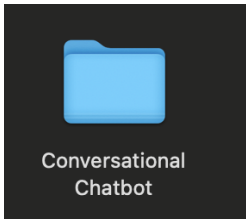
Shivansh Shukla
EE 104
SJSU

Conversational Chatbot using NLP (Natural Language Processing)

For this assignment, you will learn how to build a Conversational Chatbot using a NLP model and Python. We will be referring to the Analytics Vidhya website for this lab: https://www.analyticsvidhya.com/blog/2021/10/complete-guide-to-build-your-ai-chatbot-with-nlp-in-python/

Initial Setup

1) Create a new folder on your personal computer named "Conversational Chatbot using NLP" on your Desktop or any other location on your personal computer.



a) Go to your terminal/PowerShell and ensure Python is installed by typing "python --version." It is advised to use Python 3.9 for this lab.

```
(ExtraCredit) (base) shivanshshukla@Shivanshs-MacBook-Pro Conversational Chatbot
 using NLP % python --version
Python 3.9.18
```

b) If Python is not found, download from either https://www.python.org/downloads/release/python-3918/ or https://anaconda.org/anaconda/python/files?sort=ndownloads&version=3.9.18&drop_args=channel&sort_order=desc&type=&page=1

```
shivanshshukla — -zsh — 80×24
Last login: Fri Dec  8 15:48:27 on ttys000
(base) shivanshshukla@Shivanshs-MacBook-Pro ~ % conda deactivate
shivanshshukla@Shivanshs-MacBook-Pro ~ % python --version
zsh: command not found: python
shivanshshukla@Shivanshs-MacBook-Pro ~ %
```

2) Make sure to cd (change directory) to the project directory in the terminal

```
shivanshshukla@Shivanshs-MacBook-Pro ~ % cd Desktop/SJSU/Fall\ \'23\ \(last\)/EE
104/Extra\ Credit/Conversational\ Chatbot\ using\ NLP
shivanshshukla@Shivanshs-MacBook-Pro Conversational Chatbot using NLP %
```

a) Create a virtual environment in your chatbot folder by typing “python -m venv your_env_name” in the terminal. After creating the venv, activate it by typing “source your_env_name/bin/activate” for macOS. The activation command will be different for Windows so adjust accordingly. I have provided reference images below.

```
(ExtraCredit) (base) shivanshshukla@Shivanshs-MacBook-Pro Conversational Chatbot
 using NLP % python -m venv venv_bot
```

```
(ExtraCredit) (base) shivanshshukla@Shivanshs-MacBook-Pro Conversational Chatbot
 using NLP % source your_env_name/bin/activate
```

b) In my case, the venv name is “ExtraCredit” and is already activated, as seen in the figure below.

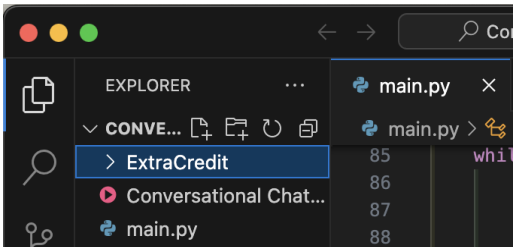
```
(ExtraCredit) shivanshshukla@Shivanshs-MacBook-Pro Conversational Chatbot using NLP %
```

3) Once the environment is activated, pip install the following libraries below in the venv.

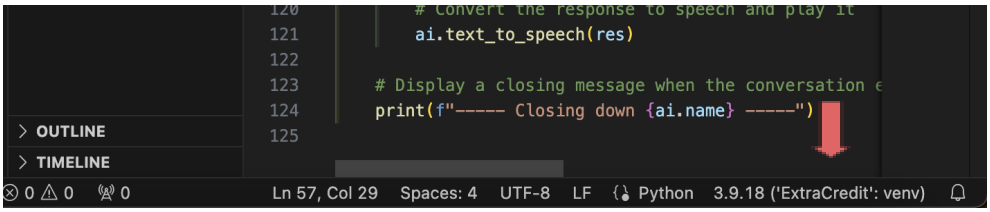
```
pip install SpeechRecognition
pip install gTTS
pip install tensorflow
pip install transformers==4.30
```

Importing Libraries

1. Open Visual Studio Code (VS code) or any other preferred IDE like Spyder. Open your Chatbot folder in the IDE which will show your venv folder. Make a new file named main.py which we will use to type the code in.



a. Ensure that the VS code is running the correct Python interpreter which should be the venv you created.



2. Import the following libraries in the main.py file

main.py

```python
#Source: Arnab Mondal (Analytics Vidhya)
#Modified: Shivansh Shukla (SJSU, Graduated Fall '23)


# Import necessary libraries for speech-to-text, text-to-speech, language model, and other functionalities
import speech_recognition as sr
from gtts import gTTS
import transformers
import os
import time
import datetime
import numpy as np
```

Initialization of the Chatbot class

This section encapsulates the functionality of a speech-based chatbot. It has methods for converting speech to text using Google's speech recognition, converting text to speech using the gTTS library, and checking for the wake-up phrase. Additionally, it includes a static method to retrieve the current time. The class initializes with a startup message displaying the chatbot's name, and it is designed to be instantiated with a specific name parameter.

1. Create a chatbot class.

```python
# Define a class for the ChatBot
class ChatBot():
    def __init__(self, name):
        # Display a message indicating the ChatBot is starting up
        print("----- Starting up", name, "-----")
        self.name = name
```

2. Inside the chatbot class, write a function that will handle speech-to-text conversion

```python
    # Method for converting speech to text using Google's speech recognition
    def speech_to_text(self):
        # Create a speech recognizer instance
        recognizer = sr.Recognizer()
        # Use the microphone to capture audio
        with sr.Microphone() as mic:
            # Display a message indicating that the ChatBot is listening
            print("Listening...")
            # Listen to the microphone input
            audio = recognizer.listen(mic)
            # Set a default value for the text (in case of an error)
            self.text="ERROR"
        try:
            # Attempt to recognize speech using Google's speech recognition
            self.text = recognizer.recognize_google(audio)
            # Display the recognized text
            print("Me  --> ", self.text)
        except:
            # Display an error message if speech recognition fails
            print("Me  -->  ERROR")
```

3. Write another static method function inside the chatbot class to handle text-to-speech conversion.

```
    # Static method for converting text to speech using gTTS library
    @staticmethod
    def text_to_speech(text):
        # Display a message indicating that the AI is responding
        print("AI --> ", text)
        # Create a gTTS instance with the specified text and language
        speaker = gTTS(text=text, lang="en", slow=False)
        # Save the speech as an MP3 file
        speaker.save("res.mp3")
        # Get information about the MP3 file
        statbuf = os.stat("res.mp3")
        # Calculate the duration of the speech based on file size
        mbytes = statbuf.st_size / 1024
        duration = mbytes / 200
        # Play the MP3 file using the appropriate system command (Mac or Windows)
        os.system('afplay res.mp3')  # If you are using Mac -> afplay, for Windows -> start
        # Pause for a specific duration to allow speech playback
        time.sleep(int(50*duration))
        # Remove the temporary MP3 file
        os.remove("res.mp3")
```

4. Write one more function that will give the chatbot the ability to tell the correct time.

```
    # Static method to get the current time
    @staticmethod
    def action_time():
        # Return the current time in the format HH:MM
        return datetime.datetime.now().time().strftime('%H:%M')
```

Execution of AI chatbot

This code orchestrates the chatbot's main execution: instantiating "Jarvis," loading a conversational model, and managing an ongoing conversation loop that responds to user inputs, including wake-up phrases, time inquiries, politeness, and exit commands. The loop utilizes a language model for varied responses, concluding with a closing message.

1. Initialize the chatbot by naming it anything you like. I have it named as “Jarvis.” Next, load a conversational model from the transformers library. Set up an environment variable for parallel tokenization and define a loop control variable to manage the main conversation loop.

```
# Main part of the code where the AI is instantiated and runs
if __name__ == "__main__":
    # Create an instance of the ChatBot class with the name "Jarvis"
    ai = ChatBot(name="Jarvis")

    # Load a conversational language model using Hugging Face's Transformers library
    nlp = transformers.pipeline("conversational", model="microsoft/DialoGPT-medium")

    # Set an environment variable for parallel tokenization
    os.environ["TOKENIZERS_PARALLELISM"] = "true"

    # Variable to control the loop
    ex=True
```

2. This code segment constitutes the primary conversation loop (while ex), capturing user speech input with ai.speech_to_text(). If the wake-up phrase "Jarvis" is detected, it responds with a predefined greeting: "Hello, I am Jarvis, what can I do for you?"

```
    # Main loop for the conversation with the AI
    while ex:
        # Get input from the user through speech
        ai.speech_to_text()

        # Check for wake-up phrase
        if any(i in ai.text for i in ["Jarvis","Jarvis"]): #ai.wake_up(ai.text) is True:
            # Generate a response if the wake-up phrase is detected
            res = "Hello I am Jarvis, what can I do for you?"
```

3. This code segment handles specific user inputs in the conversation loop. If the user requests the current time, it responds with the current time. If the user expresses gratitude, it randomly selects a polite response. If the user mentions "exit" or "close," it provides a random farewell message and sets the loop control variable (ex) to False to exit the loop.

```
    # Check for the request for the current time
    elif "time" in ai.text:
        # Get the current time and set it as the response
        res = ai.action_time()
    # Respond politely to "thank you"
    elif any(i in ai.text for i in ["thank","thanks"]):
        # Choose a random polite response from a list
        res = np.random.choice(["you're welcome!","anytime!","no problem!","cool!","I'm here if you need me!","mention not"])
    # Exit the conversation if the user says "exit" or "close"
    elif any(i in ai.text for i in ["exit","close"]):
        # Choose a random farewell message and set it as the response
        res = np.random.choice(["Tata","Have a good day","Bye","Goodbye","Hope to meet soon","peace out!"])
        # Set the loop control variable to False to exit the loop
        ex=False
```

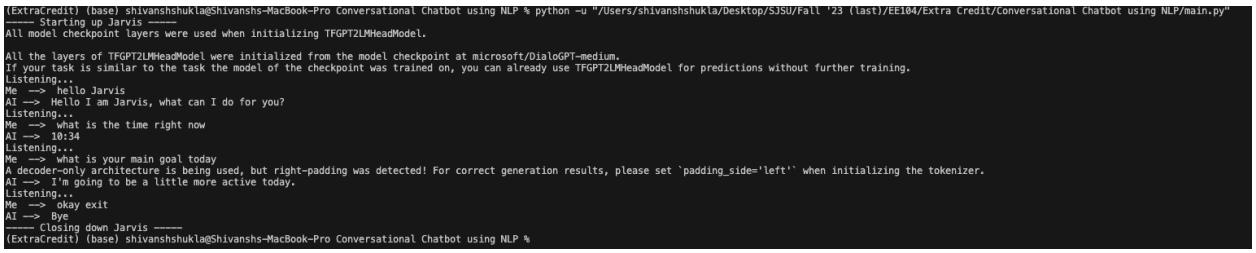
4. This section manages the ongoing conversation by utilizing a language model. If there is an error in speech recognition, it responds with an apology. Otherwise, it employs the language model to generate a response, extracts the bot's reply, converts it to speech, and plays it. Upon concluding the conversation loop, a closing message is displayed, indicating the shutdown of the ChatBot named "Jarvis."

```
        # Continue the conversation using the language model
        else:
            # Handle the case when there is an error in speech recognition
            if ai.text=="ERROR":
                # Set an error message as the response
                res="Sorry, come again?"
            else:
                # Use the language model to generate a response
                chat = nlp(transformers.Conversation(ai.text), pad_token_id=50256)
                # Extract the bot's response from the model's output
                res = str(chat)
                res = res[res.find("bot >> ")+6:].strip()

        # Convert the response to speech and play it
        ai.text_to_speech(res)

    # Display a closing message when the conversation ends
    print(f"----- Closing down {ai.name} -----")
```

5. Now you are done with this assignment. You can run it directly from the IDE or can run it through terminal by typing “python main.py” while making sure that you are in the correct directory. Below is a sample output.



The complete main.py file

```
#Author: Arnab Mondal (Analytics Vidhya)
#Modified: Shivansh Shukla (SJSU)


# Import necessary libraries for speech-to-text, text-to-speech,
language model, and other functionalities
import speech_recognition as sr
from gtts import gTTS
import transformers
import os
import time
import datetime
import numpy as np

# Define a class for the ChatBot
class ChatBot():
    def __init__(self, name):
        # Display a message indicating the ChatBot is starting up
        print("----- Starting up", name, "-----")
        self.name = name

    # Method for converting speech to text using Google's speech
recognition
    def speech_to_text(self):
        # Create a speech recognizer instance
        recognizer = sr.Recognizer()
```

```
        # Use the microphone to capture audio
        with sr.Microphone() as mic:
            # Display a message indicating that the ChatBot is
listening
            print("Listening...")
            # Listen to the microphone input
            audio = recognizer.listen(mic)
            # Set a default value for the text (in case of an error)
            self.text="ERROR"
        try:
            # Attempt to recognize speech using Google's speech
recognition
            self.text = recognizer.recognize_google(audio)
            # Display the recognized text
            print("Me  --> ", self.text)
        except:
            # Display an error message if speech recognition fails
            print("Me  -->  ERROR")

    # Static method for converting text to speech using gTTS library
    @staticmethod
    def text_to_speech(text):
        # Display a message indicating that the AI is responding
        print("AI --> ", text)
        # Create a gTTS instance with the specified text and language
        speaker = gTTS(text=text, lang="en", slow=False)
        # Save the speech as an MP3 file
        speaker.save("res.mp3")
        # Get information about the MP3 file
        statbuf = os.stat("res.mp3")
        # Calculate the duration of the speech based on file size
        mbytes = statbuf.st_size / 1024
        duration = mbytes / 200
        # Play the MP3 file using the appropriate system command (Mac
or Windows)
        os.system('afplay res.mp3')  # If you are using Mac ->
afplay, for Windows -> start
        # Pause for a specific duration to allow speech playback
        time.sleep(int(50*duration))
```

```
        # Remove the temporary MP3 file
        os.remove("res.mp3")

    # Method to check if the wake-up phrase is detected in the input
text
    # def wake_up(self, text):
    #     # Return True if the wake-up phrase is found in the
lowercase version of the text
    #     return True if self.name in text.lower() else False

    # Static method to get the current time
    @staticmethod
    def action_time():
        # Return the current time in the format HH:MM
        return datetime.datetime.now().time().strftime('%H:%M')

# Main part of the code where the AI is instantiated and runs
if __name__ == "__main__":
    # Create an instance of the ChatBot class with the name "Jarvis"
    ai = ChatBot(name="Jarvis")

    # Load a conversational language model using Hugging Face's
Transformers library
    nlp = transformers.pipeline("conversational",
model="microsoft/DialoGPT-medium")

    # Set an environment variable for parallel tokenization
    os.environ["TOKENIZERS_PARALLELISM"] = "true"

    # Variable to control the loop
    ex=True

    # Main loop for the conversation with the AI
    while ex:
        # Get input from the user through speech
        ai.speech_to_text()

        # Check for wake-up phrase
        if any(i in ai.text for i in ["Jarvis","Jarvis"]):
```

```
#ai.wake_up(ai.text) is True:
            # Generate a response if the wake-up phrase is detected
            res = "Hello I am Jarvis, what can I do for you?"
        # Check for the request for the current time
        elif "time" in ai.text:
            # Get the current time and set it as the response
            res = ai.action_time()
        # Respond politely to "thank you"
        elif any(i in ai.text for i in ["thank","thanks"]):
            # Choose a random polite response from a list
            res = np.random.choice(["you're welcome!","anytime!","no
problem!","cool!","I'm here if you need me!","mention not"])
        # Exit the conversation if the user says "exit" or "close"
        elif any(i in ai.text for i in ["exit","close"]):
            # Choose a random farewell message and set it as the
response
            res = np.random.choice(["Tata","Have a good
day","Bye","Goodbye","Hope to meet soon","peace out!"])
            # Set the loop control variable to False to exit the loop
            ex=False
        # Continue the conversation using the language model
        else:
            # Handle the case when there is an error in speech
recognition
            if ai.text=="ERROR":
                # Set an error message as the response
                res="Sorry, come again?"
            else:
                # Use the language model to generate a response
                chat = nlp(transformers.Conversation(ai.text),
pad_token_id=50256)
                # Extract the bot's response from the model's output
                res = str(chat)
                res = res[res.find("bot >> ")+6:].strip()

        # Convert the response to speech and play it
        ai.text_to_speech(res)

    # Display a closing message when the conversation ends
```

```
    print(f"----- Closing down {ai.name} -----")
```

References:

Mondal, A. (2023, October 25). *How to build your AI chatbot with NLP in python?*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/10/complete-guide-to-build-your-ai-chatbot-with-nlp-in-python/