ABSTRACT

Over recent years we have seen various customs for conversational agents. Chatterbots are conversational agents where a computer program is designed to vitalize an intelligent conversation with the users [7]. These are virtual person who can effectively talk to any human being using interactive textual skills. These are programs that work on Artificial Intelligence (AI) & Machine Learning Platform [1]. NLP plays an important role in training the chatbot. The concept of chatbots came into existence to check whether the machines could trick users and make them think that they are actually talking to humans and not robots. As this idea grow popularly, many different companies started developing chatterbots for different needs [6]. There are many platforms to build the chatbot, this paper mainly concentrates on developing chatbot using python.

General Terms

Natural language processing, Artificial Intelligence.

Keywords

Artificial Intelligence (AI), Natural Language Processing (NLP).

1. INTRODUCTION

A chatbot is an instant messaging account that able to provide services using instant messaging frameworks with the aim of providing conversational services to users in an efficient and friendly manner. In technological view it is a set of queries, that contains predefined questions and answers leveraging Natural Language Processing (NLP). It is a python library which generates automated user responses. Chatbot is trained with a data set, from which it predicts the properties of the new data. At first it receives the user request, analyses the request and complies the response. Chatbots are different from other applications, because they do not have last seen, status, posts [8]. Figure 1, represents few different domains in which chatbots are used.

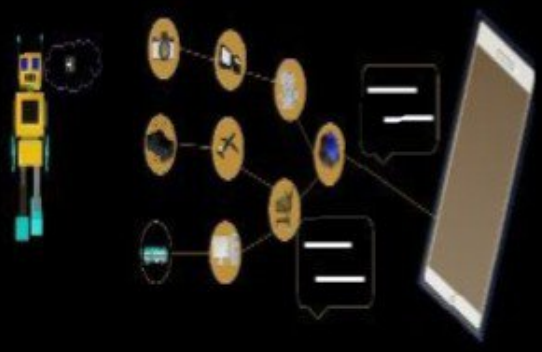

Figure 1: Chatbot in Different Domain

Main purpose of this chatbot is to reduce man power and to satisfy the frequently asked queries by the user. Chatbots can be attached to the web browser with different operating systems and also be used as an android application. Recent days, chatbots use machine learning algorithm to analyze and respond to the user. This analysis helps the bot to respond to each user query in a perfect manner. Sometimes, the response will not be so accurate for certain queries. This issue can be solved by moderate dataset, which can be trained to the bot.

This paper primarily focuses on the development of user friendly chatbot using python. This has a practical deployment in SCSVMV website.

2. SYSTEM ARCHITECTURE

The main purpose of this chatbot is to respond to user queries without man power. User can work with the chatbot in any web browser. The chatbot receives the request sent by the user, analyses it and responds to the user in return. This analysis is done,

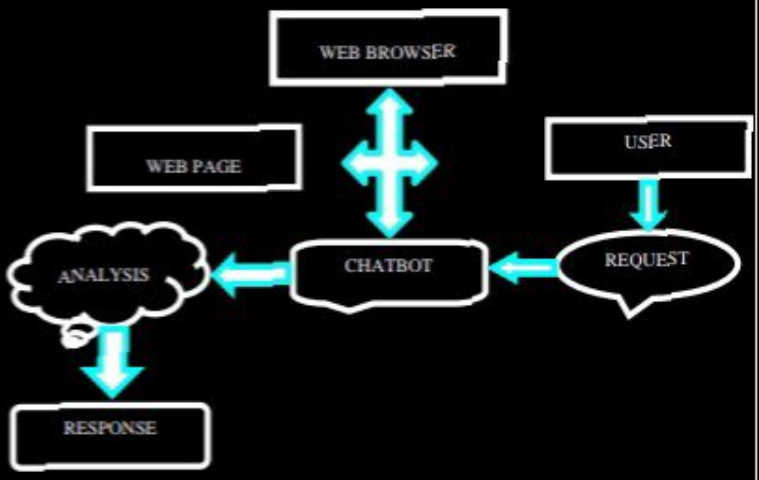

Figure 2: General Architecture of the Propose System

using the machine learning algorithm. The queries are predefined with a particular tag for each set. This tag is nothing but a keyword that helps the chatbot to analyze the user request. After the analysis, the chatbot responds to the user with required reply. If the user request is not clear to the chatbot, the response will be a default message defined by the developer.

Almost all the queries from the user will be clearly responded, only few are exceptional cases.

3. IMPLEMENTATION

This section describes the working of the system on an overall basis and further with specific focus on the software part of the chatterbot and the predefined query data set. An algorithm of the process, proceeded by a design motive of the system is also included.

3.1 Coding Part

The coding part is fully worked with python. This includes many library functions like NLTK, TensorFlow, NumPy and few other. These library functions help the chatbot to analyze the user request and decide the response to be given. Python itself has a package for chatbots, which is mainly required for the development of a user-friendly chatbot. Json is a python package that helps the query data set to get parsed by the Python code. This json file has the query with different tags. Each tag has a set of patterns and responses. This tag has default tag with it. The GUI is also developed using a Python package called tkinter. Tkinter is standard interface for GUI creations.

```
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)
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'])
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'))
```

Figure 3: Few Lines of Code.

```
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 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")
```

Figure 4: Few Lines of Code.

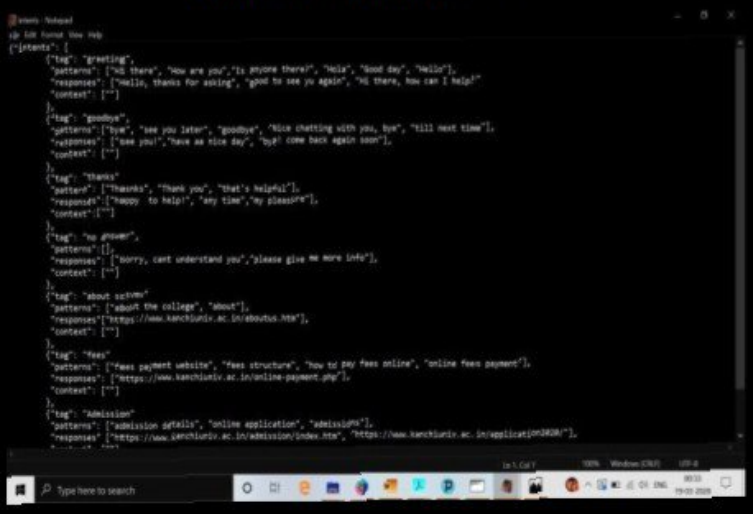

Figure 5: Json File Containing Tags with Patterns and Responses.

Figure 6: Working of the Code.

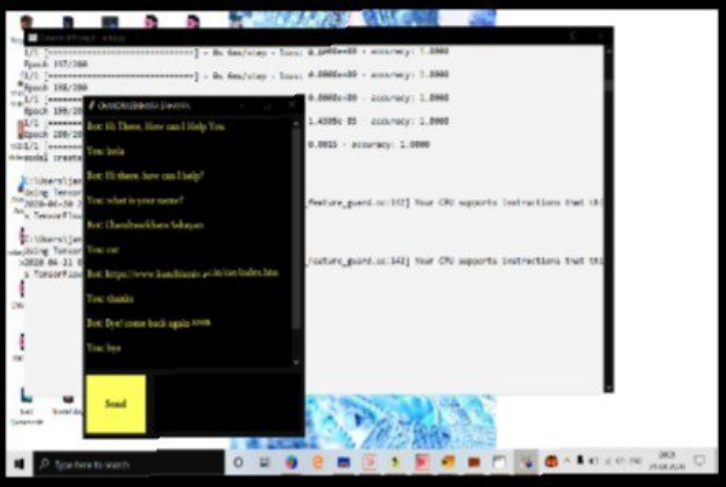

Figure 7: Output and the working model

3.2 Working Algorithm

Step 1) Start

Step 2) Select a data set, for which we need to develop a chatterbot.

Step 3) Prepare the set of tags with the patterns and responses.

Step 4) Install the required packages in Python.

Step 5) Train the Chatbot with predefined queries.

Step 6) Develop the GUI

Step 7) Execute the codes for the results.

Step 8) Stop.

3.3 Design Motive

- Reduce man-power: The user can easily be clarified with all his FAQ's within few minutes with the help of the chatbot.
- Accessibility: Any user can easily access the chatbot from any web browser.
- Overall friendliness: Ultimately, the chatbot is a user-friendly artificial machine that satisfies the user query at faster rate.

4. CONCLUSION

The goal of the project is to reduce man-power and to respond to user query at faster rate. Early days, the user's use to send a query mail to the particular site administrator and it would take few days for the site administrator to reply to the mails. Chatbots can overcome this delay, chatbot satisfies the user request or query immediately with relevant responses. These days many websites of banks, educational institutions, business sectors have developed their chatbots to satisfy user request in a faster time. Chatbots are user-friendly artificial

machines. This paper presents the overall development and working of the chatbot. This chatbot is being feasibility study under SCSVMV University.

5. FUTURE SCOPE

This project can be developed even more by adding multi-languages, speech recognition. We can add many more tags to the data set, as the website gets developed. The chat history of a particular user can be sent as a mail to him/her after the conversation is over. This can be done by authorizing the users and receiving their mail id's. This project is a small initiative to make the website user-friendly and easily understandable by the user.

Some of the recent works implemented to detect fake and fabricated news have been discussed in the following section. Machine learning and deep learning-based neural network models execute the identification and classification of real and fake news. For instance, in [5], the authors worked on fake news detection with the help of a mixed deep learning technique, CNN-LSTM. This paper has used the Fake News Challenge (FNC) dataset, which was created in 2017. They matched the claim with the news article body whether the claim matches with the article body or not. The authors have developed four data models. First, they use data without preprocessing, second with preprocessing. The authors obtained different results when they preprocessed data and when they did not. The third and fourth models are built on dimensionality reduction techniques by using PCA and Chi-square approaches. Finally, they trained on forty-nine thousand and nine hundred seventy-two samples and tested on twenty-five thousand and four hundred thirteen headlines and articles by CNN-LSTM. On their model with no knowledge of cleanup or preprocessing, the achieved accuracy was 78%. When preprocessing was done, the accuracy increased up to 93%. Next, the application of Chi-square raises the accuracy by 95%. Lastly, they conclude that using PCA with CNN and LSTM design resulted in the highest accuracy of 96%, significantly reducing the prediction time. In [6], T. Jiang *et al.* used baseline fake news identification techniques to locate the baseline methods' flaws and provide a viable alternative. First, the authors performed five completely different conventional machine learning models and three deep learning models to compare their efficiency. The authors used two datasets (ISOT and KDnugget) of various sizes to test the corresponding models' performance in this work. Finally, they take advantage of an adaptation of modified McNemar's check to decide if they are square measure contrasts between these two models' presentation, then determine the simplest model for detecting the fake news. The authors obtained

Training data.

"The training data for Rasa 2.0 is all formatted as .yml files. There are 3 types of training data.

- There is the main domain.yml file that contains a list of intents, entities, slots, responses and some actions/form configuration - these equate to what the bot thinks you mean (intent), the text the bot responds with (responses), the bots "memory" (slots and entities), and some setup if you're using the Action server above.
- The next file is an nlu.yml file - this is your training data, it lists all of your intents and the text, words, phrases that the bot can learn from and interpret as that specific intent. The easiest example is a "greet" intent. That would have things like - hi, hello, how are you, what's up?, how are you doing? as the training data and Rasa will process all that when you train.
- The last part of the training data is the stories.yml file - these are actual story flows that start with an intent and you script how you want the conversation to flow. These are fluid, they're not super strict "wired up" flows, but they help teach the bot how a conversation should look. If done properly, it will handle tangents (someone talking about one topic and switching to another and the bot will keep track). You can, and should, have as many stories as possible, different permutations and flows to give Rasa the best idea of a conversation. It'll surprise you with how it works, and will (again if done properly with enough data) converse very well.

To train the bot, it is as simple as typing - rasa train. and getting a cup of coffee or a beer depending on how big your bot is. In the beginning it'll train pretty quickly, but as you add more and more data, it'll take longer and longer. I have one that takes over an hour to train because of the data involved."[4]