Speech-Based Virtual Travel Assistant For Visually Impaired

N. Sripriya,
Department of Information Technology,
SSN College of Engineering,
Chennai, India
sripriyan@ssn.edu.in

S.Poornima,
Department of Information Technology,
SSN College of Engineering,
Chennai, India
poornimas@ssn.edu.in

S.Mohanavalli,
Department of Information Technology,
SSN College of Engineering,
Chennai, India
mohanas@ssn.edu.in

R.Pooja Bhaiya
Department of Information Technology,
SSN College of Engineering,
Chennai, India

V. Nikita,
Department of Information Technology,
SSN College of Engineering,
Chennai, India

2020 4th International Conference on Computer, Communication and Signal Processing (ICCCSP) | 978-1-7281-6509-7/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICCCSP49186.2020.9315217

Abstract— **Visually-impaired people often feel handicapped and find it difficult to explore the world around them freely without any kind of help and assistance. This lack of independence restricts them from feeling confident about themselves. Furthermore, all virtual travel assistants currently existing in the market, are chatbots associated with textual interaction. While this may be useful more often than not, in certain situations, users may find textual interaction uncomfortable and prefer a conversational interface over text. An important issue of concern in the tourism industry is the monetary expense invested in travel guides by tourists. A travel assistant capable of acting as a virtual guide can help reduce this investment by a large factor. Hence, it is proposed to develop a speech-based travel bot capable of acting as a virtual tour guide. The bot will play the role of a tour guide by suggesting places and giving information about the place such as opening hours, rating, address to aid the user in knowing more about the place by interacting with the user and providing relevant information. The proposed system can also find great use in situations where a person may feel uncomfortable to text such as when driving. This speech-based virtual travel assistant/bot is implemented using speech recognition, speech synthesis, and natural language techniques to gather the user's preference and provide the intended output. The developed bot has proved to be efficient for searching for different kinds of places and interacts well with the user and helps to find further details about a specific place as per the user's queries. The intents for most of the queries are correctly recognized, which helps in efficient dialogue management.**

Keywords— Virtual Assistant, Speech Recognition, Speech Synthesis, Natural Language Processing, Intent Classification

I. INTRODUCTION (*HEADING 1*)

The process of getting in touch with a company through a representative at the call center has always been a slow, tedious and frustrating approach owing to the speed, accuracy in understanding the user's query and responsiveness. Advancements in technology have opened doors to faster and easier solutions with the uprise of chatbots in the digital market. Chatbots can converse with customers without being completely limited by technology since the AI technology is built into the software. These bots have the ability to dig through huge amounts of data to pick out the best piece of information for a customer, be it a troubleshooting solution or a recommendation for a new product to try.

Speech bots are a form of chatbot that provide seamless and highly personalized assistance to users. Speech bots help in organizing various aspects of your life with reminders, appointment booking, tickets, etc., all through a human-like interaction. This leaves less room for human-errors and increases overall efficiency and also makes way for better productivity in your enterprise. To set up a conversational interface, these bots use natural language processing to receive voice-based inputs from the users. The bots then respond to the queries with relevant information.

A chatbot is meant to understand a human language, respond to it like a human and learn along the way to behave more humanely. Siri and Google assistant are classic examples of chatbots. Some common examples of chatbot applications are:

- A restaurant allowing customers to order using a chatbot, either in the store or at home.
- A retail store offering promotions for customers in the shopping mall via the chatbot
- A chatbot that allows customers to book flights and receive relevant information when they are in the airport.
- A chatbot that helps customers make e-commerce purchases.

Chatbots can have certain advantages over human agents such as round-the-clock availability, access to a broad array of information and functionality and exemplary performance in terms of speed and accuracy. Voice bots suit the modern consumer's lifestyle extremely well. If your business use case is a conversational application, like ordering food/groceries or even customer support, then using a voice bot would be most apt.

The major difference between a messenger chatbot and a voice bot is the way we interact with them. A text-based messenger chatbot exists on one or more messaging platforms, including both SMS and web-based messengers and users primarily interact through a screen via text or button presses. A user's interaction with a voice-enabled bot is different: they converse with such a bot via their voice in natural language. The voice bot then answers back using pre- recorded messages, text-to-speech responses or a mixture of both.

Visually-impaired people often feel handicapped and find it difficult to explore the world around them freely without any kind of help and assistance. This lack of independence restricts them from feeling confident about themselves. Furthermore, all virtual travel assistants currently existing in the market, are chatbots associated with textual interaction. While this may be useful more often than not, in certain situations, users may find textual interaction uncomfortable and prefer a conversational interface over text. An important issue of concern in the tourism industry is the monetary

expense invested in travel guides by tourists. A travel assistant capable of acting as a virtual guide can help reduce this investment by a large factor.

Our proposed system is a speech-based travel bot capable of acting as a virtual tour guide. The bot will play the role of a tour guide by suggesting places and giving information about the place such as opening hours, rating, address to aid the user in knowing more about the place by interacting with the user and providing relevant information. The proposed system can also find great use in situations where a person may be uncomfortable such as when driving.

The paper has been organized into the following sections. Section II gives a literature survey about other related work similar to the proposed system. Section III explains the various technical concepts and algorithms used for the implementation of the system in detail. Section IV explains the implementation of each module and its functionality. Section V throws light on the evaluation results for each module of the proposed system Section VI gives a conclusion and information about possible future work in the system.

II. Literature Review

Bots are emerging at a fast rate. There have been many bots which provide assistance to travellers, even for basic functions such as searching and booking hotels to stay. The following section talks about some of the existing travel bots and their functionalities.

A. *Expedia Facebook Messenger Bot: An experimental hotel search chatbot*

The Expedia bot for Facebook Messenger provides a convenient interface for travelers to browse hotel options and make a booking. This bot was developed by the Expedia Group to aid users in hotel bookings. The bot opens up with a simple "Get Started" message allowing the customer to proceed with looking for hotels. The bot interacts with the user by asking three primary questions: location, check-in date and the number of nights for a stay. The understanding capabilities of this bot are limited to predefined answer templates which require the user to answer to the point with a single word/number.

The entire conversational interface of the bot is predefined with a template and hence the bot does not customize itself to the user. Also, the auto-correction capability of the bot works aggressively by typecasting user inputs to closest city names irrespective of the type of input. The bot displays the outputs to the user's queries in the form of 5 cards displaying information, images, prices, and links to the official websites. Once a user has successfully made their booking, the bot immediately returns the user's travel itinerary to the Messenger window. The bot also allows a user to restart the conversation with the keyword "restart".

B. *Hello Hipmunk: A virtual travel agent that combs your email and calendar to create personalized travel recommendations*

Hello Hipmunk is a free virtual travel agent available on Facebook Messenger and Slack. This bot has two major features: Hello Email and Hello Calendar. Using Hello Email, the user loops hello@hipmunk.com into your email conversations with travel queries. Hipmunk replies, in conversational, friendly emails, with flight options and also performs hotel searches. This bot also allows us to create alerts for all our travels and take general travel advice.

The conversational interface for the bot is a simple card with buttons for looking up flights, hotels and travel advice. The bot interacts with the user through question answers and often puts forth additional questions for more clarity on the user's query. Results are displayed as cards with images, prices, and links to the website. 8 This bot is capable of understanding complex user inputs as single words/complete sentences.

C. *MedWhat: Making Medical Diagnoses Faster*

This chatbot aims to make medical diagnoses faster, easier, and more transparent for both patients and physicians. MedWhat is powered by a sophisticated machine 11 learning system that offers increasingly accurate responses to user questions based on behaviors that it "learns" by interacting with human beings. In addition to the ever-growing range of medical questions fielded by MedWhat, the bot also draws upon vast volumes of medical research and peer-reviewed scientific papers to expand upon its already considerable wealth of medical expertise. MedWhat is much closer to a virtual assistant rather than a conversational agent.

D. *Woebot- The Self-care expert*

Woebot, an artificially intelligent chatbot, uses the principles of cognitive behavioral therapy. The world's first mental health chatbot, Woebot was founded in 2017 for young adults in college and graduate school. Designed to use natural language processing, therapeutic expertise, excellent writing, Woebot comes with "occasional dorky joke". The therapeutic framework is said, "to create the experience of a therapeutic conversation for all of the people that use him".

Woebot aims to create an environment that is more conversational and less 12 automated. The chatbot relies on scientifically validated approaches to mental health, provides insights and teaches stuff through cognitive-behavioral therapy. Woebot acknowledges that there will always be a need for a human connection from therapists and it does not intend to replace the human connection. But given the fact that there are millions of people around the world that will never see a therapist, chatbots like Woebot could be of immense help.

III. Proposed System

As depicted in the following figure, the user speaks out a query to the system, which is recognized by the recognizer and the speech signals are converted to text. The text is used as the input and processed to understand what the user wants, and once the response is generated it is forwarded to the speech synthesizer which is used to read it out to the user.

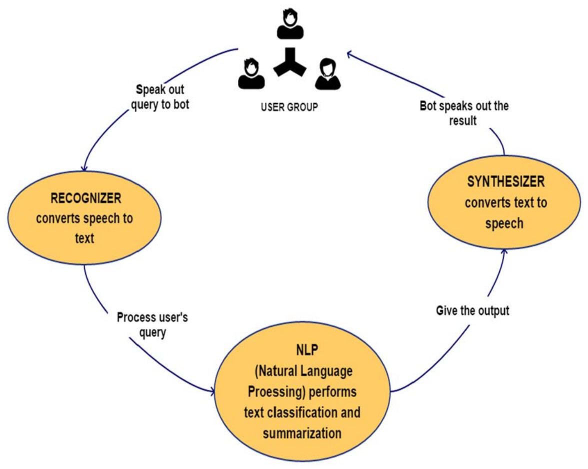

USER GROUP
Speak out query to bot
Bot speaks out the result
RECOGNIZER converts speech to text
SYNTHESIZER converts text to speech
Process user's query
Give the output
NLP (Natural Language Proessing) performs text classification and summarization

Fig. 1. Example of a figure caption.

A. Speech Recognition

Speech recognition is the process of converting a speech signal into a sequence of words. Speech must be converted from physical sound to an electrical signal with a microphone, and then to digital data with an analog-to-digital converter. Once digitized, several models can be used to transcribe the audio to text. Most modern speech recognition systems rely on what is known as a Hidden Markov Model.

In the proposed system, the speech recognition is done using Python's SpeechRecognition package. It takes the audio input from the microphone and returns the spoken words in the form of a string.

B. Speech Synthesis

Speech synthesis is the artificial production of human speech. A text-to-speech (TTS) system converts normal language text into speech; other systems render symbolic linguistic representations like phonetic transcriptions into speech. Synthesized speech can be created by concatenating pieces of recorded speech that are stored in a database. A synthesizer can incorporate a model of the vocal tract and other human voice characteristics to create a completely "synthetic" voice output.

The quality of a speech synthesizer is judged by its similarity to the human voice and by its ability to be understood clearly. There are essentially three stages involved in converting the written text into speech, namely text to words, words to phonemes, and phonemes to sound. Finally, the sound is read out. To implement text-speech conversion we have used Python's Pyttsx3 package in our system. Pyttsx3 is a cross-platform speech library for Python 3+.

C. Natural Language Processing

Natural Language Understanding refers to the computer's ability to understand what the user is speaking. It bridges the gap of communication between a human and a computer. It enables a computers to read the text, hear the speech, interpret it, measure sentiment and determine which parts are important..

Every bot is expected to go beyond statistical text and information-retrieval methods to develop contextually meaningful responses. Virtual assistants require articulated knowledge about the possible meanings of words and phrases, connected with each other and with the real-world context. They must continue to improve their language models, but also start to gather information to create richer situational awareness, understand both individual and general context, and become attuned to the intent of the questioner. Hence Natural Language Processing is very important for adding a human touch and to make the user feel that the bot is understanding his queries and needs. Natural Language Processing is what allows the bots to understand your messages and respond appropriately.

To efficiently perform Natural Language Understanding the bot needs to be created and trained based on sets of data. Machine learning lets bots develop a growing set of knowledge and understanding, by studying the training examples. NLU's job is to take this input, understand the intent of the user and find the entities in the input. Suppose the user says "I want to go to a chinese restaurant". In the above sentence, the intent is searching for a restaurant and the entity is chinese. Hence we need to train our bot by using a custom dataset. This training will build a model which will then convert the data into a structured format consisting of entities and intents. We feed in the training json file with few configuration details, and we would get trained ML models at the end of training.

The bot is trained for classifying using the Support Vector Classifier along with Gridsearch for parameter optimization. Support vector classifier is a supervised learning method for classification, which constructs a hyperplane in multidimensional space to separate different classes. It aims at generating a maximum marginal hyperplane which best divides the dataset into two classes. When a query from the user is taken as input, the intent of the user has to be found. Hence to classify the query into one of the intents, the above mentioned algorithm is used. Gridsearch is used to optimize the parameters taken for classification. It methodically builds and evaluates a model for each combination of parameters and in the end retains the best configuration.

IV. Implementation

We propose to develop a speech based virtual assistant for travellers, which helps them find nearby places such as restaurants, temples, bakeries and a lot more. It also provides various specific details, like rating, opening hours, address, contact number, etc., about the places and a short summary if required. The proposed methodology is majorly divided into three phases :

- Speech Recognition
- Speech Synthesis
- Natural Language Processing

These three phases work with each other to interact with the user and to help user find the required place and also answer users queries. The Google API- Places API is used to retrieve the places as well as the place details as per the user's request.

A. Speech Recognition Module

The main task in this phase involves recognising what the user is speaking and converting it to text such that it can be used for further processing. We use Python's Speech Recognition library to understand the audio and transcribe it. This library supports various other APIs and engines for speech recognition just as a function call. We also need to install an additional package PyAudio to get the input via a microphone.

The following steps are performed using the library to transcribe the audio to text:

- An instance of the Recognizer class is initialized as shown below

```
r = sr.Recognizer()
```

- We use the default system microphone with the help of the PyAudio package as the source of input. This is accessed by creating an instance of the Microphone class and then using using the listen() method of the Recognizer class inside of the with block.
- To handle ambient noise, we use the adjust_for_ambient_noise() method of the Recognizer class

```
with sr.Microphone() as source:
r.adjust_for_ambient_noise(source)
audio = r.listen(source)
```

- And finally the recognize_google() function of the Recognizer class is used to retrieve the text form of the speech input. The audio is passed as a parameter,and the function returns the string.

```
youSaid= r.recognize_google(audio)
```

B. Speech Synthesis Module

Speech synthesis refers to synthesizing the text into a speech format. This gives the user a feeling that the bot is talking to him or her. In this phase we take in a text stringwhich has to be spoken out to the user, and this text is synthesized into a human sound. We use the cross-platform library, pyttsx3, to convert the text to speech. The pyttsx3 module supports native Windows and Mac speech APIs making it one of the most useful package.

To synthesize the text into speech using the pyttsx3 library the following steps have to be followed:

- Initialize the pyttsx.Engine instance by invoking the pyttsx.init() factory function which gets a reference to the Engine interface and generates an instance of it.

```
engine = pyttsx3.init()
```

The bot uses the engine object to register and unregister event callbacks; produce and stop speech; get and set speech engine properties; and start and stop event loops.

- Pass the string as input parameter to the say() function which synthesizes the text based on the properties set in the interface.

engine.say("Hello welcome")

In the above code snippet "Hello welcome" is given as input text which is synthesized as per the default voice and rate properties if not overwritten in the code.

C. Processing Module

This phase is the crux of the bot. Any bot should be intelligent to understand and interpret what the user is trying to say, and also provide appropriate response. In this phase there are two major tasks:

- Intent Recognition and Classification
- Response Generation

D. Intent Recognition And Classification

In this phase, the job is to take the input string from the speech recognition module, understand the intent of the user and find the entities in the input. Suppose the user says "I want to go to a chinese restaurant". In the above sentence, the intent is searching for a restaurant and the entity is chinese. Hence we need to train our bot by using a custom dataset. This training will build a model which will then convert the data into a structured format consisting of entities and intents.

Before feeding the data for training, the sentences need to be preprocessed so that it is in a format that the training model understands. We use the Rasa Stack for doing the preprocessing and classification. .The Rasa Stack is a set of open source machine learning tools for developers to create contextual chatbots and assistants. It doesn't have any pre- built models on a server that can be called using an API, hence we can train our models the way we require. The preprocessing in Rasa is done as in the following steps :

- Tokenization: This step converts each training sample from your training file and converts them into a list of tokens(words). At the end of this step, we have a bag of words.
- Featurization: Now the bag of words can be fed into the machine learning algorithm for training. However, an ML algorithm understands only numerical data. It is the featurizer's job to convert tokens into word vectors. At the end of this step, we will have a list of numbers which will make sense only for ML models. This step is done using the Word2Vec model. It is performed in the backend by Spacy. Word2Vec is a group of models which helps derive relations between a word and its contextual words.

Finally after these steps we have the word embeddings. Word embeddings are vector representations of words, meaning each word is converted to a dense numeric vector. Word embeddings capture semantic and syntactic aspects of words. This means that similar words should be represented by similar vectors. Word embeddings are specific for the language they were trained on. Rasa NLU takes the average of all word embeddings within a message, and then performs a gridsearch to find the best parameters for the support vector classifier which classifies the averaged embeddings into the different intents.

E. Training Data for Proposed Bot

The dataset is written in json format as in the figure below. Each object has a text which corresponds to the input user can give. The intent attribute specifies what intent the text of given kind should be mapped to.

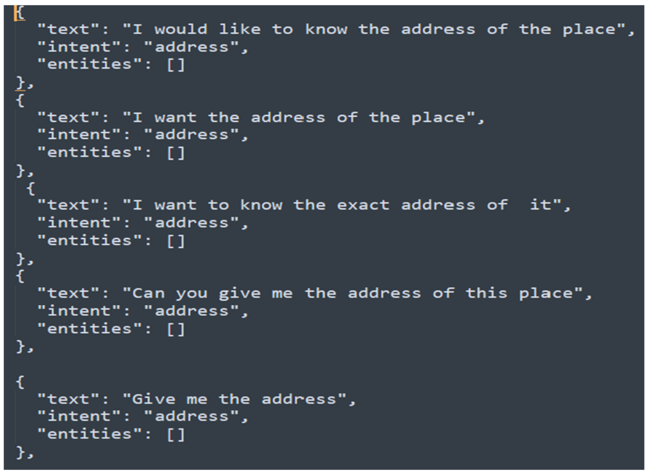
Let's consider the following example :

The user has decided what place he wants to go to and for getting the address of the chosen place, the intent is trained using the following sentences :

1. I would like to know the address of the place
2. I want the address of the place
3. I want to know the exact address of it
4. Can you give me the address of this place
5. Give me the address
6. Give me the address of the place
7. I want the address of this place
8. Tell me the address of this place
9. What is their address

And later the model is tested with the following sentences and the received intent is given :

1. *Tell me their address - address*
2. ***I would like to know their address - price***
3. *I would like to know the address - rating*
4. ***What is the address - address***
5. ***Can you give me the address - address***
6. ***Can you give me their address - address***
7. ***Tell me the address - address***
8. ***Give me their address - address***
9. ***Find me the address - address***
10. ***Find me their address - address***
11. ***I want their address - address***
12. ***I want the address - address***
13. ***What is their address - address***



```
{
  "text": "I would like to know the address of the place",
  "intent": "address",
  "entities": []
},
{
  "text": "I want the address of the place",
  "intent": "address",
  "entities": []
},
 {
  "text": "I want to know the exact address of  it",
  "intent": "address",
  "entities": []
},
{
  "text": "Can you give me the address of this place",
  "intent": "address",
  "entities": []
},

{
  "text": "Give me the address",
  "intent": "address",
  "entities": []
},
```

Fig. 2.

F. Response Generation

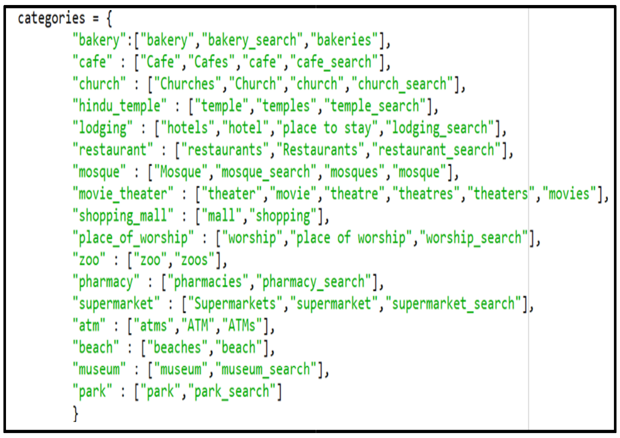
In response generation we have three modules

- Place Search module

This module is called when the user has to search for a specific kind of a place. When the user's question is encountered, using the intent classification step the appropriate intent is identified.

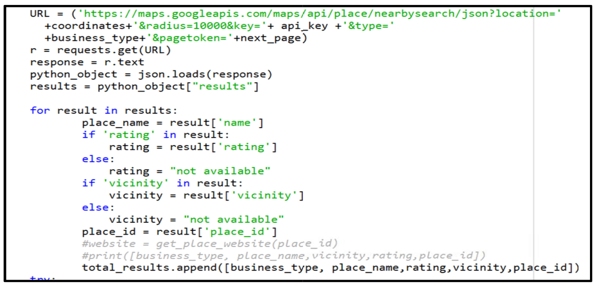
Hence when the intent matches one of these we use the dictionary to map this intent to the appropriate keyword which will match the Google Places API categories. The dictionary is depicted in below figure.

Suppose the intent is found to be temple_search then the input to the API should be hindu_temple, as the API can respond only to those specific categories. Then using the supported keyword, the API is called , and the names along with the area to which they belong is displayed for the first five places found. The following code snippet explains it



```
categories = {
        "bakery":["bakery","bakery_search","bakeries"],
        "cafe" : ["Cafe","Cafes","cafe","cafe_search"],
        "church" : ["Churches","Church","church","church_search"],
        "hindu_temple" : ["temple","temples","temple_search"],
        "lodging" : ["hotels","hotel","place to stay","lodging_search"],
        "restaurant" : ["restaurants","Restaurants","restaurant_search"],
        "mosque" : ["Mosque","mosque_search","mosques","mosque"],
        "movie_theater" : ["theater","movie","theatre","theatres","theaters","movies"],
        "shopping_mall" : ["mall","shopping"],
        "place_of_worship" : ["worship","place of worship","worship_search"],
        "zoo" : ["zoo","zoos"],
        "pharmacy" : ["pharmacies","pharmacy_search"],
        "supermarket" : ["Supermarkets","supermarket","supermarket_search"],
        "atm" : ["atms","ATM","ATMs"],
        "beach" : ["beaches","beach"],
        "museum" : ["museum","museum_search"],
        "park" : ["park","park_search"]
        }
```

Fig. 3.

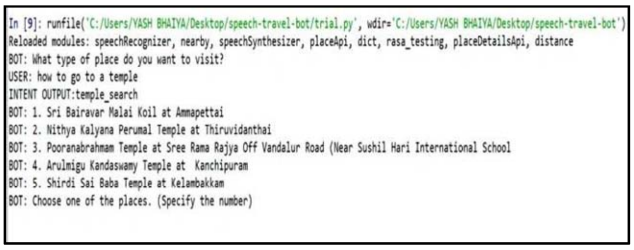


```
URL = ('https://maps.googleapis.com/maps/api/place/nearbysearch/json?location='
  +coordinates+'&radius=10000&key='+ api_key +'&type='
  +business_type+'&pagetoken='+next_page)
r = requests.get(URL)
response = r.text
python_object = json.loads(response)
results = python_object["results"]

for result in results:
        place_name = result['name']
        if 'rating' in result:
            rating = result['rating']
        else:
            rating = "not available"
        if 'vicinity' in result:
            vicinity = result['vicinity']
        else:
            vicinity = "not available"
        place_id = result['place_id']
        #website = get_place_website(place_id)
        #print([business_type, place_name,vicinity,rating,place_id])
        total_results.append([business_type, place_name,rating,vicinity,place_id])
```

.Fig. 4.

After which the user chooses a desired place, and the place id of that place is forwarded to the next module for further processing. The below image shows the intent identification and place search results.



```
In [9]: runfile('C:/Users/YASH BHAIYA/Desktop/speech-travel-bot/trial.py', wdir='C:/Users/YASH BHAIYA/Desktop/speech-travel-bot')
Reloaded modules: speechRecognizer, nearby, speechSynthesizer, placeApi, dict, rasa_testing, placeDetailsApi, distance
BOT: What type of place do you want to visit?
USER: how to go to a temple
INTENT OUTPUT:temple_search
BOT: 1. Sri Bairavar Malai Koil at Ammapettai
BOT: 2. Nithya Kalyana Perumal Temple at Thiruvidanthai
BOT: 3. Pooranabrahmam Temple at Sree Rama Rajya Off Vandalur Road (Near Sushil Hari International School
BOT: 4. Arulmigu Kandaswamy Temple at  Kanchipuram
BOT: 5. Shirdi Sai Baba Temple at Kelambakkam
BOT: Choose one of the places. (Specify the number)
```

Fig. 5.

- Place Details module

This module takes the place id for the user's choice of place as input and this id is used to retrieve further details about that place. Using this place-id the Place Details API is called to give detailed information about the place. Details supported by our bot :

- Summary
- Address
- Phone number
- Price level
- Rating
- Distance
- Opening hours
- Website

The user is asked for the kind of detail he requires and then appropriate responses are generated by using the data in the json output received from the API,and the answer is synthesized back to the user. The above interaction continues until the user does not want any more details.

The details which the user requires are first preprocessed to respond in a manner which the user would easily understand. Like distance from the user's current position is not given in the API, hence using the latitude and longitude of the place chosen and the current geolocation the distance is calculated.

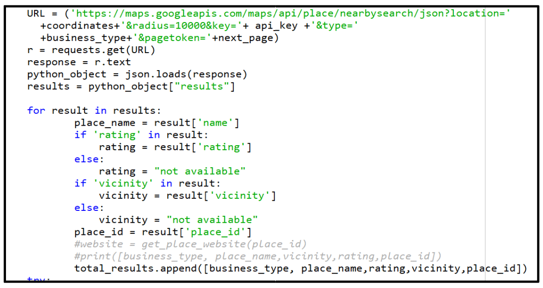
- *Place Information Summary Module*

This module is used to generate a short summary about the place. It uses the Wikipedia library for the above functionality. The name of the place is given to the summary() function as input and then it gives us a 10-20 line short summary about the place as available in wikipedia. The function description is :

The function which is used to return a plain text summary of the page is :

wikipedia.summary(query,sentences=0,chars=0,auto_suggest=True,redirect=True)

Keyword arguments:

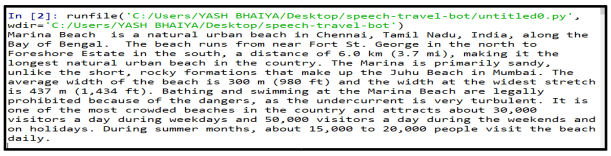


```
URL = ('https://maps.googleapis.com/maps/api/place/nearbysearch/json?location='
  +coordinates+'&radius=10000&key='+ api_key +'&type='
  +business_type+'&pagetoken='+next_page)
r = requests.get(URL)
response = r.text
python_object = json.loads(response)
results = python_object["results"]

for result in results:
        place_name = result['name']
        if 'rating' in result:
            rating = result['rating']
        else:
            rating = "not available"
        if 'vicinity' in result:
            vicinity = result['vicinity']
        else:
            vicinity = "not available"
        place_id = result['place_id']
        #website = get_place_website(place_id)
        #print([business_type, place_name,vicinity,rating,place_id])
        total_results.append([business_type, place_name,rating,vicinity,place_id])
```

- **sentences** - if set, return the first sentences can be no greater than 10). **query -** is the name of the place for which summary has to be obtained
- **chars** - if set, return only the first chars characters (actual text returned may be slightly longer).
- **auto_suggest** - let Wikipedia find a valid page title for the query
- **redirect** - allow redirection without raising RedirectError.
- Figure below is the output received when the below function is run.

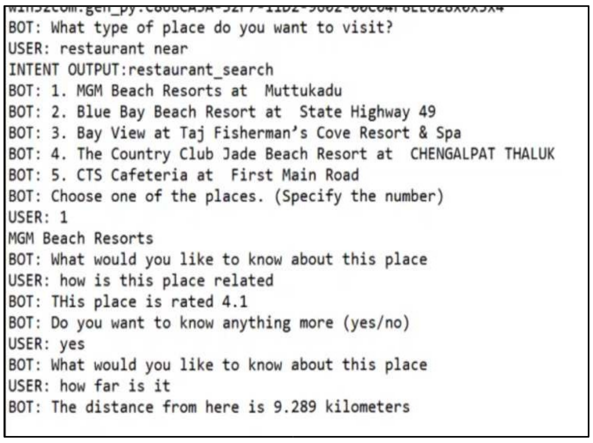
wikipedia.summary("MarinaBeach",sentences=1, auto_suggest=False)



```
In [2]: runfile('C:/Users/YASH BHAIYA/Desktop/speech-travel-bot/untitled0.py',
wdir='C:/Users/YASH BHAIYA/Desktop/speech-travel-bot')
Marina Beach  is a natural urban beach in Chennai, Tamil Nadu, India, along the
Bay of Bengal.  The beach runs from near Fort St. George in the north to
Foreshore Estate in the south, a distance of 6.0 km (3.7 mi), making it the
longest natural urban beach in the country. The Marina is primarily sandy,
unlike the short, rocky formations that make up the Juhu Beach in Mumbai. The
average width of the beach is 300 m (980 ft) and the width at the widest stretch
is 437 m (1,434 ft). Bathing and swimming at the Marina Beach are legally
prohibited because of the dangers, as the undercurrent is very turbulent. It is
one of the most crowded beaches in the country and attracts about 30,000
visitors a day during weekdays and 50,000 visitors a day during the weekends and
on holidays. During summer months, about 15,000 to 20,000 people visit the beach
daily.
```

Fig.6 .

The below image depicts the working of the bot.



```
BOT: What type of place do you want to visit?
USER: restaurant near
INTENT OUTPUT:restaurant_search
BOT: 1. MGM Beach Resorts at  Muttukadu
BOT: 2. Blue Bay Beach Resort at  State Highway 49
BOT: 3. Bay View at Taj Fisherman's Cove Resort & Spa
BOT: 4. The Country Club Jade Beach Resort at  CHENGALPAT THALUK
BOT: 5. CTS Cafeteria at  First Main Road
BOT: Choose one of the places. (Specify the number)
USER: 1
MGM Beach Resorts
BOT: What would you like to know about this place
USER: how is this place related
BOT: THis place is rated 4.1
BOT: Do you want to know anything more (yes/no)
USER: yes
BOT: What would you like to know about this place
USER: how far is it
BOT: The distance from here is 9.289 kilometers
```

Fig. 7 .

V.EVALUATION RESULTS

Speech Recognition Module

For speech recognition, we tested the module with different kinds of sentences the user can ask related to a category, and also the different ways in which the user can ask the same query by varying the length, pronunciation, rate of speech and tone. Below table depicts the results obtained when the module was tested over different sentences.

NO. OF SENTENCES TESTED	165
CORRECTLY RECOGNIZED SENTENCES	152
INCORRECTLY RECOGNIZED SENTENCES	13
ACCURACY PERCENTAGE	**92.1**

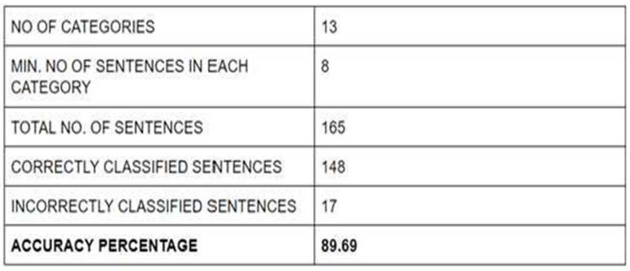
Tab. 1 .

Speech Synthesis Testing Results

For speech synthesis, we tested the module for synthesis of different forms of text, ranging from single sentences to long paragraphs. This module showed high accuracy for speech synthesis of different forms of text, varying in length. The only limitation faced was that the pronunciation of certain parts of text was inaccurate and incorrect, owing to the voice model used that has been trained for a different form of text. Hence, the accuracy percentage for the speech synthesis module can be estimated to be **98%**.

Intent Classification Testing Results

For intent classification we train the dataset with different kinds of sentences the user can ask related to a category, and also the different ways in which the user can ask the same query. Table 6.2 depicts the results obtained when the model was tested over a different set of questions from the trained sample



NO OF CATEGORIES	13
MIN. NO OF SENTENCES IN EACH CATEGORY	8
TOTAL NO. OF SENTENCES	165
CORRECTLY CLASSIFIED SENTENCES	148
INCORRECTLY CLASSIFIED SENTENCES	17
ACCURACY PERCENTAGE	**89.69**

Tab. 2.

Summarization testing results

For testing the performance of the summarization module, we generated summaries for 20 different topics. The quality of each summary was rated by 3 different users within a sale of 1-5 with 1 for poor and 5 for excellent. A sample for the same has been illustrated in the table below. The mean opinion score for each summary was estimated and used to calculate the overall average mean opinion score. Table 6.3 in the following page gives a sample to calculate the mean opinion score for summarized results.

SUMMARY	USER 1	USER 2	USER 3	MEAN OPINION SCORE
TOPIC 1	3	4	5	4.0
TOPIC 2	4	3	4	3.6
TOPIC 3	5	5	4	4.6

Tab. 3.

Similarly, the mean opinion score was calculated for summaries generated for all 20 topics. On average, the mean opinion score for summaries generated by this module for different topics was found to be 4.0 out of 5.

VI. CONCLUSION

The developed bot has proved to be efficient for searching for different kinds of places such as temple, restaurants, malls, theatres, church, mosque and many more within the radius of 50 kms. The bot interacts well with the user and helps to find further details about a specific place as per the user's queries. The intents user's queries. The intents for most of the queries are correctly recognized, which helps in efficient dialogue management. Most importantly, the speech interaction works in a smooth flow without much hindrances thus making the user feel comfortable.

VII. FUTURE WORK

With better implementation of deep learning techniques, the user can be provided with a more personalized experience. To provide enriched results a greater number of categories can be added and also a shorter and crisper summary of a place can be generated using web scraping. Use of web scraping algorithms will provide more enhanced information about the place by surfing through multiple websites. The bot can also be enabled to understand a more advanced queries which involve two nouns and provide the required result. For example, if a user asks for: "Find me a temple near a beach" - the bot should understand the intent as not just a temple search but also a beach nearby.

REFERENCES

[1] Mark Gales and Steve Young (2008), "The Application of Hidden Markov Models in Speech Recognition", Foundations and Trends® in Signal Processing: Vol. 1: No. 3, pp 195-304.

[2] Nirav S. Uchat (2006), Hidden Markov Model and Speech Recognition, Mumbai:Indian I stitute ofTechnology.

[3] https://www.wordstream.com/ log/ws/2017/10/04/chatbots

[4] https://www.30secondstofly.com/ai-software/ultimate-travel-bot-list/

[5] https://viewfinder.expedia.com/features/introducing-expedia-bot-facebook-messenger/

[6] https://www.hipmunk.com/tailwind/lets-talk-travel-try-hello-hipmunk-to-chat-before-you-pack/

[7] https://chatbotslife.com/the-ch tbot-review-1-skyscanner-52f8af8e7d1c

[8] https://medwhat.com/about-us/index.html

[9] https://woebot.io/the-science

[10] https://realpython.com/python-speech-recognition/

[11] https://blog.rasa.com/rasa-nlu-in-depth-part-1-intent-classification/

[12] https://en.wikipedia.org/wiki/Hyperparameter_optimization

[13] https://pyttsx3.readthedocs.io/en/latest/engine.html

[14] https://www.json.org/

[15] https://www.seguetech.com/ajax-technology/

[16] https://developers.google.com/places/web-service/intro

[17] https://wikipedia.readthedocs.io/en/latest/

[18] https://www.expertsystem.com/natural-language-processing-applications/

[19] Wikipedia contributors (2019, March 18). Natural language processing. In Wikipedia, The Free Encyclopedia. Retrieved 19:36, March 25, 2019, https://en.wikipedia.org/w/index.php?title=Natural_language_processing&oldid=888277178

[20] https://www.explainthatstuff.com/how-speech-synthesis-works.html

[21] https://pypi.org/project/PyAudio/0.2.7/

[22] https://pypi.org/project/SpeechRecognition/

[23] https://pyttsx3.readthedocs.io/en/latest/engine.html#pyttsx3.voice.Voice

[24] https://rasa.com/docs/nlu/0.11.4/pipeline/

[25] O. Eray, S. Tokat and S. Iplikci, (2018) "An application of speech recognition with support vector machines," 2018 6th International Symposium on Digital Forensic and Security (ISDFS), Antalya, pp. 1-6.

[26] H. Zen and K. Tokuda, (July 2009) "TechWare: HMM-based speech synthesis resources [Best of the Web]," in IEEE Signal Processing Magazine, vol. 26, no. 4, pp. 95-97.

[27] A. Ganapathiraju, J. E. Hamaker and J. Picone,(Aug. 2004) "Applications of support vector machines to speech recognition," in IEEE Transactions on Signal Processing, vol. 52, no. 8, pp. 2348-2355,.

[28] T. Luong, M. Cao, D. Le and X. Phan,(2017) "Intent extraction from social media texts using sequential segmentation and deep learning models," 2017 9th International Conference on Knowledge and Systems Engineering (KSE), Hue, pp. 215-220.

[29] Z. Wang, Y. Qi and J. L. Z. Ma (2016) "User intention understanding from scratch," 2016 First International Workshop on Sensing, Processing and Learning for Intelligent Machines (SPLINE), Aalborg, pp. 1-4.

[30] A. Noorithaya, M. K. Kumar and A. Sreedevi (2014) "Voice assisted navigation system for the blind," International Conference on Circuits, Communication, Control and Computing, Bangalore, pp. 177-181.

[31] A. Argal, S. Gupta, A. Modi, P. Pandey, S. Shim and C. Choo (2018) "Intelligent travel chatbot for predictive recommendation in echo platform," 2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC), Las Vegas, NV, pp. 176-183.