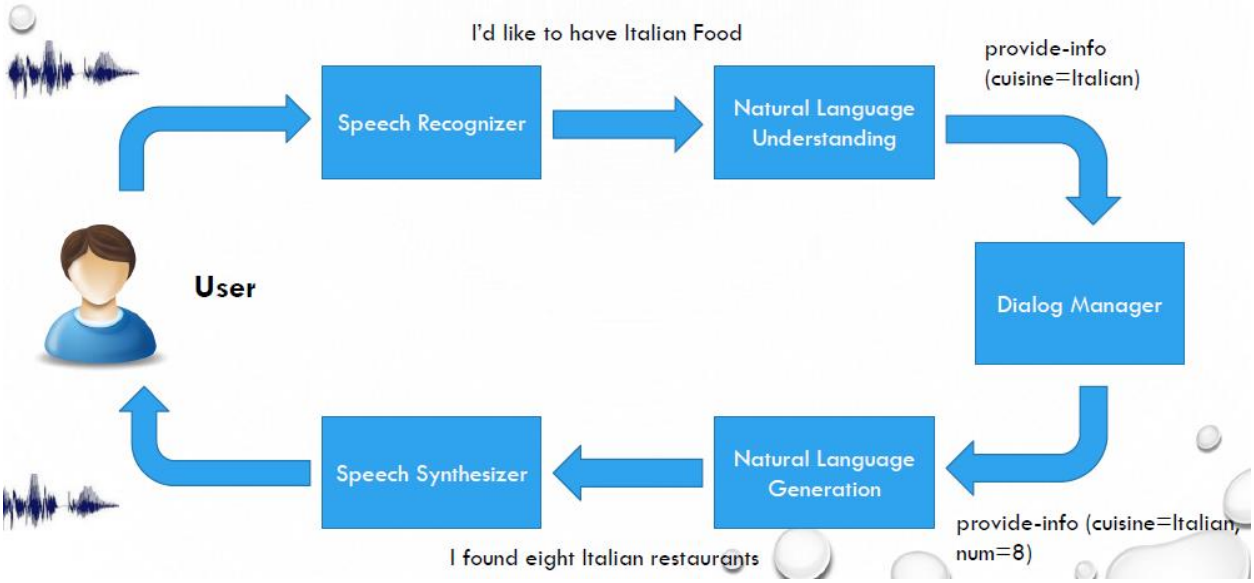
Individual Report on Project (GROUP NUMBER): Restaurant Genie (Group 19)
Name: Arpit Nigam
Email address: arpitnig@usc.edu
Rest of my group: Harish Narayanan, Jasmine Deuri, Arjun Sureshbabu
GitHub repository: https://github.com/nigamarpit/NLP-Project/tree/master/main

**1. Project Overview**

Restaurant Genie is a dialogue system that can be used by users for finding restaurant based upon their preferences. Users can interact with system using speech as well as text. Similarly, system also gives results in voice as well as text form. At present, this system has restricted domain (Cuisine, Location, Price, Rating) from which it can understand user queries.




This system follows the complete 5 module spoken dialogue system architecture.

First module being Speech Recognizer which is responsible for understanding the input in voice form and later converting it to the text form so that it can be used by Natural Language Understanding module for further processing.

Natural Language Understanding is second module which is responsible for understanding the user query and also for extracting the keywords of importance by entity detection on the tokenized sentences. Segmentation, tokenization, entity detection and relation detection are submodules inside the Natural Language Understanding module and are implemented using python nltk library.

Keyword which are extracted by Natural Language Understanding module are input to third module Dialogue Manager. Dialogue Manager's main responsibility is to get the data from Factual API based upon these keywords and if suitable query can't be made out of the keywords then ask the user for required information to form meaningful API call. Result set returned by the API call is then sent to the Natural Language Generation module.

Natural Language Generation is fourth module and it processes and filters out relevant records from the result set based upon user preference. These filtered record are then used to form meaningful sentences and then passed on to last module which is Speech Synthesizer.

Speech Synthesizer is responsible for generating all output in form of voice using win32com python libraries.

Factual API is working in background which is considered to have the most up to date information about the restaurants in United States. Dialogue Manager is making all the API calls using Factual utilities and the result which is returned is in JSON format. This JSON result set is parsed and further processed/ filter by Natural Language Generator.

Evaluations are formed at 2 modules i.e., Speech Recognizer and Natural Language Understanding; and at overall system level. Word Error Rate is used to compute the accuracy at Speech Recognizer Level. Accuracy for Speech Recognizer at initial testing came out close to 97%. While precision, recall and F1 scores are used to calculate accuracy at Natural Language Understanding and at overall level. Initial testing F1 scores were close to 95% and 94% for Natural Language Understanding and overall system respectively.

**2. My Primary Responsibility**

I worked on first - Speech Recognition and last - Speech Generation module of the overall system. Restaurant Genie dialogue system always gives preference to voice input over text input. If speech recognizer is not able to identify any speech, then only it will ask for the text input from the user. Since Dialogue Manager is the startup program for this system, every time it starts or needs any missing information it essentially invokes the Speech Generation followed by invocation of the Speech Recognizer module.

Speech Recognition module is built using CMU PocketSphinx and python speech_recognition libraries. Speech recognition module works in 2 step process:

a. Listen Audio: Every time user asks any query speech_recognition module listen to the voice and once query is finished it saves it into the .raw file format locally.

```
import os
import speech_recognition as sr
from pocketsphinx import AudioFile, get_model_path, get_data_path

model_path = os.getcwd()
data_path = os.getcwd()

def microphoneListen():
    r = sr.Recognizer()
    print('Listening...')
    with sr.Microphone() as source:
        audio = r.listen(source)
    print('Processing speech input...')
    with open('microphone-results.raw', 'wb') as f:
        f.write(audio.get_raw_data())
```

b. Decode Audio: Above step is followed by decoding of .raw file and it uses the PocketSphinx library, default acoustic model and our custom domain specific language model.

```
    audio = AudioFile(**config)
    for phrase in audio:
        print(phrase)
        if len(str(phrase))==0:
            phrase=input()
        return str(phrase)
```

CMU PocketSphinx package comes with default acoustic and language model. Achieving good accuracies with default language model is very hard and sometimes tuning of acoustic model also helps to achieve good accuracies. Since this system is using English as only language for interaction we are able to achieve good accuracies without tweaking the acoustic model but with default language model, PocketSphinx was not able to recognize user input properly so we have to come up with our own language model.

Custom language model is built using 300 sentences from various possible voice inputs in which users can ask queries with system. Once I had these dataset of possible questions I created language model using CMU lmtools. CMU lmtools generates 5 different files out of which 2 are of our use .dic and .lm file.

```
config = {
    'audio_file': os.path.join(data_path, 'microphone-results.raw'),
    'hmm': os.path.join(model_path, 'model\\en-us'),
    'lm': os.path.join(model_path, 'model\\model.bin'),
    'dict': os.path.join(model_path, 'model\\7886.dic')
}
```

Since PocketSphinx requires binary file in configuration setting I have to convert .lm file to .bin binary using sphinx_lm_convert.exe which comes with cmuSphinx package. Apart from .bin file PocketSphinx also requires .dic file which I generated through lmtools mentioned above. Last required setting we need for PocketSphinx which is hmm configuration setting which is nothing but the acoustic model and I used the default acoustic model which comes with the CMU pocketSphinx i.e., en-us.

Last module in our project was Speech Generation and it is getting invoked by the Natural Language Generation module which in turn is dependent on Dailogue Manager for invocation. For Speech Generation we have used python win32com library and its inbuilt methods like Speak. Whenever Speech Generation module is getting invoked it is essentially converting text to speech and this text is coming from Natural Language Generation module.

```
import win32com.client as wincl

speak = wincl.Dispatch("SAPI.SpVoice")
def SpeechOutput(str):
    print(str)
    speak.Speak(str)
```

Finally, for Speech Recognizer I conducted multiple surveys with multiple users to identify the performance of the system. For calculation of

performance I used the Word-Error Rate matrix to identify the accuracy of the system.

$$\text{Word Error Rate (WER)} = 100\left(\frac{S+D+I}{N}\right)\%$$

where, S = number of substitutions
D = number of deletions
I = number of insertions
N = number of words in the reference transcript

$$\text{Accuracy} = 100 - \text{WER}\%$$

From surveys I am found that WER is around 4% which gave me and overall accuracy for Speech Recognition close to 96%.

Below is one sample conversation with the system and all the text which is in capital letter is decode from speech input by our Speech Recognizer module. Except (Listening...), (Processing speech input...) and (all capital words) which are in below screenshot rest every statement is spoken by our Speech Synthesizer module.

```
C:\Users\Xenon\Documents\GitHub\NLP-Project\main>python DailogueManager.py
Hello, how may I help you today?
Listening...
Processing speech input...
I AM AM INTERESTED IN CHINESE FOOD
You asked for CHINESE cuisine. Is that correct?
Listening...
Processing speech input...
YES
Do you have any location preference?
Listening...
Processing speech input...
NO
Please enter your preference for price from Cheap, Moderate, Expensive
Listening...
Processing speech input...
CHEAP
You entered cheap. Is that correct?
Listening...
Processing speech input...
YES
Please enter your preference for rating [1 - 5]
Listening...
Processing speech input...
FIVE
You entered 5. Is that correct?
Listening...
Processing speech input...
YES
I found 2 restaurants that serve CHINESE food near you.
All are highly rated
and all of them are cheap with price range of $15 per person.
The most preferred restaurants based on rating are:
El Ranchito Restaurant
1.El Ranchito Restaurant        Website: http://www.elranchitorestaurant.com/   Tel: (213) 747-0443
Thai Trio
2.Thai Trio     Website: http://triohouse.com   Tel: (213) 742-6764
```

**3. Other Project Work**

Integration of Dialogue Manager with Speech Recognizer and Speech Synthesizer. Since system can interact with user and vice versa only using these two modules i.e., Speech Recognizer and Speech Generation, I worked on integration of Speech Recognizer with Natural Language Understanding module

for understanding speech input; and Speech Synthesizer with Natural Language Generation module for generating speech output for user.
Presentation content for Speech Recognizer and Speech Synthesis module of the system.

**4. Online Resources**

a) CMU Sphinx
http://cmusphinx.sourceforge.net/

b) CMU PocketSphinx
https://github.com/cmusphinx/pocketsphinx

c) CMU Language Modelling Toolkit
http://www.speech.cs.cmu.edu/tools/lmtool-new.html

d) Python Speech Recognizer
https://pypi.python.org/pypi/SpeechRecognition/2.1.3
https://github.com/Uberi/speech_recognition

e) MIT pyAudio
https://people.csail.mit.edu/hubert/pyaudio/
https://pypi.python.org/pypi/PyAudio

f) Python pywin32
https://pypi.python.org/pypi/pywin32

**5. References**

Rabiner L.R., Juang B.H.(1986), An introduction to Hidden Markov Models. IEEE ASSP Magazine, 3(1) 4-16 doi: 10.1109/MASSP.1986.1165342

F.Chen Stanley, Goodman Joshua (1996), An empirical study of smoothing techniques for language modelling, ACL '96 Proceedings of the 34th annual meeting on Association for Computational Linguistics, 310-318 doi: 10.3115/981863.981904