ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of our project would be impossible without the set of people, whose guidance and encouragement crowned our effort with success.

We take this opportunity to express our profound gratitude to **Dr. S. F Patil,** Principal of KLE Dr M S Sheshgiri College of Engineering and Technology, for his constant support and encouragement.

We would like to thank **Dr. Dattaprasad Torse** head of Department of Electronics and Communication Engineering, for constantly monitoring the development of the project and setting up precise deadline. His valuable suggestions were the key factors in the completion of project.

We express our gratitude to **Prof. Rudrapppa B Gujanatti,** Project coordinator, Department of Electronics and Communication Engineering, for guiding us in course of our project.

We have great pleasure in expressing our deep sense of gratitude to **Prof Dr. Swati. M,** Project Guide, Department of Electronics and Communication Engineering for providing us with the necessary infrastructure and pleasant environment.

Finally, a note of thanks to the teaching and non-teaching staff of Department of Electronics and Communication Engineering, for their extreme support and who always supported us during the course of our project.

ABSTRACT

Every year freshers and their parents do visit our college website in order to get their queries clarified. Also, college students do visit the website in order to get their queries resolved. Thus we felt the need for an ‘Intelligent Enquiry Bot’ to be associated with the official college website. The Bot should be intelligent enough to resolve the queries of freshers, parents, students, and faculty. The college enquiry chatbot is designed using certain algorithms which understands and analyzes the user queries. This System is basically a web application that provides valid responses to the various queries of the users, which will make use of Natural Language Processing (NLP) and Long Short Term Memory (LSTM) networks, which are a special kind of recurrent Deep Neural Networks (DNN). In this paper, we have completed building a quite intellegent chatbot based on NLP and DNN for basic college-related enquiries and admission related queries especially.

CONTENTS

CHAPTER 1

INTRODUCTION

Chatbot is an intelligent software program that interacts with humans. A chatbot works similar to human-like conversations on chat. Its primary task is to help users by providing answers to their questions by understanding what human wants and guides them to their desired outcome. Nowadays, various chatbots are responsible to solve a number of business related tasks in order to improve customer experiences across many industries like Insurance, E-Commerce, Banking, Healthcare, and many others.

A Deep Learning chatbot uses Natural Language Processing (NLP) to map user inputs to some intents. It will classify the messages to send a prepared response. Thus, using deep learning and natural language processing, the chatbot becomes an intelligent software piece that enables it to process, comprehend and as well response using the natural language understanding. Usually, we use special RNNs called LSTMs to build a chatbot.

When using NLP to develop a chatbot, the main thing one should achieve is to create a chatbot that requires very little or say no human interaction at all. However, it is tough to improve answers and selecting best model to guarantee the most relevant response in the field of chatbots.

The Aim of taking up this project is to provide a chatbot system that deals with academic activities like inquiring about admissions, fees structure, getting details of departments, etc. And using this chat-bot system, the freshers, students and faculty can directly clear their queries in lesser time.

1.1 Overview on Smart College Chatbot

In the modern educational landscape, technology plays a pivotal role in enhancing the student experience. One of the most promising innovations in this domain is the smart college chatbot. Designed to streamline communication, provide instant assistance, and improve overall student engagement, these chatbots are transforming how students and institutions interact.

A smart college chatbot is a virtual assistant tailored to meet the specific needs of college students, faculty, and administrative staff.

By leveraging natural language processing (NLP) and machine learning, these chatbots can understand and respond to a wide range of queries in real-time. They are accessible 24/7, offering a convenient and efficient way to obtain information and support.

Instant Information Access:

Course Details: Provides information on courses, schedules, syllabus, and enrollment.

Campus Services: Offers details about library hours, dining options, and campus events.

Administrative Assistance: Helps with administrative tasks such as registration, fee payments, and form submissions.

Users will not need to go to the college or college website for requests. Users need to enlist to the system and needs to login to the system. After login users can get to the different helping pages. There will be different helping pages through which users can chat by asking questions related with college activities. This chatbot system is an internet application that gives an answer to the broken-down queries of a user.

CHAPTER 2
LITERATURE SURVEY

Reference Paper	Author	Outcomes
[1]	Ms. Ch. Lavanya Susanna.	Student chatbot project is developed with the help of a code igniter which is widely called as a php framework. It analyzes the user queries and also perceives user messages.
[2]	Prof. Ram Manoj Sharma.	College enquiry chatbot system that was made using AI (Artificial Intellegence) algorithms and included few modules like Online chatbot, Online Noticeboards, etc.
[3]	Payal Jain.	Built up a database that consisted of all the related information and also created a web interface (UI) that has two sections. One of them was for basic clients and another was for the admin.
[4]	Sagar Pawar, Omkar Rane.	Developed a UI for which the users have to register before accessing the chatbot.
[5]	Harsh Pawar.	Chatbot was designed by them using the database knowledge. Their proposed system had online enquiry and chatbot system. They used various programming languages in development. They created a user-friendly graphical user interface to send and receive user responses.
[6]	Nitesh Thakur.	Used NLP (Natural Language Processing) for making chatbot. It could be done in two ways, namely

		written text and verbal or voice communication.
[7]	Nidhi Sharma.	Have developed a website and has three modules front end, chatbot, the backend is admin login and chatbot is a college enquiry bot that provides the information regarding fee structure of the different courses using NLP, pattern matching.
[8]	Amit Tiwari.	Author made use of artificial intellegence based algorithms to analyse users queries and understand their message. Its answered appropriately to the user queries. If the answers were found to be invalid, the user just needed to select the invalid answer button which would notify the admin about it.

CHAPTER 3
PROBLEM STATEMENT (IDENTIFICATION)

Colleges and universities face a growing demand for instant, accurate, and efficient communication with students, faculty, and staff. Traditional methods of communication, such as email and in-person consultations, are often slow and can result in information overload or miscommunication. A smart chatbot can serve as a solution by providing 24/7 assistance, automating routine tasks, and enhancing the overall user experience within the college community.

The chatbot should have a clean and intuitive user interface, accessible through popular messaging platforms like Facebook Messenger, WhatsApp, or a dedicated web interface. Implement advanced natural language processing (NLP) techniques to understand and interpret user queries accurately. This includes handling variations in language, slang, and context to provide relevant responses. The chatbot should be able to personalize responses based on user preferences, previous interactions, and demographic information. This could include recommending relevant campus events, study resources, or clubs based on the user's interests.

CHAPTER 4
OBJECTIVES

By providing a clean and initiative interface coupled with natural language understanding, the chatbot aims to enhance the overall user experience for college students seeking assistance or information. By providing a clean and intuitive interface coupled with natural language understanding, the chatbot aims to enhance the overall user experience for college students seeking assistance or information.

By providing information on upcoming events and activities, the chatbot encourages student participation and engagement in campus life, contributing to a vibrant and connected campus community. The chatbot serves as a readily available resource for answering frequently asked questions, resolving issues, and providing support, reducing the need for students to wait in queues or search for information manually.

Ensuring compliance with data protection regulations and implementing robust security measures safeguards user privacy and instills trust in the chatbot as a reliable and secure resource for accessing college-related information and services.

CHAPTER 5
REQUIRMENT ANALYSIS

5.1 SOFTWARE AND HARDWARE REQUIREMENTS SPECIFICATION DOCUMENT

SOFTWARE AND HARDWARE REQUIREMENTS:

Hardware:

Operating system: Windows 7 or 7+

RAM: 2 GB MEMORY

Hard disc or SSD: More than 500 GB

Processor: Processor Dual Core

Software:

Software: Python 3.6 or high version

IDLE: PyCharm.

Framework: Flask 3.2 SYSTEM USE CASE

5.2 SYSTEM USE CASE

A college enquiry chatbot can have several use cases, including: Admission Enquiry: The chatbot can provide information about the admission process, eligibility criteria, important dates, and documents required for admission. 16 Course Information: The chatbot can provide detailed information about the courses offered by the college, including the duration of the course, syllabus, fees, and career opportunities. Campus Facilities: The chatbot can provide information about the various facilities available on the college campus, such as libraries, laboratories, sports facilities, and accommodation options. Fees and Scholarships: The chatbot can provide information about the fees structure for different courses and scholarships available for students based on accommodation options. Fees and Scholarships: The chatbot can provide information about the fees structure for different courses and scholarships available for students based on their academic performance. Important Dates: The chatbot can remind students about important dates such as admission deadlines, fee payment dates, and exam schedules. FAQs: The chatbot can answer frequently asked questions by students, such as how to apply for admission, how to check the admission status. Student life: The chatbot can provide information about student life at the college, including clubs and societies, extracurricular activities, and student resources. Counselling: The chatbot can provide counselling to students regarding their career options, course selection, and academic performance. Academic support: The chatbot can assist students with academic enquiries, including course registration, exam schedules, and study resources. Admission and enrolment enquiries: The chatbot can assist prospective students with admission and enrolment enquiries, including deadlines, application requirements, and documentation. Overall, a college enquiry chatbot can provide a seamless and hassle-free experience for students who are looking for information about the college and its courses.

CHAPTER 6
SYSTEM DESIGN ARCHITECTURE

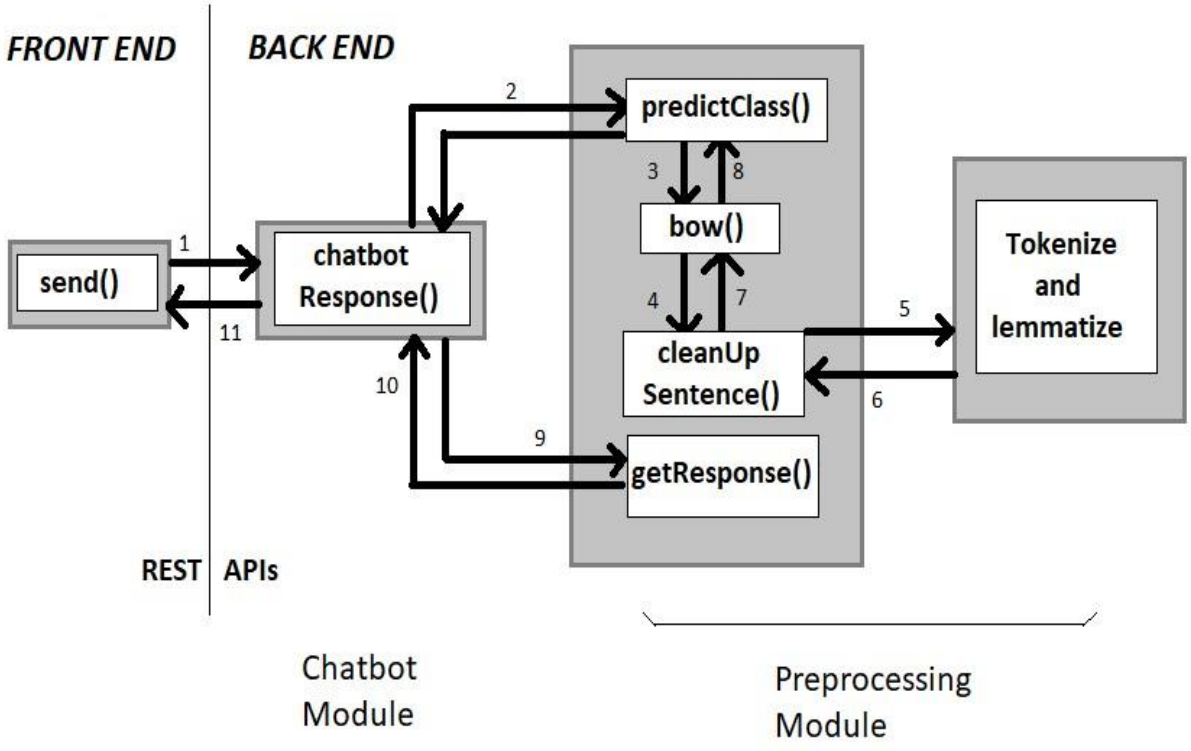


Fig. 6.1 System Design Architecture

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 to update the chats. This is how the frontend and backend are connected.

Not only we had just built the development code, but also, we had written a few test-cases to check whether the expected functionality is achieved in most of the circumstances.

We have written a few test-cases to ensure that expected functionality of the app is retained. For this, we have used Jest framework to test the React front-end and Pytest and Unittest libraries to test the Python back-end.

The back-end test cases are written for the API requests being sent from UI to backend and receiving the proper response from backend back to the UI.

Coding

{"tag": "EC",

"patterns": ["Total number of teching faculty in EC department", "No of faculties in ec", "faculty in ec", "no of faculty available in ec", "faculty strength"],

"responses": ["The Department is established in the year 1986.This is one of the premier Departments in the college and is most preferred by the student community.
 The department has well qualified teaching faculty comprising of 02-Profs, 01-Assoc. Profs and 13-Asst.Profs.
 Training the students for the state of the art technology is the main aim of the department.
"],

"context": [""]

},

{"tag": "HODEC",

"patterns": ["Who is HOD of EC?","HOD of EC","HOD EC","HOD of E&C","HOD of Electronics and communications"],

"responses": ["Dr.Dattaprasad Torse(Professor & H.O.D)
Experience of 22 years "],

"imgLink": ["static/datta.jpeg"],

"context": [""]

},

{"tag": "MECH",

"patterns": ["Total number of teching faculty in Mechanical department", "No of faculties in mechanical", "faculty in mechanical", "no of faculty available in mechanical", "faculty strength"],

"responses": ["Number of Faculties in Mechanical Dept are \n 19 are Teaching \n 8 are non teaching "],

"context": [""]

},

{"tag": "MECH STAFF",

"patterns": ["MECH Staff Details","Faculties in MECH","Faculty details of MECH","Teachers in MECH","MECH Staff","E&C Staff Information","Teaching Staff in MECH"],

"responses": ["1)Dr.S.F.Patil (Ph.D)

2)Dr.C.V.Adake(Professor,Ph.D(IIT Bombay))

 3) Dr.A.H.Gadagi(Associate Professor,Ph.D(IIT Kharagpur))

4)Dr.D.C.Patil(Associate Professor,Ph.D)

5)Dr.R.G.Lingannavar(Assistant Professor,Ph.D(IIT Guwahati))

6)Dr.S.B.Anagadi(Associate Professor,Ph.D)

7)Mr.S.B.Yadwad(Assistant Professor,M.Tech)

8)Dr.R N.Chikkanagoudar(Assistant Professor,Ph.D)

9)Mr.N.K.Kelageri(Assistant Professor, M.Tech)

10)Mr.S.A.Janawade(Assistant Professor, M.Tech)

11)Mr.M.Sadiq A.Pachapuri(Assistant Professor,M.Tech)

12)Mr.S.N.Nadurkar(Assistant Professor, M.Tech)

13)Mr.Sachidanand T.G(Assistant Professor, M.Tech)

14)Mr.B.G.Koujalagi(Assistant Professor, M.Tech)

15)Mr.Ramesh H Katti(Assistant Professor, M.tech)

16)Mrs.Niranjan L.Pattar (Assistant Professor,M.Tech)

17)Mr.Satish L Hulamani(Assistant Professor,M.Tech)

18)Mr.S.M.Golabhanvi(Assistant Professor,M.tech)

19)Mrs.Mallesh Sanjeeannavar(Assistant Professor)

"],

"context": [""]

},

Python Code

```
import random
import numpy as np
import pickle
import json
from flask import Flask, render_template, request
from flask_ngrok import run_with_ngrok
import nltk
from keras.models import load_model
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()



# chat initialization
model = load_model("chatbot_model.h5")
intents = json.loads(open("intents.json").read())
words = pickle.load(open("words.pkl", "rb"))
classes = pickle.load(open("classes.pkl", "rb"))


app = Flask(_name_)
# run_with_ngrok(app)


@app.route("/")
def home():
    return render_template("index.html")
```

```
@app.route("/chat")
def chatbot():
    return render_template("index1.html")



@app.route("/get", methods=["POST"])
def chatbot_response():
    msg = request.form["msg"]
    if msg.startswith('my name is'):
        name = msg[11:]
        ints = predict_class(msg, model)
        res1 = getResponse(ints, intents)
        res =res1.replace("{n}",name)
    elif msg.startswith('hi my name is'):
        name = msg[14:]
        ints = predict_class(msg, model)
        res1 = getResponse(ints, intents)
        res =res1.replace("{n}",name)
    else:
        ints = predict_class(msg, model)
        res = getResponse(ints, intents)
    return res



# chat functionalities
def clean_up_sentence(sentence):
    sentence_words = nltk.word_tokenize(sentence)
```

```
    sentence_words = [lemmatizer.lemmatize(word.lower()) for word in sentence_words]
    return sentence_words


# return bag of words array: 0 or 1 for each word in the bag that exists in the sentence
def bow(sentence, words, show_details=True):
    # tokenize the pattern
    sentence_words = clean_up_sentence(sentence)
    # bag of words - matrix of N words, vocabulary matrix
    bag = [0] * len(words)
    for s in sentence_words:
        for i, w in enumerate(words):
            if w == s:
                # assign 1 if current word is in the vocabulary position
                bag[i] = 1
                if show_details:
                    print("found in bag: %s" % w)
    return np.array(bag)


def predict_class(sentence, model):
    # filter out predictions below a threshold
    p = bow(sentence, words, show_details=False)
    res = model.predict(np.array([p]))[0]
    ERROR_THRESHOLD = 0.25
    results = [[i, r] for i, r in enumerate(res) if r > ERROR_THRESHOLD]
    # sort by strength of probability
```

```
    results.sort(key=lambda x: x[1], reverse=True)
    return_list = []
    for r in results:
        return_list.append({"intent": classes[r[0]], "probability": str(r[1])})
    return return_list



def getResponse(ints, intents_json):
    tag = ints[0]["intent"]
    list_of_intents = intents_json["intents"]
    for i in list_of_intents:
        if i["tag"] == tag:
            result = random. choice(i["responses"])
            break
    return result



if _name_ == "_main_":
    app.run ()
```

To train the chatbot

```
import random
from tensorflow.keras.optimizers import SGD
from keras.layers import Dense, Dropout
from keras.models import load_model
from keras.models import Sequential
import numpy as np
import pickle
import json
import nltk
import tensorflow as tf
from nltk.stem import WordNetLemmatizer
from tensorflow.keras.optimizers import Adam

lemmatizer = WordNetLemmatizer()
nltk.download('omw-1.4')
nltk.download("punkt")
nltk.download("wordnet")
lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate=0.01,
    decay_steps=10000,
    decay_rate=0.9)
optimizer = tf.keras.optimizers.Adam(learning_rate=lr_schedule)



# init file
```

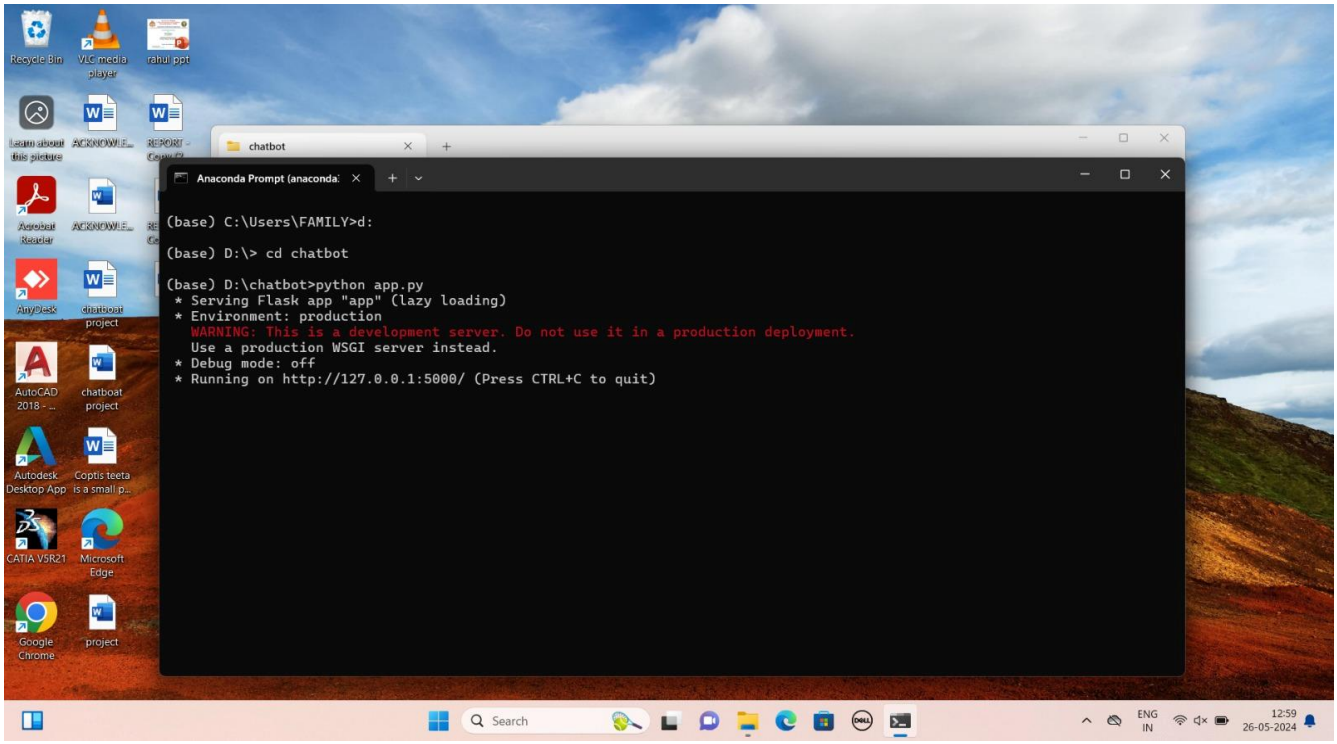
```
words = []
classes = []
documents = []
ignore_words = ["?", "!"]
data_file = open("intents.json").read()
intents = json.loads(data_file)

# words
for intent in intents["intents"]:
    for pattern in intent["patterns"]:

        # take each word and tokenize it
        w = nltk.word_tokenize(pattern)
        words.extend(w)
        # adding documents
        documents.append((w, intent["tag"]))

        # adding classes to our class list
        if intent["tag"] not in classes:
            classes.append(intent["tag"])

# lemmatizer
words = [lemmatizer.lemmatize(w.lower()) for w in words if w not in ignore_words]
words = sorted(list(set(words)))

classes = sorted(list(set(classes)))
```

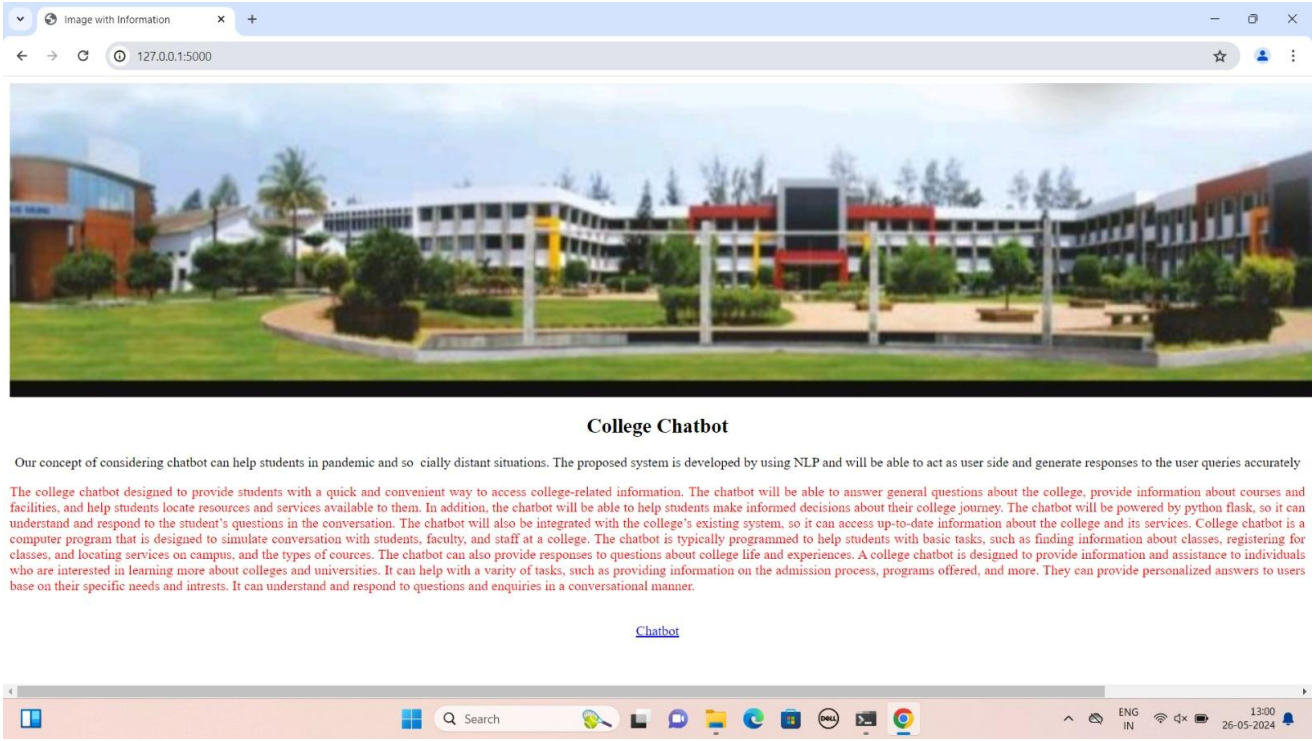
```
print(len(documents), "documents")

print(len(classes), "classes", classes)

print(len(words), "unique lemmatized words", words)



pickle.dump(words, open("words.pkl", "wb"))
pickle.dump(classes, open("classes.pkl", "wb"))

# training initializer
# initializing training data
training = []
output_empty = [0] * len(classes)
for doc in documents:
    # initializing bag of words
    bag = []
    # list of tokenized words for the pattern
    pattern_words = doc[0]
    # lemmatize each word - create base word, in attempt to represent related words
    pattern_words = [lemmatizer.lemmatize(word.lower()) for word in pattern_words]
    # create our bag of words array with 1, if word match found in current pattern
    for w in words:
        bag.append(1) if w in pattern_words else bag.append(0)

    # output is a '0' for each tag and '1' for current tag (for each pattern)
    output_row = list(output_empty)
```

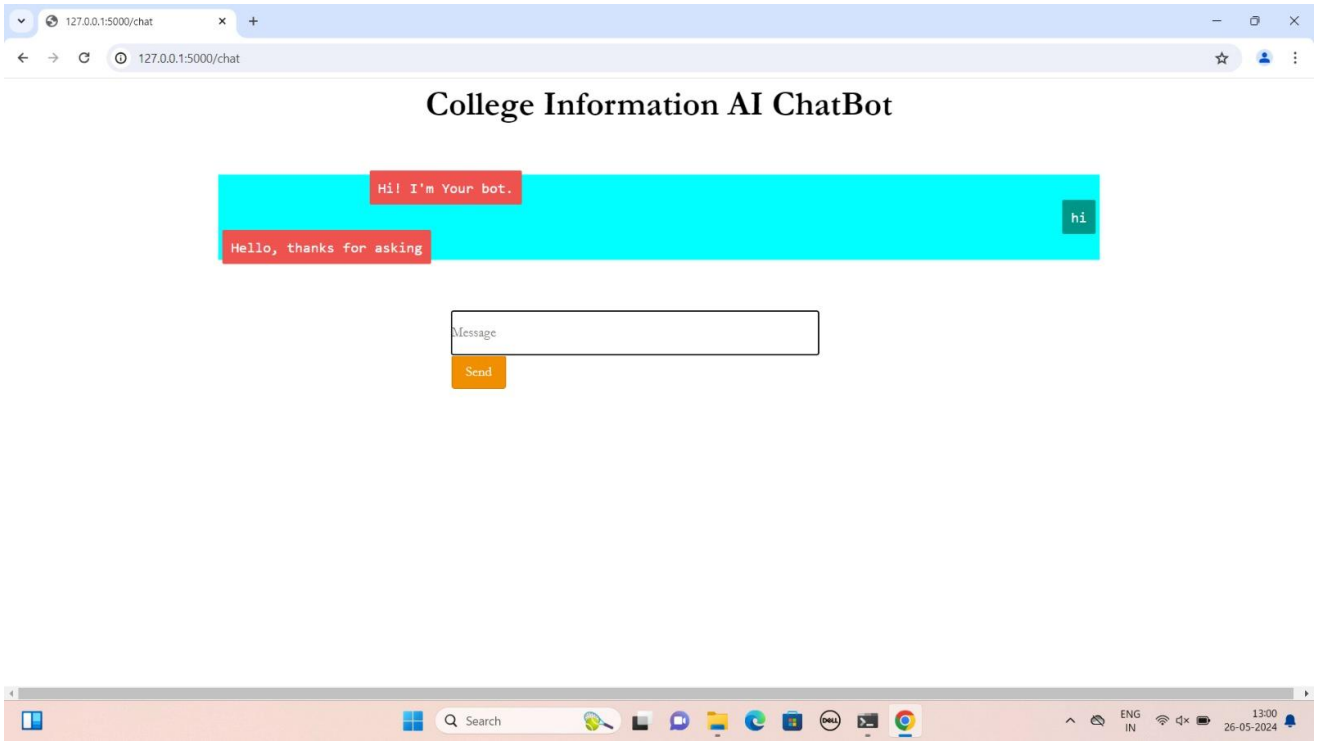
```
    output_row[classes.index(doc[1])] = 1


    training.append([bag, output_row])
# shuffle our features and turn into np.array
random.shuffle(training)
training = np.asarray(training, dtype="object")
# create train and test lists. X - patterns, Y - intents
train_x = list(training[:, 0])
train_y = list(training[:, 1])
print("Training data created")


# actual training
# Create model - 3 layers. First layer 128 neurons, second layer 64 neurons and 3rd output layer contains number of neurons
# equal to number of intents to predict output intent with softmax
model = Sequential()
model.add(Dense(128, input_shape=(len(train_x[0]),), activation="relu"))
model.add(Dropout(0.5))
model.add(Dense(64, activation="relu"))
model.add(Dropout(0.5))
model.add(Dense(len(train_y[0]), activation="softmax"))
model.summary()


# Compile model. Stochastic gradient descent with Nesterov accelerated gradient gives good results for this model
"""lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate=0.01,
```

```
    decay_steps=1e-6,
    decay_rate=0.9)"""

#sgd = tf.keras.optimizers.SGD(learning_rate=lr_schedule)
sgd = tf.keras.optimizers.legacy.SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss="categorical_crossentropy", optimizer=sgd, metrics=["accuracy"])

# for choosing an optimal number of training epochs to avoid underfitting or overfitting use an early stopping callback to keras
# based on either accuracy or loos monitoring. If the loss is being monitored, training comes to halt when there is an
# increment observed in loss values. Or, If accuracy is being monitored, training comes to halt when there is decrement observed in accuracy values.

# from keras import callbacks
# earlystopping = callbacks.EarlyStopping(monitor ="loss", mode ="min", patience = 5, restore_best_weights = True)
# callbacks =[earlystopping]

# fitting and saving the model
hist = model.fit(np.array(train_x), np.array(train_y), epochs=200, batch_size=5, verbose=1)
model.save("chatbot_model.h5", hist)
print("model created")
```

Screenshots






127.0.0.1:5000/chat
127.0.0.1:5000/chat
College Information AI ChatBot
Hi! I'm Your bot.
hi
Hello, thanks for asking
Message
Send
Search
ENG
IN
13:00
26-05-2024

CHAPTER 7
METHODOLOGY

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

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.

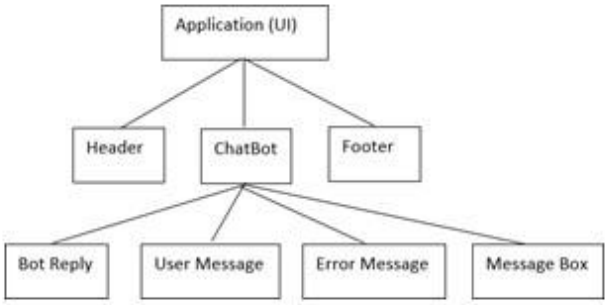


Fig7.1 Frontend Modules Hierarchy

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 displaying the user queries that are entered by the user in the message box provided after sending to the chatbot

ii. **Bot Reply Component** – 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. **Message Box 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. **Error Message Component** – which displays 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 text data, 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:

i. **Tokenization** – where it is the process of breaking down the complete input text (i.e., user query) into smaller parts i.e., words. Then we iterate through the patterns that are provided in the intents file and then tokenize the sentences using "nltk.word_tokenize()" function. After

that, we append every word contained in the words list to create a classes list for the tags in the intents file.

ii. **Lemmatization** – This is a process of converting a word to its lemma (root) form. We also then create a pickle file to store the python objects which we used while predicting the responses. It generally finds the root word from the given word.

c) Creating training and testing data

As a next step, we create the training and testing data by splitting the available data. Our input data is the patterns and output data is the classes to which the pattern belongs. As the computer couldn't understand text data, we will convert the text into numbers.

d) Building the deep learning model

As we now have the data i.e., both training and testing data ready, we now concentrate on building a Deep Neural Network model. As far as speaking about the Deep Learning model, we created a model consisting of 3 layers as follows:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 128)               14720
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 11)                715
=================================================================
Total params: 23,691
Trainable params: 23,691
Non-trainable params: 0
```

Fig.7.2 Model summary

Thus we trained the chatbot on the dataset containing categories (intents), patterns and responses. We made use of 'Long Short Term Memory' (LSTM), a special 'Recurrent Neural Network' (RNN) in order to classify the user's message and then we provide some random response from the list of predefined responses. We used Keras Sequential API for doing this. And as mentioned earlier, the model is saved as "chatbot_model.h5" file for future purposes.

e) Predict the response

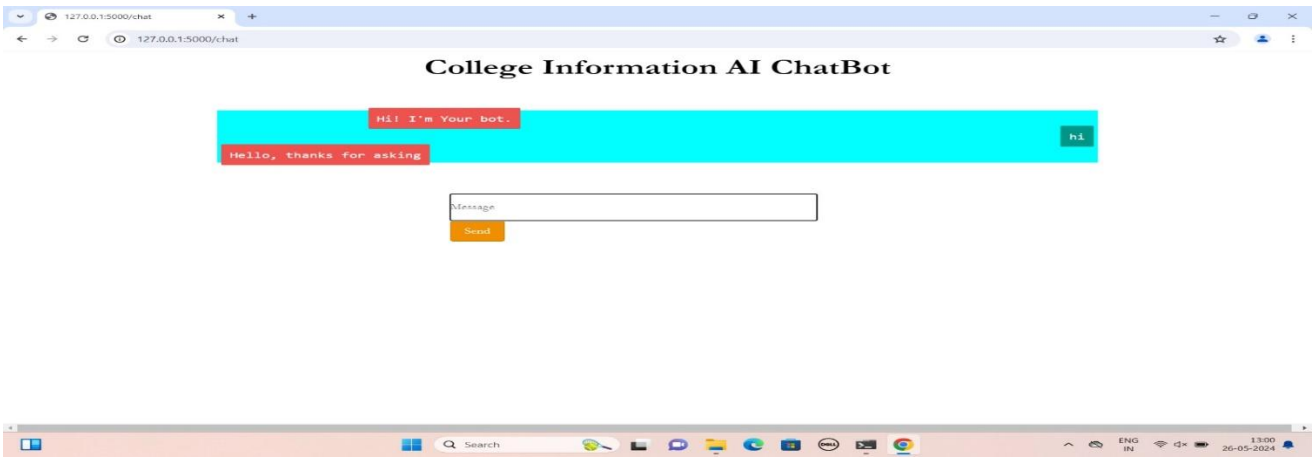
Now, the final phase of a chatbot is to make prediction of the response. Now, we load the saved model and use React App User Interface in order to predict the response and display the same. The model only tells us the class to which it belongs, then we make use of a few helper functions to identify the class label and retrieve a random response from 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.

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 back-end test cases are written for the API requests being sent from UI to backend and receiving the proper response from backend back to the UI.

Screenshots of Outcomes



CHAPTER 8

RESULTS AND ANALYSIS

The React UI's Front-end and Python Back-end unit testing reports are as follows:

Fig. 8.1 React UI Testing Report

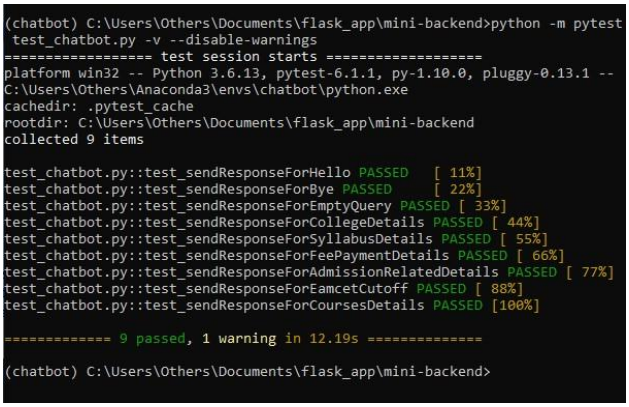


Fig. 8.2 Flask App Testing Report

The deep learning model we have used was trained to be very accurate in returning appropriate responses to the user queries. It almost achieved an accuracy of 99% after training it for about 200 epochs.

Consists of few plots depicting the model's accuracy and loss based on the training and validation data sets. As you can see, after 200 epochs, the accuracy has been maximized whereas loss has been minimized.

CHAPTER 9
CONCLUSION AND FUTURE WORK

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] Ms.Ch.Lavanya Susanna, R.Pratyusha, P.Swathi, P.Rishi Krishna, V.Sai Pradeep, "College Enquiry Chatbot", International Research Journal of Engineering and Technology (IRJET), e-ISSN: 2395- 0056, p-ISSN: 2395- 0072, Volume: 07 Issue: 3 Mar 2020 pp 784-788 .

[2] Assistant Prof Ram Manoj Sharma, Chatbot based College Information System", RESEARCHREVIEW International Journal of Multidisciplinary, ISSN: 2455-3085 (Online), Volume-04, Issue03, March-2019, pp 109-112.

[3] P.Nikhila, G.Jyothi, K.Mounika, Mr. C Kishor Kumar Reddy and Dr. B V Ramana Murthy on, "Chatbots Using Artificial Intelligence", International Journal of Research and Development, Volume VIII, Issue I, January/2019, ISSN NO:2236- 6124, pp1-12.

[4] Payal Jain, "College Enquiry ChatBot Using Iterative Model", International Journal of Scientific Engineering and p 80-8Research (IJSER), ISSN (Online): 2347-3878, Volume 7 Issue 1, January 2019, p3

[5] Sagar Pawar, Omkar Rane, Ojas Wankhade, Pradnya Mehta, "A Web Based College Enquiry Chatbot with Results", International Journal of Innovative Research in Science, Engineering and Technology, ISSN(Online): 2319- 8753, ISSN (Print): 2347-6710, Vol. 7, Issue 4, April 2018, pp 3874-3880.

[6] Prof.K.Bala, Mukesh Kumar ,Sayali Hulawale, Sahil Pandita, "Chatbot For College Management System Using A.I", International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056, p-ISSN: 2395- 0072, Volume: 04 Issue: 11 | Nov -2017, pp2030-2033.

[7] Jincy Susan Thomas, Seena Thomas, "Chatbot Using Gated End-to End Memory Networks", International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056, p-ISSN: 2395-0072, Volume: 05 Issue: 03 Mar 2018, pp 3730- 3735.

[8] Prof. Suprita Das, Prof. Ela Kumar, "Determining Accuracy of Chatbot by applying Algorithm Design and Defined process", 4th International Conference on Computing Communication and Automation (ICCCA), 2018, 978-1-5386-6947-1/18/2018 IEEE, pp 1-6.

[9] Prof.K.Bala, Mukesh Kumar ,Sayali Hulawale, Sahil Pandita, "Chatbot For College Management System Using A.I", International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056, p-ISSN: 2395-0072, Volume: 04 Issue: 11 | Nov -2017, pp 2030-2033.

[10] Nitesh Thakur, Akshay Hiwrale, Sourabh Selote, Abhijeet Shinde and Prof. Namrata Mahakalkar, "Artificially Intelligent Chatbot", Universal Research Reports , ISSN : 2348 – 5612, Volume : 04 , Issue : 06, July - September 2017, pp 43-47.

[11] Amey Tiwari, Rahul Talekar, Prof.S.M.Patil, "College Information Chat Bot System", International Journal of Engineering Research and General Science, ISSN 2091-2730, Volume 5, Issue 2, March April, 2017, pp 131-137.

[12] Malusare Sonali Anil, Kolpe Monika Dilip, Bathe Pooja Prashant, "Online Chatting System for College Enquiry using Knowledgeable Database", Savitribai Phule Pune University, Preliminary Project Report, pp 1-53,2017.

[13] Balbir Singh Bani, Ajay Pratap Singh, "College Enquiry Chatbot Using A.L.I.C.E (Artificial Linguistic Internet Computer Entity)", International Journal of New Technology and Research (IJNTR), ISSN:2454-4116, Volume-3, Issue-1, January 2017 Pages 64-65.

[14] Ms.Ch.Lavanya Susanna, R.Pratyusha, P.Swathi, P.Rishi Krishna, V.Sai Pradeep, "College Enquiry Chatbot", International Research Journal of Engineering and Technology (IRJET), e-ISSN: 2395-0056, p-ISSN: 2395-0072, Volume: 07 Issue: 3 Mar 2020 pp 784788.

[15] Agnese Augello, Giovanni Pilato, Alberto Machi' ICAR - Istituto di Calcolo e Reti ad Alte Prestazioni CNR - Consiglio Nazionale elleRicercheVialedelleScienze , 978-0-7695-4859-3/12 $26.00 © 2012 IEEE . "An Approach to Enhance Chatbot Semantic Power and Maintainability: Experiences within the FRASI Project"

[16] Emanuela Haller, Traian Rebedea Faculty of Automatic Control and Computers university Politehnica of Bucharest, 978-0-7695-4980-4/13 $26.00 © 2013 IEEE. "Designing a Chat-bot that Simulates a Historical Figure".

[17] BayuSetiaji, Ferry Wahyu Wibowo , Department of Informatics Engineering STMIK AMIKOM Yogyakarta, Yogyakarta, Indonesia, 2166-0670/16 $31.00 © 2016 IEEE "Chatbot Using A Knowledge in Database-Human-to-Machine Conversation Modeling".