CHATBOT FOR FRESHERS

A project submitted in participation of the representation for the miniproject of the Degree of

BACHELOR OF

TECHNOLOGY IN

COMPUTER SCIENCE ENGINEERING

Submitted by

SUCHARITHA.S – **O170720**
MOUNIKA.CH - **O170726**
GONGOTHRI.P- **O170738**
SRI SAIKRISHNA.J - **O170745**
NAGABABU.B - **O170753**

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE AND TECHNOLOGIES

ONGOLE-523225

AUGUST 2022

ACKNOWLEDGEMENT

We take this opportunity to express deep gratitude to the people who have been instrumental in the successful completion of the project.

We would like to thank our project guide Ms.GBL Prasanna, which was greatly helpful in the progression and smoothness of the entire project work. We would like to show our gratitude to our beloved HOD, Mr.Buthapati Sampath Babu for his encouragement in the aspect of our course, RAJIV GANDHI UNIVERSITY OF KNOWLEDGE AND TECHNOLOGIES, who gave us official support for the progress of our project.

We would also like to extend our gratitude to Lab Technicians and our sincere gratitude to all the faculty members and Non-Teaching Staff in the Department of Information Technology for supporting us. We would also like to thank our parents and friends, who have willingly helped us out with their abilities in completing the project work.

With Gratitude,

Sucharitha Sikhinam -o170720
Mounika Chettibilli -o170726
Gangothri Pathuri -o170738
Sri Saikrishna Jogi -o170745
Naga Babu.Bavireddy -o170753

TABLE OF CONTENTS

ABSTRACT

In this Python project, we are going to build a chatbot using deep learning techniques. The chatbot will be trained on the dataset which contains categories (intents), pattern and responses. The chatbot is a program design to simulate human conversations. We use a special recurrent neural network (LSTM) to classify which category the user's message belongs to and then we will give a random response from the list of responses. In our project, we aim to build a student friendly chatbot which gives information about the college.

INTRODUCTION

CHATBOT

A chatbot is an intelligent piece of software that is capable of communicating and performing actions similar to a human. Chatbots are used a lot in customer interaction, marketing on social network sites and instantly messaging the client. There are two basic types of chatbot models based on how they are built; Retrieval based and Generative based models

1. Retrieval based Chatbots

A retrieval-based chatbot uses predefined input patterns and responses. It then uses some type of heuristic approach to select the appropriate response. It is widely used in the industry to make goal-oriented chatbots where we can customize the tone and flow of the chatbot to drive our customers with the best experience.

2. Generative based Chatbots

Generative models are not based on some predefined responses. They are based on seq 2 seq neural networks. It is the same idea as machine translation. In machine translation, we translate the source code from one language to another language but here, we are going to transform input into an output. It needs a large amount of data and it is based on Deep Neural networks.

TYPES OF CHATBOTS

1. Menu/Button-Based Chatbots

The most commonly used and the simplest type of chatbots in the market today are the menu based chatbots, which are in form of buttons and top-down menus. These chatbots follow the principles of decision trees, where you make your decisions to get the ultimate answers. The user is instructed to make these decisions by selecting their options and dig deeper to- wards the appropriate response from the AI. However, these menu-based chatbots are comparatively slower in terms of performance and cannot be completely reliable to get the desired answer.

2.Keyword Recognition-Based Chatbots

These chatbots recognizes specific keywords in order to produce a desired result. They listen to what the users enter and respond accordingly. With the help of the AI technology and customized keywords list, the bot determines an appropriate response to the user by using the algorithms. These chatbots will start to fail when there are keyword redundancies between several related questions. For example, if a user asked the question 'How do I set up an auto-login authentication on my phone?', the bot would likely use the keywords like 'auto', 'login', to determine which answer is the best to respond with.

3.Contextual Chatbots

Contextual chatbots are one of the most technologically advanced bots in the market today. They utilize Machine Learning and Artificial Intelligence technologies like voice recognition, speech-to-text conversion algorithms, etc to interpret the user's sentiments. The underlying ideology of this type of bot is to figure out what the user's intentions are and correspondingly present a thoughtful answer by deciphering the pattern in the database. The bot learns and grows over time by encountering many more experiences. A simple example of such a bot can be seen in a food delivery application. Here the previous order history along with the user's payment options and delivery address are stored on the database. These chatbots analyze the user's perspective and suggest recommendations based on consecutive orders and user's likings.

EXISTING SYSTEM

RULE-BASED CHATBOT

Menu/Button-Based Chatbots

The most commonly used and the simplest type of chatbots in the market today are the menu based chatbots, which are in form of buttons and top-down menus. These chatbots follow the principles of decision trees, where you make your decisions to get the ultimate answers. The user is instructed to make these decisions by selecting their options and dig deeper to- wards the appropriate response from the AI. However, these menu-based chatbots are comparatively slower in terms of performance and cannot be completely reliable to get the desired answer.

Rule based chatbots are also referred to as decision based chatbots. They use a defined set of rules. They provide pre-defined options to select which restricts the user to chat only in the given options. If the user asks a new question, the Bot won't be able to respond. Rule-based chatbots are also referred to as decision-tree bots. As the name suggests, they use a series of defined rules. These rules are the basis for the types of problems the chatbot is familiar with and can deliver solutions for.

Like a flowchart, rule-based chatbots map out conversations. They do this anticipation of what a customer might ask, and how the chatbot should respond. Rule-based chatbots can use very simple or complicated. Also, they only perform and work with the scenarios you train them for. A rule-based chatbot uses a tree-like flow instead of AI to help guests with their queries. This means that the chatbot will guide the guest with follow-up questions to eventually get to the correct resolution.

The structures and answers are all pre-defined so that you are in control of the conversation. Rule-based chatbots are great for smaller numbers and straightforward queries, like to book a table at a restaurant or when asking for opening hours. The chatbot doesn't need extensive training which makes the implementation process faster and less complicated. Since the technology and implementation are simpler, the price is usually also more affordable. By pre-defining the structures and answers, you can better control the behavior and responses of the chatbot.

LIMITATIONS OF EXISTING SYSTEMS

- Rule based chatbots can’t learn on their own, they only provide answers your legal team provides from a predefined set of rules.
- That means your clients experience with rule based bots is very linear. In other words if your client asked questions outside its preset understanding they fail and need human intervention.
- A rule-based chatbot can’t capture typos which means that in some cases it won’t understand the guest, which can cause frustrations.
- The interactions with a simple chatbot feel robotic rather than conversational.
- They cannot learn on their own which means that any improvements need to be made manually.
- They can't, however, answer any questions outside of the defined rules.
- These chatbots do not learn through interactions.
- If the user asks a new question, the Bot won’t be able to respond.

PROPOSED WORK

INTRODUCTION

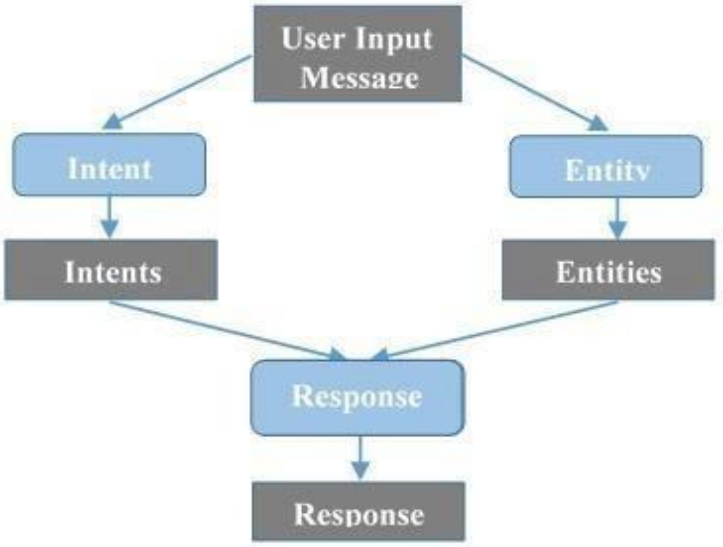
In this Python project with source code, we are going to build a chatbot using deep learning techniques. The chatbot will be trained on the dataset which contains categories (intents), pattern and responses. We use a special recurrent neural network (LSTM) to classify which category the user's message belongs to and then we will give a random response from the list of responses.

We proposed to build an interactive chatbot to answer fresher student queries regarding the college. The main aim of our system is to give response/answer to the different college queries that were asked by the students who aiming to join the college.

Many people will have different queries regarding what college to join. To clear those doubts, they might browse the Internet, try to read different articles. But the problem here is the exact doubt may not be cleared when directly searching for their doubt as a question. Even the answer found, may vary with some other article on another website. The user needs to read and search the whole content of the official website for finding the answer.

BLOCK DIAGRAM

This is the Block diagram representing the work of chatbot. Firstly, the intents.json file is creating with the most frequently asked questions of the particular college which contains tags and patterns and their corresponding responses. After taking input from the user data will be pre-processed which removes unnecessary and inconsistent data. Then the algorithm matches the entities which means user input with the intents and predicts a response. This response is then displayed in a GUI window.



PROPOSED SYSTEM

Creating a dataset

The dataset ‘intents.json’ is created which contains tags, patterns and corresponding responses. This is a JSON file that contains the patterns we need to find and the responses we want to return to the user.

Pre-requisites

The project uses Python, Keras and Natural Language Processing (NLTK- Natural Language Toolkit)

NATURAL LANGUAGE PROCESSING

Natural language processing (NLP) refers to the branch of computer science—and more specifically, the branch of artificial intelligence or AI—concerned with giving computers the ability to understand text and spoken words in much the same way human beings can. NLP combines computational linguistics—rule-based modeling of human language—with statistical, machine learning, and deep learning models.

Together, these technologies enable computers to process human language in the form of text or voice data and to ‘understand’ its full meaning, complete with the speaker or writer’s intent and sentiment. NLP drives computer programs that translate text from one language to another, respond to spoken commands, and summarize large volumes of text rapidly—even in real time.

DEEP LEARNING

Deep learning is a subset of machine learning, which is essentially a neural network with three or more layers. These neural networks attempt to simulate the behavior of the human brain—albeit far from matching its ability—allowing it to “learn” from large amounts of data. While a neural network with a single layer can still make approximate predictions, additional hidden layers can help to optimize and refine for accuracy.

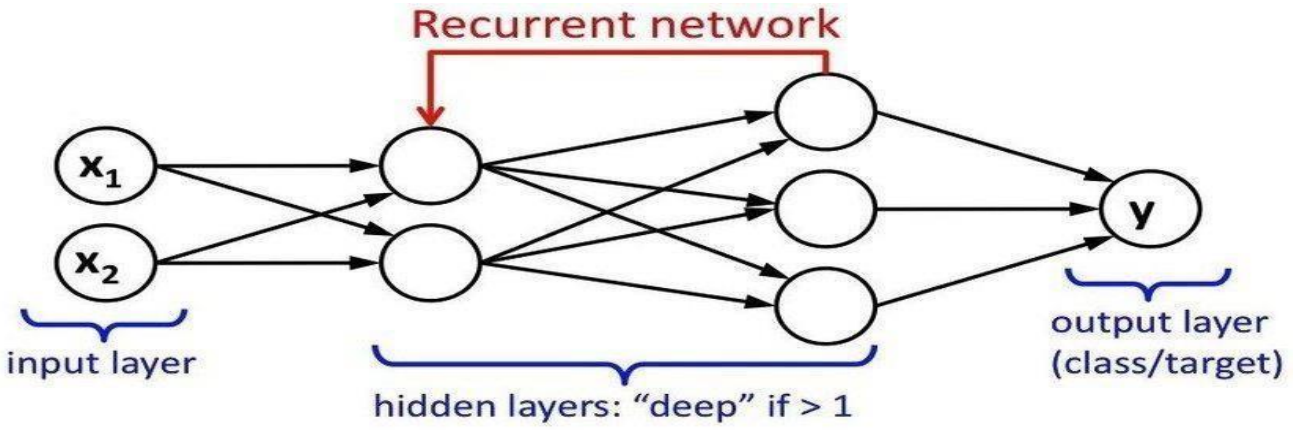
Deep learning technology lies behind everyday products and services (such as digital assistants, voice-enabled TV remotes, and credit card fraud detection) as well as emerging technologies (such as self-driving cars).

LSTM (Long Short-Term Memory Networks)

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. This is a behavior required in complex

problem domains like machine translation, speech recognition, and more.

LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series.



METHODOLOGY

1. Import and load the data file
2. Preprocess data
3. Create training and testing data
4. Build the model
5. Predict the response

Import and load the data file

First, make a file name as train_chatbot.py. We import the necessary packages for our chatbot and initialize the variables we will use in our Python project. The data file is in JSON format so we used the json package to parse the JSON file into Python.

```
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer =  WordNetLemmatizer()
import json
import pickle
import numpy as np
from keras.models import Sequential
from keras.layers import Dense, Activation, Dropout
from keras.optimizers import SGD
import random
words=[]
classes = []
documents = []
ignore_words = ['?', '!']
data_file = open('intents.json').read()
intents = json.loads(data_file)
```

This is how our intents.json file looks like.

```
{
   "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?"
                ]
        },

{
                "tag": "course",
                "patterns": [
                        "list of courses",
                        "list of courses offered",
                        "list of courses offered in vignan lara",
                        "what are the courses offered in your college?",
                        "courses?",
                        "courses offered",
                        "courses offered in vlits",
                        "courses you offer",
                        "branches?",
                        "courses available at lara?",
                        "branches available at lara?",
                        "what are the courses in vignan?",
                        "what are branches in lara?",
                        "what are courses in vignan lara?",
                        "branches available in vlits?",
                        "can you tell me the courses available in vlits?",
                        "can you tell me the branches available in vlits?",
                        "Civil engineering?",
                        "civil",
                        "it",
                        "IT"
                ]
}
```

Preprocess data

When working with text data, we need to perform various preprocessing on the data before we make a machine learning or a deep learning model. Based on the requirements we need to apply various operations to preprocess the data.

Tokenizing is the most basic and first thing you can do on text data. Tokenizing is the process of breaking the whole text into small parts like words.

Here we iterate through the patterns and tokenize the sentence using nltk.word_tokenize() function and append each word in the words list. We also create a list of classes for our tags.

```
for intent in intents['intents']:
for pattern in intent['patterns']:
#tokenize each word
w = nltk.word_tokenize(pattern)
words.extend(w)
#add documents in the corpus
documents.append((w, intent['tag']))
# add to our classes list
if intent['tag'] not in classes:
classes.append(intent['tag'])
```

Now we will lemmatize each word and remove duplicate words from the list. Lemmatizing is the process of converting a word into its lemma form and then creating a pickle file to store the Python objects which we will use while predicting.

```
# lemmatize, lower each word and remove duplicates
words = [lemmatizer.lemmatize(w.lower()) for w in words if w not in ignore_words]
words = sorted(list(set(words)))
# sort classes
classes = sorted(list(set(classes)))
# documents = combination between patterns and intents
print (len(documents), "documents")
# classes = intents
print (len(classes), "classes", classes)
# words = all words, vocabulary
print (len(words), "unique lemmatized words", words)
pickle.dump(words,open('words.pkl','wb'))
pickle.dump(classes,open('classes.pkl','wb'))
```

Create training and testing data

Now, we will create the training data in which we will provide the input and the output. Our input will be the pattern and output will be the class our input pattern belongs to. But the computer doesn't understand text so we will convert text into numbers.

```
# create our training data
training = []
# create an empty array for our output
output_empty = [0] * len(classes)
# training set, bag of words for each sentence
for doc in documents:
# initialize our 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.array(training)
# create train and test lists. X - patterns, Y - intents
train_x = list(training[:,0])
train_y = list(training[:,1])
print("Training data created")
```

BUILD A MODEL

We have our training data ready, now we will build a deep neural network that has 3 layers. We use the Keras sequential API for this. After training the model for 200 epochs, we achieved 100% accuracy on our model. Let us save the model as 'chatbot_model.h5'.

```
# 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'))
# Compile model. Stochastic gradient descent with Nesterov accelerated gradient gives good results for this model
sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
#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")
```

Predict the response (Graphical User Interface)

To predict the sentences and get a response from the user to let us create a new file ‘chatapp.py’.

We will load the trained model and then use a graphical user interface that will predict the response from the bot. The model will only tell us the class it belongs to, so we will implement some functions which will identify the class and then retrieve us a random response from the list of responses.

Again we import the necessary packages and load the ‘words.pkl’ and ‘classes.pkl’ pickle files which we have created when we trained our model:

```
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer =  WordNetLemmatizer()
import pickle
import numpy as np
from keras.models import load_model
model = load_model('chatbot_model.h5')
import json
import random
intents = json.loads(open('intents.json').read())
words = pickle.load(open('words.pkl','rb'))
classes = pickle.load(open('classes.pkl','rb'))
```

To predict the class, we will need to provide input in the same way as we did while training. So we will create some functions that will perform text preprocessing and then predict the class.

```
def clean_up_sentence(sentence):

# tokenize the pattern - split words into array

sentence_words = nltk.word_tokenize(sentence)
# stem each word - create short form for word
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
```

After predicting the class, we will get a random response from the list of intents.

```
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
def chatbot_response(text):
ints = predict_class(text, model)
res = getResponse(ints, intents)
return res
```

Now we will develop a graphical user interface. Let's use Tkinter library which is shipped with tons of useful libraries for GUI. We will take the input message from the user and then use the helper functions we have created to get the response from the bot and display it on the GUI. Here is the full source code for the GUI.

```
#Creating GUI with tkinter
import tkinter
from tkinter import *
def send():
msg = EntryBox.get("1.0",'end-1c').strip()
EntryBox.delete("0.0",END)
if msg != '':
ChatLog.config(state=NORMAL)
ChatLog.insert(END, "You: " + msg + '\n\n')
ChatLog.config(foreground="#442265", font=("Verdana", 12 ))
res = chatbot_response(msg)
ChatLog.insert(END, "Bot: " + res + '\n\n')
ChatLog.config(state=DISABLED)
ChatLog.yview(END)
base = Tk()
base.title("Hello")
base.geometry("400x500")
base.resizable(width=FALSE, height=FALSE)
#Create Chat window
ChatLog = Text(base, bd=0, bg="white", height="8", width="50", font="Arial",)
ChatLog.config(state=DISABLED)



#Bind scrollbar to Chat window
scrollbar = Scrollbar(base, command=ChatLog.yview, cursor="heart")
ChatLog['yscrollcommand'] = scrollbar.set
#Create Button to send message
SendButton = Button(base, font=("Verdana",12,'bold'), text="Send", width="12", height=5,
bd=0, bg="#32de97", activebackground="#3c9d9b",fg='#ffffff',
command= send )
#Create the box to enter message
EntryBox = Text(base, bd=0, bg="white",width="29", height="5", font="Arial")
#EntryBox.bind("<Return>", send)
#Place all components on the screen
scrollbar.place(x=376,y=6, height=386)
ChatLog.place(x=6,y=6, height=386, width=370)
EntryBox.place(x=128, y=401, height=90, width=265)
SendButton.place(x=6, y=401, height=90)
base.mainloop()
```

Run the chatbot

To run the chatbot, we have two main files; **train_chatbot.py** and **chatapp.py**. First, we train the model using the command in the terminal:

```
python train_chatbot.py
```

If we don't see any error during training, we have successfully created the model. Then to run the app, we run the second file.

```
python chatgui.py
```

The program will open up a GUI window within a few seconds. With the GUI you can easily chat with the bot.

SOFTWARE REQUIREMENTS

PYTHON

Python is an interpreted, high-level, general-purpose programming language. Created by Guidovan Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It has a wide range of applications from Web development (like: Django and Bottle), Scientific and mathematical computing (Orange, SciPy, NumPy) to desktop graphical user Interfaces (Pygame, Panda3D).

- Reason for increasing popularity
- Emphasis on code readability, shorter codes, ease of writing
- Programmers can express logical concepts in fewer lines of code in comparison tolanguages such as C++ or Java.
- Python supports multiple programming paradigms, like object-oriented, imperativeand functional programming or procedural.
- There exists an inbuilt function for almost all of the frequently used concepts.
- Philosophy is “Simplicity is the best”.

REQUIRED MODULES

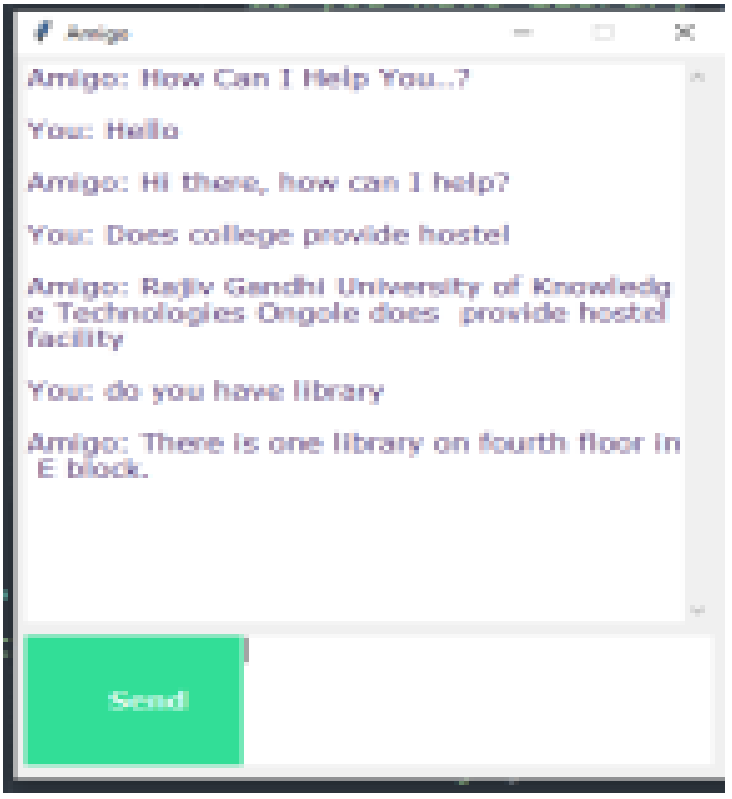
- **Nltk** NLTK is one of the leading platforms for working with human language data and Python, the module NLTK is used for natural language processing. NLTK is literally an acronym for Natural Language Toolkit.
- **Keras** Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. Keras is designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.

- Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code.
- **Json** Python has a built-in package called json, which can be used to work with JSON data. JSON (JavaScript Object Notation) is a lightweight data interchange format inspired by JavaScript object literal syntax (although it is not a strict subset of JavaScript).json exposes an API familiar to users of the standard library marshal and pickle module.

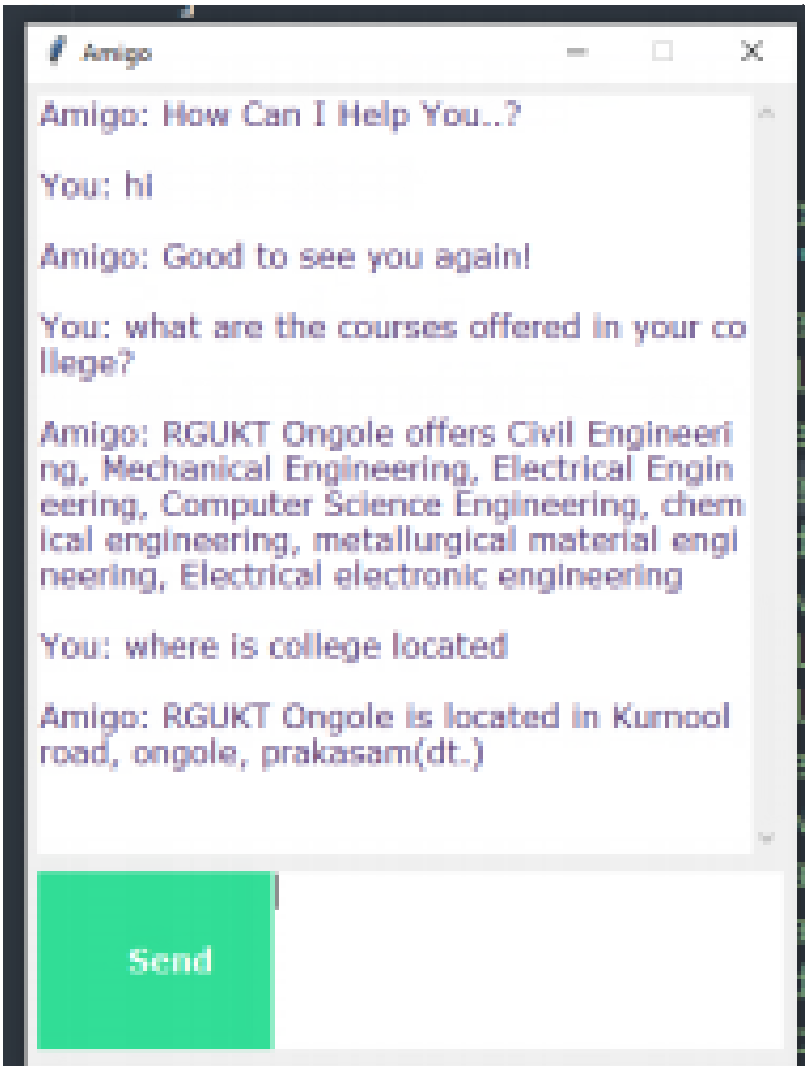
RESULT

RESULT-1:

RESULT-2:


Amigo
Amigo: How Can I Help You..?
You: Hello
Amigo: Hi there, how can I help?
You: Does college provide hostel
Amigo: Rajiv Gandhi University of Knowledg e Technologies Ongole does provide hostel facility
You: do you have library
Amigo: There is one library on fourth floor in E block.
Send

RESULT-3:


Amigo
Amigo: How Can I Help You..?
You: hi
Amigo: Good to see you again!
You: what are the courses offered in your college?
Amigo: RGUKT Ongole offers Civil Engineering, Mechanical Engineering, Electrical Engineering, Computer Science Engineering, chemical engineering, metallurgical material engineering, Electrical electronic engineering
You: where is college located
Amigo: RGUKT Ongole is located in Kurnool road, ongole, prakasam(dt.)
Send

APPLICATIONS

- This chatbot is used in colleges and universities.
- This chatbot can be modified and can be used in the following domains
    - Retail and e-commerce
    - Travel and hospitality
    - Banking , finance
    - Healthcare
    - Media and entertainment
    - Education

ADVANTAGES

Improves Query Handling System

People today expect fast online services with a timely response to their queries. They want complaints to be resolved quickly and expect companies to provide 24/7 customer support service. Millennials are even more eager. They expect you to reply instantly.

Here comes Chatbots. Chatbots can become a vital part of your admission counselling ecosystem. It can enhance user interaction with your educational enterprise resulting in good customer experience.

Besides the college website, the increasing number of students are tuning in to social media messaging to contact admission desk. While there are online and offline tools that will help you manage the inflow of students' queries but sometimes they're just not enough, particularly during the peak season of admission cycle.

Social media chatbots such as Facebook Messenger chatbot can share the influx of students messages by letting you automatically respond to their questions timely. They can be helpful in handling preliminary enquiries on courses offered, eligibility criteria, selection process and other frequently asked questions. This can help colleges to put their resources to handle other important concerns of prospective students thereby improving the overall productivity of the admission team.

Chatbot Marketing Give Boost to Admission

Chatbots help educational brands to establish meaningful touchpoints of engagement to connect with a broader potential audience. Notification bot helps in sharing important updates with your target audience. Successful deployment of chatbots can give huge benefits to educational entities in optimizing admission marketing efforts.

Chatbots can be deployed as a standalone bot on your digital assets such as website, app or through social platforms like Fb / Messenger etc.

You can treat your chatbot as a powerful content marketing tool to engage the audience. Online messenger bots are the best option to engage users on your digital platforms (website/mobile). Messenger bots can significantly improve counselling services offered by any academic institution. Messenger bots offer personalized messaging by recognizing users.

Visitors can be re-targeted by integrating it with your Ads giving a boost to your admission marketingdrive. Leads captured can be reached out again with more targeted messaging next time.

Use Chatbots as Powerful PR Tool for Online Reputation Management (ORM)

Chatbot marketing can be introduced in your college as a PR tool to take advantage of this fast and user-friendly medium of customer engagement.

A successful goal of this platform can be to create favourable connect between your audience and the institute. As a result of its longer-term benefit, it can help improve the overall brand image of your college.

Chatbots can be a huge asset for academic entities to handle online reputation marketing when the volume of queries is extremely high. This is especially needed in large universities and colleges offering multiple courses.

In this way, whether your prospective students are using a personal computer or a smartphone, they will remain accessible and hooked to your brand.

CONCLUSION

The goal of the system is to help the students to stay updated with their college activities. Artificial Intelligent is the fastest growing technology everywhere in the world, with the help of Artificial Intelligent and Knowledgeable database. We can make the transformation in the pattern matching and virtual assistance.

This system is developing chat bot based on android system so with the combination of Artificial Intelligent Knowledgeable database and virtual assistance. We can develop such chat bot which will make a conversion between human and machine and will satisfy the question raised by user. The main motive of the project is to reduce the work load on the college's office staff and reduce the response time to a user's query.

FUTURE SCOPE

To improve the current functionalities of College Enquiry Chatbot, in the future, the scope of the chatbot can be increased by inserting data for all the departments, training the bot with varied data, testing it on live website, and based on that feedback inserting more training data to the bot. Some of the new features which can be added to the bot are

1) Speech recognition feature through which students can ask their queries verbally and get the answers from the bot,

2) Integration with multiple channels such as phone call, SMS, and various social media platforms like Skype, Facebook and Twitter,

3) Handling context aware and interactive queries in which bot will be aware of the context of an ongoing conversation with a student,

4) Integration with services such as password reset and course 46 enrollment, and

5) Adding a capability for the bot to perform analytics based on user's sentiment based on which the bot can be re-trained on human emotions so that more empathy can be added to the bot.

REFERENCES

[1] “Chatbot”,en.wikipedia.org,2010 [Online]. Available: https://en.wikipedia.org/wiki/Chatbot. [AccessedJul. 10, 2018].

[2] "Chatbot: What is Chatbot? Why are Chatbots Important? -", Expertsystem.com, 2018. [Online]. Available: https://www.expertsystem.com/chatbot/. [Accessed Sep. 11, 2018].

[3] "Amazon.com: Online Shopping for Electronics, Apparel, Computers, Books, DVDs & more", Amazon.com, 2017. [Online]. Available: https://www.amazon.com/. [Accessed Sep. 12, 2018].

[4] "Amazon Alexa - Build for Amazon Echo Devices", Developer.amazon.com, 2017. [Online]. Available: https://developer.amazon.com/alexa. [Accessed Sep. 12, 2018].

[5] "Meet Erica, Your Financial Digital Assistant From Bank of America", Bank of America, 2018. [Online]. Available: https://promo.bankofamerica.com/erica/. [Accessed Sep. 12, 2018].

[6] C. Doug Gross, "Facebook: Get Messenger app or else - CNN", CNN, 2014. [Online]. Available: https://www.cnn.com/2014/07/29/tech/socialmedia/facebook-messenger/index.html. [Accessed Sep. 12, 2018].

[7] A. Storman, "5 Chatbot Challenges and How to Overcome Them", Chatbots Magazine, 2016. [Online]. Available: https://chatbotsmagazine.com/5-chatbotchallenges-and-how-to-overcome-them-caccc3a26d7c. [Accessed Sep. 11, 2018]

[8] A. M. Rahman, A. A. Mamun and A. Islam, "Programming challenges of chatbot: Current and future prospective," in 2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC), Dhaka, 2017.

[9] "Problems with Chatbots Today - Attunix", Attunix, 2018. [Online]. Available: https://attunix.com/problems-chatbots-today/. [Accessed Sep. 11, 2018].

[10] "Azure Bot Service - chatbot | Microsoft Azure", Azure.microsoft.com, 2017. [Online]. Available: https://azure.microsoft.com/en-us/services/bot-service/. [Accessed Jul. 22, 2019].