A Proposed Cognitive Framework Model for a Student Support Voice based Chatbot using XAI

Debrita Ghosh (debmitra.ghosh@jisuniversity.ac.in)
Sayani Ghatak
Hrithik Paul

Research Article

Keywords: Artificial intelligence, LSTM algorithm, machine learning, Calling Agent, Sinch API

Posted Date: May 8th, 2023

DOI: https://doi.org/10.21203/rs.3.rs-2888180/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License
Abstract

As the calling agent directly calls and makes interaction (checking for valid phone numbers before calling) with the user such as using a pie event, even the agent can also record the conversation and transcribe information. The other voice chat agent is capable of taking voice input as well as text input. The user's information can be interpreted by it. And lastly, make a CSV or xlsx file of the user's input. The model has been designed in such a way that it can convert the voice provided as input by the user and analyze the sentiments contained in it by converting it into text using the speech-to-text module of the pypaudio package in Python. In the case of data cleaning, special characters have been removed from the .JSON file and provided to the NLTK tokenizer. When the model has been successfully built, users can enter their queries through voice and text, and the model will predict based on the text entered. The model first deciphers the voice entered quantitatively and ascertains the pitch, decibel, and amplitude, and enumerates a categorical cross-validation of the voice with the parameters that are set. It undergoes polarity checking under the positive and negative polarity counter and then the values of the positive and negative polarities of the text are stored in the program. When the agent finally works, it takes into account all the aspects with which it is trained, and provides the perfect outcome. If the user also enters the incorrect dialogue on their phone, the call will instantly cease. After that, we'll use an api post request to create and establish a connection. When it comes to encouraging different individuals to call for admission-related or medical-related reasons, the calling agent has a huge effect on that issue. In some cases, the user's number has been used. In this paper we have designed and implemented automation testing framework for testing web applications. This new automation testing framework has been implemented using selenium WebDriver tool. Using this framework tester can easily write their test cases efficiently and in less time. Tester need not to study the selenium web driver tool in detail. This framework is helpful to developer to analyze their code due to screen shot property of framework. This framework produces the customized test report to tester. It is very easy to maintain and repair the test suite for new release of the application using this framework. Software organizations are more committed to creating high-quality software solutions at lower costs. Previously, manual testing was used, but due to the increased complexity of the program, testing automation is now required. Selenium, GUI testing, Cross Browser testing, and Web automation are used more effectively. we used Selenium web driver Python to accomplish Web Automation testing.

I. INTRODUCTION

It is fairly difficult for persons to phone individual person at a time. Even in our hectic everyday lives, we have less time to complete multiple procedures at once. And contacting people, particularly for diverse goals such as admission, is highly essential and unique. As a result, an agent is required to resolve it by phoning automatically. It also determines whether the number from which the call will be made is correct or valid. If the number is true and legitimate, it requests an outgoing call to the target number over a rest API. we'll look for Python and rest API sinch. The limitations that we see nowadays, primarily in the education sector are as follows: A single person cannot call so many students at once. In order to
interpret the student's information, more people are needed. Calling, gathering information, and interacting with students can take more time. The AI agent makes it possible in a quick and efficient manner with fewer people involved. Thus, in order to call utilising the Sinch API, we must first create an account on the Sinch platform. When we establish an app, the platform will provide us with a key and secret key that we can use to fetch and obtain access to the REST API. We will seek for the virtual number before developing the app. Almost any internet-connected device can be used to converse with anyone in the world at any time using a virtual phone number. A virtual number is a phone number without a directly connected landline, commonly known as direct inward dialling (DID) or access numbers. These numbers are typically set up to route incoming calls to one of the pre-established phone numbers the client selects, whether it be fixed, mobile, or VoIP. A virtual number can function as a bridge between VoIP and PSTN communications. Virtual number subscribers can utilise their current phones without having to buy any new hardware by using one of the many pieces of software that are readily available. The api fetch link needs to be connected to our code when we have obtained the virtual number, key, and secret key. Therefore, anytime the call has been made, a callout request has been made. Additionally, a json file will load in the backend to manage the ttsCallout, also known as the "text-to-speech call-out." The to argument in the request body is called by this code after it submits a POST request to the Voice API's /callouts endpoint. Callouts are used by the Voice API to place voice calls.

In this paper, we'll demonstrate how to call a number, play a text-to-speech message, and then hang up using the ttsCallout or "text-to-speech callout." The "to" option must then be added; "to" will be our validated phone number. Ultimately, the trial account only allows us to call confirmed numbers; but, after we upgrade it, we will be able to call anyone. This is a significant restriction on our application. The word "locale" must be added after the "to" parameter. The text-to-speech call's language and location preferences. Language code according to ISO 639, dash, and country code according to ISO 3166-1 alpha-2 are used to specify locale. For instance, en-US stands for American English. Following the collection of all the parameters, we must add some details and a key for pyload. Pyload functions much like a json file, although it may also be thought of as a dictionary in Python. We will add the necessary keys and values to ttsCallout in the pyload. The model has been designed in such a way that it can convert the voice which is provided as input by the user and analyze the sentiments contained in it by converting it into text by using the speech to text module of the pyaudio package in Python. The model first deciphers the voice entered quantitatively and ascertains the pitch, decibel, and amplitude, and enumerates a categorical cross-validation of the voice with the parameters that are set. It undergoes polarity checking under the positive and negative polarity counter and then the values of the positive and negative polarities of the text are stored in the program.

It is observed that the decibels are unevenly distributed and hence, the model only takes into account the areas where the decibels are above the 0 value since reaching 0 would mean that the area contains no sound and as a result, it would not produce any text and no sentiments can be detected from that region either. For this reason, cross-validation of the sound is required for every voice that is entered as input. The model first recognizes the speech, that is the voice entered by the user. It is able to do so because we already have the speech recognition module of python. The external noise level is considered first
because the ambient noise quality always cannot be maintained at level 0 since there will be some sound in the background of the user. Hence, we have to set a threshold for the noise level and the energy of the input sound is adjusted according to the threshold which is read by the program. After the speech is recognized, it is translated to text by the model which is then split into tokens in order to analyze every bit of the input present to gain the presence of all the sentiments attached to it.

Here is how the paper is organized: Section "LITERATURE SURVEY" defines some existing state-of-the-art methods for the AI Agent. Therefore, section-III describes the necessary method that we have used in our research work. The section-IV discusses the Testing of our model. Results are described in section V. Then, a discussion on the applications and benefits of our proposed model is discussed in section VI and conclusion is described in section VII. Eventually, lastly we add references.

II. LITERATURE SURVEY

It is presented here that the author has designed and developed an intelligent voice-recognition chatbot. The paper presents a technology demonstration to demonstrate a proposed framework for supporting such an automated system (a Web service). In spite of the fact that a black box approach is employed, the Web-service allows all types of clients to communicate with the server from any platform since the communication structure is controlled at the Web-service level. Questions asked to the bot cannot be understood, so they are processed using a third-party expert system (an online intelligent research assistant), and the response is archived, improving the artificial brain capabilities for future generations of responses [1]. In this paper, sentiment analysis is used to identify whether customers are satisfied with the ASR system's performance. The purpose of this paper is to present approaches and techniques for using sentiment analysis to recognize user emotions in call centers [2]. Using only emotional information from people on social networks, this study predicts stocks based on the results of Nguyen et al. (2015). Due to the fact that we classify each user's message into one of the emotional categories "Strong Buy", "Buy", "Hold", "Sell", and "Strong Sell", the model can predict whether the stock value will increase or decrease the following day. CNN models and LSTMs were used [3]. Specifically, the purpose of this study is to qualitatively investigate symptom patterns and self-management responses, as well as quantitatively examine relationships between symptom severity and sentiment scores (a measure of emotional response to events). A total of 14 minority and nonminority teenagers (age 13–17) with controlled (50%) and uncontrolled asthma reported their asthma once daily over a period of 14 days using digital recorders. We used sentiment analysis to evaluate the emotional valence of diary entries and explore whether symptoms were associated with a greater level of negative sentiment based on symptom frequency, severity, and type[4]. Voice instructions are translated into text instructions by the application flow. An LSTM (Long-Short Term Memory) in the application takes care of the conversational strategy, analyses it, and provides answers to all questions posted by users. Additionally, it allows the visually impaired to get to their destination by identifying obstacles and detecting objects in their path. Utilizing the computational resources provided by cloud servers, such as location-specific resources, the application pushes all the data to the cloud server for reference and later access [5]. This research examines whether or not it is possible to utilize this technology in order to create a system that, based on
video generation, allows for the creation of a human actor capable of interacting with a user through voice communication in real time. The actor remains invisible while simultaneously interacting with the user. The work presents and tests a prototype system in addition to discussing motivations and ethics. With the prototype, an artificially synthesized video conversation will be created using the most recent real-time video manipulation software [6]. The purpose of this research is to develop a novel IOT-based application called Twilio using ROS. In this way, the central processing unit is able to monitor and control the different sensor modules and send appropriate notifications to the mobile phone by means of SMS messaging. The research aims to automate watering greeneries, reduce water wastage, monitor waste in dustbins, send alerts during emergencies like floods and fires, and implement a smart street light network using SMS services [7]. Incorporate steps such as Feature Engineering, Feature Selection, Exploratory Data Analysis (EDA), Model Training, etc. Various models are used to train the data, as shown below. Comparing the accuracy of LR, RF, and XGB models. The customer must provide the required information to apply for a loan. When all of the conditions are met and the provided data is accurate, a message related to the approval of the loan will be displayed. As a result, a message is displayed informing the user that the loan has been denied[8]. For example, an author may create a dashboard that monitors the temperature of a building remotely or controls the motors of a valve on a factory assembly line. There is no limit to the utility of this product. Finally, to complete their learning so far in this book, the reader is guided in the final section of this chapter to develop an advanced machine learning application by utilizing the techniques covered previously[9]. Through this system, students and parents will be able to ask questions and clear doubts through simple English language text messages or audio commands. Instead of congregating at the inquiry desk to ask questions related to admission procedures, students and parents will be able to communicate with a bot [10].

III. METHODOLOGY

When using Custom Callout, the first parameter is a method that lets us specify how the callout should behave. By declaring how the call should proceed at each call event, the server can control the custom callout, which the client initiates from the servers. Then, we will add the following keys to Custom Callout, such as "ice," "ace," "pie," and "dice," so that our calling agent can use them as webhooks. We use SVAML in those keys. Sinch created the call control markup language known as SVAML. Your server or application can provide an SVAML object in response to a callback event sent by the Sinch platform so that the voice call can be managed. This is how the SVAML object type is defined:

In Fig. 1, we can see that the server or application can provide an SVAML object in response to a callback event sent by the Sinch platform so that the voice call can be managed. The SVAML object is provided as we can see in the action section where each object can include only one section. The instructions are an array of objects which contain various tasks during the SVAML call.

We add each instruction and action to each key using SVAML. "Ice," the first key, is also known as an incoming call event. A POST request is made to the designated calling callback URL by the system when a call reaches the Sinch platform. Either an inbound data call or an incoming PSTN call can start this
event. Find out what activities and instructions are permitted here. The call is disconnected and an error message is played if the callback receives no response within the allotted time. When the call is answered by the callee, the "ace" event follows (person receiving the call). A POST request is being made to the designated calling callback URL. Then, it carries out a few tasks as directed. When we have the virtual number, key, and secret key, we need to attach the api fetch link to our code. Therefore, a callout request has been made each time the call has been placed. Additionally, a json file that manages the ttsCallout, commonly known as the "text-to-speech call-out," will load in the backend. This code sends a POST request to the Voice API's /callouts endpoint and then calls the to parameter in the request body.

There are several restrictions and verifications before automating calls. Therefore, the first client must send a verification request to the backend server, where the backend developer grants access so that the api agent can fetch and call the number when it has been validated. The verification report is sent to the backend server after the verification access is received through it, where the developer receives the verification result event. As well as the verification ID and outcomes, whether successful or unsuccessful. The client will then receive a success or failure notice on their device. And the Fig. 2 describe all the steps efficiently.

Callouts are used by the Voice API to place voice calls. In this tutorial, we'll demonstrate how to call a number, play a text-to-speech message, and then hang up using the ttsCallout or "text-to-speech callout." First and foremost, we need an algorithm and flow to create a fantastic interactive bot that can communicate with users and respond to user queries in both text and voice modes. Here, we use LSTM, also referred to as long short-term memory. After cleaning and preparing the data, we will create the models. As previously mentioned, we use a json file for the dataset. Then comes data preparation and cleansing. Word tokenizer is used to accomplish that first. Tokenization is a technique used in natural language processing to break down phrases and paragraphs into simpler language-assignable elements. The collecting of data (a sentence) and its breakdown into comprehensible components are the first steps of the NLP process (words). After tokenization, special characters will not be used. Therefore, we create a list of the special characters that will be disregarded and use lemmitizer to do so. By taking context into account, lemmatization reduces a term to its logical base form. We'll sort them afterward and compile a list of significant variance words. Then, we'll normalize it to make it more stationary, which will increase our accuracy and reduce loss. After completing all of this, the preprocessing of our dataset was completed successfully. After that, we'll use lstm to create our model. Long Short Term Memory Network, then, is essentially an upgraded RNN, a sequential network, that permits information to remain. It is capable of resolving the RNN's vanishing gradient issue. RNNs, also referred to as recurrent neural networks, are utilised for persistent memory. The first section determines whether the information from the preceding timestamp needs to be remembered or can be ignored. The cell attempts to learn new information from the input to this cell in the second section. After transmitting the current timestamp to the third section, the cell finally transmits the revised data. Gates refer to these three LSTM cell components. The Forget gate, Input gate, and Output gate are the names of the three components, respectively. With these cells and components, we created our LSTM model. For the input layers, we employ 2LSTM blocks with 64,128 units and "ReLu" activation. Additionally, we apply "softmax"
activation for the output layers to normalise a network's output to a probability distribution over expected output classes. Then we used the "Adam" optimizer with categorical crossentropy to create our model. We even try alternative optimizer, but Adam optimizer has the highest accuracy.

The comparison and loss between "Adam" and other optimizers are shown in the table below.

<table>
<thead>
<tr>
<th>Optimizer function</th>
<th>Accuracy</th>
<th>Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGD</td>
<td>98.11</td>
<td>0.0664</td>
</tr>
<tr>
<td>Adam</td>
<td>99.06</td>
<td>0.0341</td>
</tr>
<tr>
<td>Adadelta</td>
<td>5.66</td>
<td>2.9551</td>
</tr>
<tr>
<td>RMSprop</td>
<td>99.06</td>
<td>0.0399</td>
</tr>
</tbody>
</table>

In Table 1, we can see the different Accuracy and Loss values obtained under different optimizers which have been used to optimize the calling agent.

Next, we are going to see the graph demonstrating the increase in accuracy values along the passage of the phases by using the different optimizers:

In Fig. 3, we see among all of the optimizers only “Adam” and “RMSprop” have most accuracy but in case of loss “Adam” upper hand the “RMSprop”.

After developing the model, we use it in real life, where we accept voice and text input from consumers. For voice input, we employ tts engines and voice engines to convert the user's voice to text. Therefore, the speech recognition module has essentially played a part in listening to the user's voice. Basically, it turns on the microphone and then, anytime a user speaks into the microphone, the Google API retrieves data by employing recognition algorithms like Viterbi search and PLP characteristics. Additionally, we used noise reduction to eliminate non-stationary disturbances from user-generated data. After receiving the user's input, the api translates the speech into the appropriate text before sending the text as a parameter to the model. After passing it, the LSTM predicts the responses that will be the closest to the user's queries. Then, after anticipating the responses, the pyttsx3 module of the tts engine converts the text to the corresponding voices. As they send the user the same response and voice, user also receives the voice from user. In case of text, user text has send to as parameter and model just predict and give the reply. For the better understanding here is the flowchart of the agent:
In Fig. 4, the flowchart shows how the dataset was processed, starting with the lemmatizer, before being sent for the previously stated numericalization and normalisation, followed by the LSTM architecture and voice or text input.

If it is text, send it to the model for prediction; if it is not text, use voice instead.

The Google api then recognises the voice provided by the user. It is converted to text using the Google API and speech recognition, as was previously explained, and the text is then sent to the model. The model once more predicts the appropriate text. The TTS engine was then used to transform text to voice. In case of text, user text has send to as parameter and model just predict and give the reply. For the better understanding here is the flowchart of the agent:-

![Flowchart](image)

\[
\text{Positive Sentiment Score} = \frac{\text{No. of Positive Index} - \text{No. of Negative Index}}{\text{Total Indexes Available}}
\]  

(1)

The program uses the outcome of the positive and negative polarities to generate the percentage of positive and negative sentiment in the provided text. The positive sentiment score is determined by the formula given in Eq. (1). As we can see in Eq. (1) (Formula for calculating Positive Sentiment score), the Positive Sentiment score can be calculated by dividing the result of the difference between the number of positive indexes and the number of negative indexes by the total number of indexes available. The negative sentiment score is determined using the formula in Fig. 3

\[
\text{Negative Sentiment Score} = \frac{\text{No. of Negative Index} - \text{No. of Positive Index}}{\text{Total Indexes Available}}
\]  

(2)

As we can see in Eq. (2) (Formula for calculating Negative Sentiment score), the Negative Sentiment score can be calculated by dividing the result of the difference between the number of negative indexes and the number of positive indexes by the total number of indexes available. The model calculates the total sentiment score of the given voice as input using the formula given in Eq. 3(Formula for calculating total sentiment score).

\[
\sum_{i=1}^{n} ((\text{score(Neg)} + \text{score(Pos)}) + 1
\]

(3)

Equation 3 The model then produces the output in the layout which shows the values of the positive sentiment rating, neutral sentiment rating, negative sentiment rating, and the overall rating of the voice. These values are considered for analyzing the intention of the users who are providing their queries as inputs and whether they mean to say the words positively, negatively, or without any feelings at all.

**IV. Testing section**
Early detection of software defects is the goal of software testing. Software testing consumes 30 to 60 percent of all life cycle cost, depending on product criticality and complexity. With the development of internet technologies, web applications became more popular. Nowadays, a large number of software systems have been implemented as web applications. The quality of these web applications is one of the important factors while deploying these web applications. So, to increase the quality of software testing plays a vital role. Software development cycle becomes shorter and shorter; this makes the software testing more difficult. Human intervention is required in manual testing, which is a time-consuming process. So to avoid these problems, automation testing came into the picture. Automation testing means to automate the testing process or activities including design and execution of test scripts and use effective software automation tools. By automating software testing, we are able to improve the quality of software testing and minimize the need for human intervention. To support these tasks, there are various commercial and open-source tools available, such as Watir, JMeter, Selenium, QTP, and many others. The Selenium automation tool is considered the most popular and open-source tool for testing web applications. In this paper, we have proposed an automation testing framework based on the Selenium webdriver and TestNG tool. Selenium is composed of multiple software automation tools such as, Selenium IDE, Selenium RC (Selenium 1.0), and Selenium webdriver (Selenium 2.0).

V. RESULTS

For making different persons call for admission related, medical-related the calling agent has a great impact on that case. For these cases, the user's number has been used. A user from a different country can get the benefits of calling. But for trial version it is limited so we can call only for the users who are verified through api. The api uses a virtual number which is used "DID" system as it told before. For authentication purposes, the api gives a key and secret key which is also discussed before. Then we used different webhooks like “pie”, “ace”, “dice”, “ice” for interactive purposes. We used vscode as it is lightweight comparatively fast. For this experiment, the environment has been maintained in order to also analyze the performance and provide a similar setup to prevent any form of hardware bias. An Intel i3 microprocessor has been used along with Zubuntu Linux operating system. For coding, Visual Studio Code has been used for lightweight, faster, and smoother UI as well as support for multiple extensions. Jupyter extension has been deployed due to its feature of running blocks of code separately rather than running the entire code all at once. It makes it easier to debug the code. Python 3.10 was the language version for the development of the algorithm.

<table>
<thead>
<tr>
<th>Name</th>
<th>Numbers</th>
<th>Verified</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rinku</td>
<td>8250229731</td>
<td>Yes</td>
</tr>
<tr>
<td>Sayani</td>
<td>9007346098</td>
<td>Yes</td>
</tr>
</tbody>
</table>
From Table 2 above, we can see that there are various numbers and that, although we only used two for testing purposes, there may be more. The call will be placed at a different peak. The call is placed automatically, with no additional outside interference. When the administrator or developer gives the agent all of the mentioned verification numbers, the agent processes and notifies each individual person automatically after a 5-second break-in time. It does this by increasing stability and eliminating confusion caused by numbers that overlap in backend post requests.

**a. Test case 1:**

In Fig. 5,6,7, We utilized the number "8250229731" for the first test case, which the Sinch Rest API validated. We can observe from this graph that a call originated from a virtual number. The call will be on hold for the same amount of time—30–45 seconds—before being picked up from the user's end. the "pie" and "ace" webhooks are activated whenever the user accepts the call. wherein the agent will instruct the user in order to facilitate interaction.

Additionally, the user can engage with the user by using a prompt key event. When a call reaches the Sinch platform, the system sends a POST request to the selected calling callback URL. This event can be initiated by either an incoming PSTN call or an inbound data call. Learn what actions and guidelines are allowed here.

**b. Test case 2:**

Here the call comes automatically by the agent into the number, "90xxxxxxx8".

In Figs. 8 and 9, After the pickup the phone from the users end the same tts has been performed. Here, the user also provides the agent with input. The agent makes advantage of the previously stated "ace" and "pie" webhooks. Some instructions and actions have been put into place here using webhooks. Where the agent performs a separate action whenever the user enters a certain quantity or input, and if the user complies, another action is then triggered by webhooks in the backend. The input is used to collect information for the agent. by pressing a predetermined quantity of DTMF digits. The “dice” webhook callback is started when the call is ended. An attempt is being made to POST data to the specified calling callback URL. This event only supports the hangup action; instructions are not supported. If the user also enters the incorrect dialogue on their phone, the call will instantly cease. The user can press the erroneous key, the agent will explain, or the call will cease for choosing the incorrect prompt input key event. After that, we'll use an API post request to create and establish a connection.

**VI. DISCUSSION**

The admin and developer use the application fluently by using verified numbers for the user. Here, the admin just provides the numbers to the agent and the agent will perform its task by calling and making tts action. The tts action or text-to-speech action has been performed by the agent which is being written in the pyload as SVAML. We previously discussed SVAML how it is and what the impact is on the agent.
Sinch created the call control markup language known as SVAML. Our server or application can provide an SVAML object in response to a callback event sent by the Sinch platform so that the voice call can be managed. Through the SVAML the agent will perform its tts operation to make interaction with the user. The custom callout which is the main method in the pyload, is referred to make a different custom call to the different provided verified user. But as we know each application has some limitations. In our scenario, there is a great limitation that bound the sinch API and also the calling agent. Through our agent, we can perform automatically with a virtual number using a phone number without a directly connected landline, commonly known as direct inward dialing (DID) or access numbers. These numbers are typically set up to route incoming calls to one of the pre-established phone numbers the client selects, whether it be fixed, mobile, or VoIP. A virtual number can function as a bridge between VoIP and PSTN communications. Virtual number subscribers can utilize their current phones without having to buy any new hardware by using one of the many pieces of software that are readily available. As it is discussed before, we can’t perform or the agent will unable to make calls to the international user directly. To do such operations there needed some pre-verification. As we said earlier that our sinch rest API used the trial version account to fetch important information, webhooks, etc. So, to remove scam calls and also make more authenticate in the calling system the restrictions are provided to all trial versions of virtual numbers. For removing this trial version error we have to buy the virtual number or upgrade the account to premium. As a result of this, the virtual number will be fully verified and also considered as the main-virtual number, not for any test cases or trial, and make any verified number in internationally without needing pre-verification to account. Optimizers used to get the results:

- Optimizers are used to adjust the parameters for a model. Optimizers adjust model weights to maximize loss functions.

- The loss function is used as a way to measure how well the model is performing. When training neural networks, an optimizer is necessary.

- Here we used “SGD” optimizer with 0.98 and here is the comparison graph with others.

- In the case of activation function, it decides whether a neuron should be activated or not. Here we used “ReLu” for input layers and “Softmax” for output layers.

- The optimisation of the sentiment analysis of the AI agent has been done and tested using three different algorithms - LSTM, RNN and KNN. After undergoing 50/50 epochs with a batch size of 20, the training model has been observed to have gone through a drastic decrease in loss and an increase in the accuracy percentage value. This suggests that the analysis and prediction of the model have become higher and more accurate with the passage of the epochs. The highest accuracy is 92.5% and the minimum loss is 0.49.

- A comparison graph between the four different optimization techniques used for the sentiment analysis namely - SGD, J-48, Adam and GWO has been shown in the image below:

In the above Fig. 10, we can see that the highest amount of accuracy has been obtained on using the optimizer Adam. This is expected for sentiment analysis as we have seen in recent trends that the
accuracy levels have been consistent while optimising with Adam and it reaches the highest value for our model as well.

a. API and Engines:

- Sinch rest API
- Google cloud speech API
- Text to speech Engine
- TTS Engine, sapi5, NSS
- pyAudio package
- Speech to Text module
- Google Cloud Platform (GCP) Translate API
- DeepSpeech API

b. Analysis of Sentiments and providing results:

During the training and testing phases of our model of the AI agent, we have observed that it is able to accurately decipher the sentiments contained in a given audio file and correctly shows the result as the audio file being marked as Positive, Negative or Neutral. So, we can understand that if our final application runs with an accurate training model for the AI agent, we would be able to predict the users’ choice regarding the admission to an institute and analyze how they are inclined toward the system. The results shown by our training model can be seen in the image below:

In Fig. 11, we can observe that the agent provides the correct outcome for the different sentiments which are contained in the given input audio files. We can see that it is able to different perfectly between the Positive, Neutral and Negative voice files given to it.

c. Dependencies used for analyzing sentiments:

- Programming Language: Python
- pyAudio package
- Speech to Text module
- Google Cloud Platform (GCP) Translate API
- DeepSpeech API

Our model works on the scalable features of the voice available in audio format which it converts into text. The amplitude and decibel log is compared from the coordinates of the starting and end points of the input as can be observed from the text. These labels are read as tokens by the model and the spectrogram generated from these values gives us an idea of how far the prediction of the sentiments can go according to the given threshold of the model. We can see the spectrograms with amplitude and decibel logs respectively below:
From Fig. 12, it is observed that the amplitude is linear with the flow of sound with time and the sentiments will affect the streamlined flow only when the desired effect brings about a significant change in the amplitude. The decibel log on the other hand is directly proportional to the time as long as the voice remains unchanged and unhindered. Hence, the sentiments can be accurately detected once we take into account the entire area of the graph which contains the boundary of the streamlined flow of the sound.

d. Training and Testing phase of sentiment Analysis:

The model has to be prepared by loading the new dataset after the speech to text conversion is completed and then splitting it into two separate files for training and testing. The training model contains the CSV or XML file which contains all the data pertaining to the different genres of positive and negative voices. The training models help to find the best possible solution that we can apply to detect the sentiments correctly from the users' voices and which training model is fit to be used in the final application. So, to use the sentiment analysis model in our final application of the voice bot, we have to look very carefully at the training and testing phase of the sentiment analysis model to select the best functioning model. The loss and accuracy percentage of the model have to be checked and based on the values we get, we can decide whether the model is fit to be used or not. The accuracy percentage and loss of our training model are given in the table below:

<table>
<thead>
<tr>
<th>Training model</th>
<th>Epoch</th>
<th>Batch_size</th>
<th>Loss</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>best_model.hdf5</td>
<td>50/50</td>
<td>20</td>
<td>0.49</td>
<td>92.5%</td>
</tr>
</tbody>
</table>

In the above Table 3, it is observed that our model has achieved the highest accuracy which is 92.5%. After undergoing 50/50 epochs with a batch size of 20, the training model has been observed to have undergone a drastic decrease in loss and an increase in the accuracy percentage value. This suggests that the analysis and prediction of the model have become higher and more accurate with the passage of the epochs. If the accuracy remains low for any model, it will create a lot of problems for the users since it would not be able to correctly decipher the sentiments of the users and ultimately provide wrong results as the outcome which will jeopardize the system and the functioning of the application. In order to get the required results, we have used the algorithm LSTM. It has four different components, namely, Forget gate, Input gate, Output gate, and Cell gate. All the gates within LSTM have their respective functions to perform and their roles are important for getting the overall result. This approach helped us to achieve the highest percentage value of accuracy and minimize the loss value to a minimum. We can see the lowering of the loss value in our model from the picture below:

In Fig. 13, we can see that the loss percentage has been decreasing steadily for each epoch and ultimately goes on following this same trend to reach the minimum value.
In Fig. 14, we can see that the loss percentage value is at the minimum level and it does not reduce further. So, the threshold level is satisfied according to the highest accuracy and the lowest loss percentage. Hence, the training model can be accepted as the best fit for the final application.

A comparison graph between different algorithms has been shown below which have been used in the training and testing phases in order to improve the training model and gain the highest accuracy.

From the graph in Fig. 15, we can get an idea of the Comparison of different algorithms used during testing phase for sentiment analysis and how each technique is valuable for understanding the overall sentiments being deciphered and the functioning of the model. Here, we can observe that the highest value of accuracy is achieved by using LSTM algorithm which helps us to analyze the sentiments correctly and arrive at a perfect result.

**e. Time Complexity Analysis:**

The AI agent layout runs smoothly after satisfying all its dependencies. We have the Google Cloud Platform (GCP) Translate API and DeepSpeech API running simultaneously for just performing the sentiment analysis of the voice that it gains as input. Since the AI agent will be running on an LSTM algorithm for performing this sentiment analysis, the time complexity will be $O(n)$ for $n$ number of inputs. There are four gates in an LSTM algorithm, Forget gate, Input gate, Output gate, and cell gate all of which have their specific tasks to perform. The running time of all these gates is taken into account. Let us consider the number of inputs to be $n$ and the running time is $t$. Also considering $k$ as the neural layers on which the agent operates. Here, the runtime $t$ is always independent of $n$ because the time taken for running the model does not depend on the size of the input. Suppose 1 unit of time is taken per weight for accurately providing the output, the time complexity comes out to be $O(1)$ because it will be operating with one neural layer only hence it shows a unit time complexity. So, the total time complexity of the training model can be calculated as $O(k(2t + k + 3t))$.

**f. Testing section:**

In Fig. 16, we can see that when we type the name of an institute in the web application it provides an output stating “How can I help you”. This shows that the perception of the agent is quite well when we test using TestNG. TestNG provides some new functionality that makes it more powerful than JUnit. TestNG covers all categories of tests such as unit, functional, integration testing.

In Fig. 17, we can observe the different metrics that have been applied for the testing. These are categorized by TestNG and provides the report which lists all the dependencies as well as the performance of the overall system. In the proposed framework we have integrated TestNG with eclipse to generate the test report and execute multiple test cases in parallel. This TestNG report contains all the passed and failed test cases. TestNG report is very tedious to understand, so it requires some modifications. Each organization has different requirements about the test report. In proposed framework we have customized TestNG report according to organization requirements. So, organization can get the
test report as they want. This report also contains the link for failure test cases. By using this functionality, developer can easily find out defects in web application.

VII. CONCLUSION

We used real-time tests and “questionnaire” techniques to perform the usability of the voice-chat-agent. Also, the AI Agent gathers expected information from students so that we could make accurate predictions. Our model works on the scalable features of the voice available in audio format which it converts into text. The amplitude and decibel log is compared from the coordinates of the starting and end points of the input as can be observed from the text. These labels are read as tokens by the model and the spectrogram generated from these values gives us an idea of how far the prediction of the sentiments can go according to the given threshold of the model. For the calling agent, we will add more flexibility so that it can handle multiple international calls at once and speak the user’s preferred regional or other languages. Even so, every time a user sends a “request,” it will be activated as a call. As for free hosting, we utilize Streamlit, which is both free and simple to use, but in the future, we’ll use it for its own domain because it allows for one-click communication with users all over the world. The calling agent will be immensely helpful to any educational organization because the primary cause of concern for every institute is the analysis of students’ attitudes. Students are always worried about which institute is going to be right for them and making a perfect choice would be easier once their queries are resolved. The institutes would be able to predict which way a particular student wants to go and which stream they want to join since the agent will analyze their sentiments and provide a result that would help to ascertain whether they want to take admission or not.

Declarations

Competing interests: The authors declare no competing interests.

References


Figures

Figure 1

<table>
<thead>
<tr>
<th>instructions</th>
<th>Array of objects (svaml.instruction)</th>
</tr>
</thead>
<tbody>
<tr>
<td>The collection of instructions that can perform various tasks during the call. You can include as many instructions as necessary.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>action</th>
<th>object (svaml.action)</th>
</tr>
</thead>
<tbody>
<tr>
<td>The action that will control the call. Each SVAML object can only include one action.</td>
<td></td>
</tr>
</tbody>
</table>
SVAML call and action

Figure 2
Flowchart of Verification process

Figure 3
Graph of Accuracy vs Optimizers
Figure 4
Flowchart of the agent
Figure 5

calling coming from agent
Figure 6

Validating Test case 1
Figure 7

Test case 2
**Figure 8**

Users picking phone
Figure 9

User provides input
Figure 10

Optimizers used during Sentiment Analysis
<table>
<thead>
<tr>
<th>Filename</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>112.wav Neutral</td>
</tr>
<tr>
<td>1</td>
<td>113.wav Neutral</td>
</tr>
<tr>
<td>2</td>
<td>115.wav Positive</td>
</tr>
<tr>
<td>3</td>
<td>119.wav Positive</td>
</tr>
<tr>
<td>4</td>
<td>123.wav Neutral</td>
</tr>
</tbody>
</table>

**Figure 11**

Results shown by the training model
**Figure 12**

Spectrograms with amplitude and decibel log
Figure 13

Improvement of loss percentage
Figure 14

Improvement of loss percentage to the minimum value
Figure 15

Comparison of different algorithms used during testing phase for sentiment analysis

Figure 16

Results from Selenium web driver
<table>
<thead>
<tr>
<th>Question Type</th>
<th>Performance</th>
<th>Devices</th>
<th>Working</th>
<th>Question Type</th>
<th>Performance</th>
<th>Device</th>
<th>Working</th>
</tr>
</thead>
<tbody>
<tr>
<td>Admission</td>
<td>Good</td>
<td>Pc</td>
<td>Yes</td>
<td>Admission</td>
<td>Good</td>
<td>Mobile</td>
<td>Yes</td>
</tr>
<tr>
<td>Greetings</td>
<td>Bad</td>
<td>Pc</td>
<td>Yes</td>
<td>Greetings</td>
<td>Bad</td>
<td>Mobile</td>
<td>Yes</td>
</tr>
<tr>
<td>Goodbye</td>
<td>Bad</td>
<td>Pc</td>
<td>Yes</td>
<td>Goodbye</td>
<td>Bad</td>
<td>Mobile</td>
<td>Yes</td>
</tr>
<tr>
<td>Help</td>
<td>Good</td>
<td>Pc</td>
<td>Yes</td>
<td>Help</td>
<td>Good</td>
<td>Mobile</td>
<td>Yes</td>
</tr>
<tr>
<td>About</td>
<td>Very Good</td>
<td>Pc</td>
<td>Yes</td>
<td>About</td>
<td>Very Good</td>
<td>Mobile</td>
<td>Yes</td>
</tr>
<tr>
<td>Timing</td>
<td>Good</td>
<td>Pc</td>
<td>Yes</td>
<td>Timing</td>
<td>Good</td>
<td>Mobile</td>
<td>Yes</td>
</tr>
<tr>
<td>Address</td>
<td>Bad</td>
<td>Pc</td>
<td>Yes</td>
<td>Address</td>
<td>Bad</td>
<td>Mobile</td>
<td>Yes</td>
</tr>
<tr>
<td>Email</td>
<td>Bad</td>
<td>Pc</td>
<td>Yes</td>
<td>Email</td>
<td>Bad</td>
<td>Mobile</td>
<td>Yes</td>
</tr>
<tr>
<td>Courses</td>
<td>Good</td>
<td>Pc</td>
<td>Yes</td>
<td>Courses</td>
<td>Good</td>
<td>Mobile</td>
<td>Yes</td>
</tr>
<tr>
<td>Cost</td>
<td>Bad</td>
<td>Pc</td>
<td>Yes</td>
<td>Cost</td>
<td>Bad</td>
<td>Mobile</td>
<td>No</td>
</tr>
<tr>
<td>Training Partners</td>
<td>Good</td>
<td>Pc</td>
<td>Yes</td>
<td>Training Partners</td>
<td>Good</td>
<td>Mobile</td>
<td>Yes</td>
</tr>
<tr>
<td>Placement Partners</td>
<td>Good</td>
<td>Pc</td>
<td>Yes</td>
<td>Placement Partners</td>
<td>Good</td>
<td>Mobile</td>
<td>Yes</td>
</tr>
<tr>
<td>Criteria</td>
<td>Bad</td>
<td>Pc</td>
<td>Yes</td>
<td>Criteria</td>
<td>Bad</td>
<td>Mobile</td>
<td>No</td>
</tr>
<tr>
<td>Website</td>
<td>Good</td>
<td>Pc</td>
<td>Yes</td>
<td>Website</td>
<td>Good</td>
<td>Mobile</td>
<td>Yes</td>
</tr>
<tr>
<td>Campus</td>
<td>Good</td>
<td>Pc</td>
<td>Yes</td>
<td>Campus</td>
<td>Good</td>
<td>Mobile</td>
<td>Yes</td>
</tr>
<tr>
<td>Hostel</td>
<td>Bad</td>
<td>Pc</td>
<td>Yes</td>
<td>Hostel</td>
<td>Bad</td>
<td>Mobile</td>
<td>Yes</td>
</tr>
<tr>
<td>Extracurricular Activity</td>
<td>Good</td>
<td>Pc</td>
<td>Yes</td>
<td>Extracurricular Activity</td>
<td>Good</td>
<td>Mobile</td>
<td>Yes</td>
</tr>
<tr>
<td>Scholarship</td>
<td>Good</td>
<td>Pc</td>
<td>Yes</td>
<td>Scholarship</td>
<td>Good</td>
<td>Mobile</td>
<td>Yes</td>
</tr>
<tr>
<td>JET</td>
<td>Good</td>
<td>Pc</td>
<td>Yes</td>
<td>JET</td>
<td>Good</td>
<td>Mobile</td>
<td>No</td>
</tr>
</tbody>
</table>

**Figure 17**

Dataset showing the different metrics used for testing