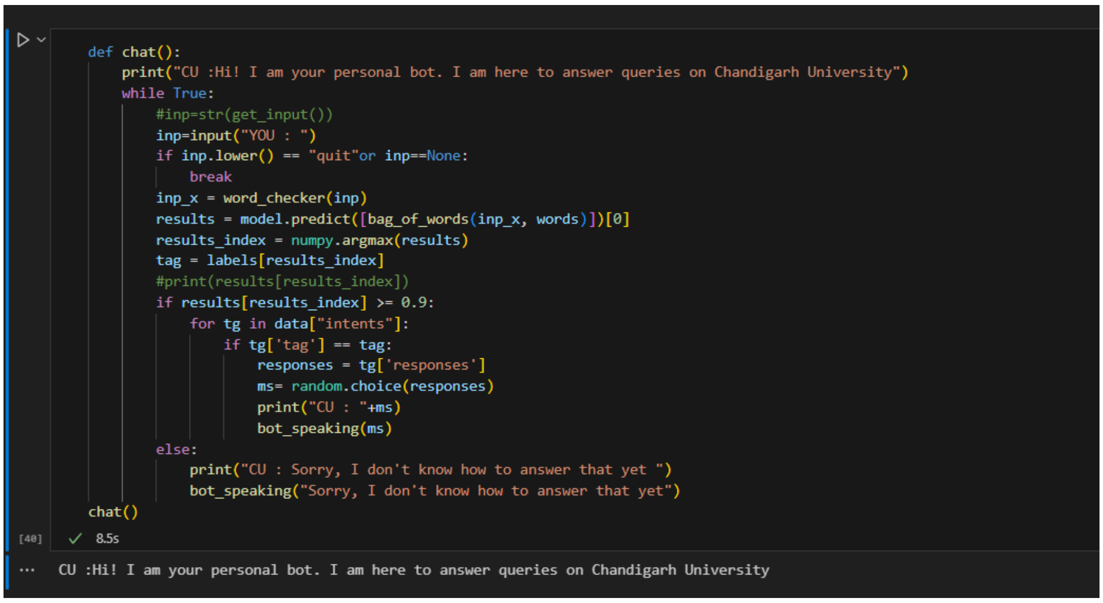
testing resport

CHAPTER 5: RESULT

It takes input words that are also pre-processed. The bag of words algorithm is applied and is matched with the document vector. If the result index is 90% matched it returns the response. Therefore, if the model found a 90% match from document vector it results in response pointing it. The main purpose of this project is to create a chatbot for CU College that will answer que?ries raised about fests, departmental activities, events, clubs, infrastructure, placement data, admis?sion procedure, and other topics. Therefore, our chatbot responds to the queries related to all of them. Typically, this chatbot is made for the outsiders to know about CU efficiently.



```
You: Hello

AVI: Good to see you again!

You: what is chandigarh University?

AVI: You can call me Chandigarh University BOT.

You: how to take admission?

AVI: A Student at CU can be admitted in 3 ways 1)A-cat(eamcet,spot admissions) 2)B-cat(jee main) 3)Nri(12th marks).Last year cutoff ranks are provided for reference http://www.Uv
jiet.ac.in/download/admissions/FIRST_LAST_EAMCET_2019_RANKS.pdf

You: thanks

AVI: I am glad I helped you
```



```
def chat():
    print("CU :Hi! I am your personal bot. I am here to answer queries on Chandigarh University")
    while True:
        #inp=str(get_input())
        inp=input("YOU : ")
        if inp.lower() == "quit"or inp==None:
            break
        inp_x = word_checker(inp)
        results = model.predict([bag_of_words(inp_x, words)])[0]
        results_index = numpy.argmax(results)
        tag = labels[results_index]
        #print(results[results_index])
        if results[results_index] >= 0.9:
            for tg in data["intents"]:
                if tg['tag'] == tag:
                    responses = tg['responses']
                    ms= random.choice(responses)
                    print("CU : "+ms)
                    bot_speaking(ms)
        else:
            print("CU : Sorry, I don't know how to answer that yet ")
            bot_speaking("Sorry, I don't know how to answer that yet")
chat()
```

[40] ✓ 8.5s

```
CU :Hi! I am your personal bot. I am here to answer queries on Chandigarh University
```

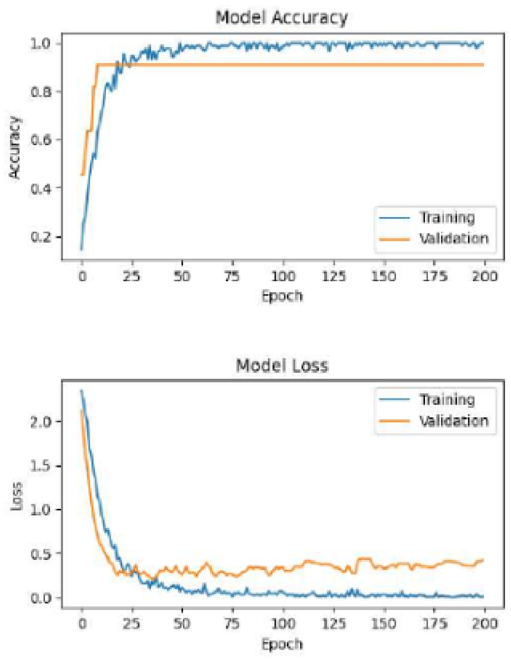
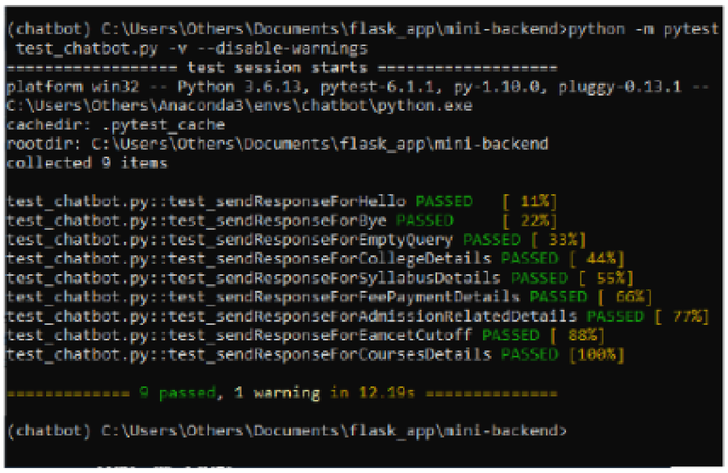


Fig. 11 Model Accuracy & Loss



```
(chatbot) C:\Users\Others\Documents\flask_app\mini-backend>python -m pytest
 test_chatbot.py -v --disable-warnings
================== test session starts ===================
platform win32 -- Python 3.6.13, pytest-6.1.1, py-1.10.0, pluggy-0.13.1 --
C:\Users\Others\Anaconda3\envs\chatbot\python.exe
cachedir: .pytest_cache
rootdir: C:\Users\Others\Documents\flask_app\mini-backend
collected 9 items

test_chatbot.py::test_sendResponseForHello PASSED   [ 11%]
test_chatbot.py::test_sendResponseForBye PASSED     [ 22%]
test_chatbot.py::test_sendResponseForEmptyQuery PASSED [ 33%]
test_chatbot.py::test_sendResponseForCollegeDetails PASSED [ 44%]
test_chatbot.py::test_sendResponseForSyllabusDetails PASSED [ 55%]
test_chatbot.py::test_sendResponseForFeePaymentDetails PASSED [ 66%]
test_chatbot.py::test_sendResponseForAdmissionRelatedDetails PASSED [ 77%]
test_chatbot.py::test_sendResponseForEamcetCutoff PASSED [ 88%]
test_chatbot.py::test_sendResponseForCoursesDetails PASSED [100%]

============= 9 passed, 1 warning in 12.19s ==============

(chatbot) C:\Users\Others\Documents\flask_app\mini-backend>
```

Fig. 10 Flask App Testing Report

CHAPTER 2: FUTURE SCOPE

The goal of our proposed system is to help the students to get information about their college activities and to post their admission-related queries on the go from anywhere, even outside the college. Another main motive is to reduce the workload on the college staff and reduce the response time for a user queries. For this, we have proposed a web-based chatbot system with the combination of Deep Leaning and NLP based techniques. It had almost 99% accuracy score in giving appropriate responses to the users for their queries. The performance as well as accuracy is very considerable for our chatbot system along with very small response time. In future, using AI/ML/DL, chatbots can provide students with learning material in an interactive manner on any topic, help them learn quicker through visuals, speech or video and evaluate their responses to gauge their learning. They can be used for – a. Admission process – To assist through documentation guidelines, enrollment procedures, campus info, generate more inquiries, etc. b. Recommending courses – A personalized assistance to students on courses offered and resolving queries on curriculum, credits, internship opportunities, etc. c. On boarding students – FAQ resolution on orientations, campus visits,events, etc. d. Student/Faculty Support point – to offload the staff's work of solving repetitive queries. Student/Faculty Feedback - Collect students' feedback through conversations on learning sessions, campus environment, course improvement, etc.

REFERENCES

1) "TransferTransfo: A Transfer Learning Approach for Neural Network Based Conversational Agents" by Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue (2020)

2) "DialogBERT: Leveraging Pretrained Transformers for End-to-End Conversational Agents" by Weiyan Shi, Yu Zhang, Bing Liu, and Wentao Wang (2020).

3) "ChatGPT: Large-Scale Language Model Fine-Tuning for Conversational Response Generation" by Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, and Y. Yang (2020)

4) "Towards Conversational Agents that Can Chat About Anything" by Emily Dinan, Varvara Logacheva, Valentin Malykh, Alexander Miller, Kurt Shuster, Jack Urbanek, Douwe Kiela, Arthur Szlam, Iulian Serban, Ryan Lowe, et al. (2020)

5) Measuring Conversational Dynamics in the Wild: A Metric for Dialogue Breakdown" by Emily Dinan, Samuel Humeau, Bharath Chintagunta, and Jason Weston (2021)

6) Training Language Models to Produce Structured Data for Chatbots" by David Jurgens, Siddharth Karamcheti, Sungho Park, Larry Chan, and Vicente Ordonez-Roman (2021)

7) Conversational Question Answering over Knowledge Graphs with Transformer Models" by Hongming Zhang, Hong Chen, Minghui Qiu, Wenqiang Lei, Chenyang Tao, Yu Xu, and Xuedong Huang (2022)