Desktop Chat Bot Application

Personal Details

Name: Aksshay Balasubramaian

University: Chennai Institute of Technology

Email: aksshaybala8846@gmail.com (connect with me on linkedin)

Telephone: 7200347086

Github: Aksshay88

Country of Residence: India

Timezone: IST (GMT + 0530)

Primary Language: English

I am a second-year student pursuing Computer Science Engineering at Chennai Institute of Technology, Chennai. My semester will be completed in late April, leaving me enough time to prepare for my GSoC project. If I am selected, I shall be able to work around 40 hours a week on the project, though I am open to putting in more effort if the work requires.

Why this project?

The main reason for this project is to provide users with a versatile and efficient desktop Chat-Bot application that operates offline and integrates advanced Natural Language Processing (NLP) capabilities. By leveraging NLP techniques and neural language models, the Chat-Bot will be able to comprehend and generate natural language responses, facilitating seamless communication with users. This project aims to address the growing demand for intelligent conversational agents that can assist users in various tasks, such as information retrieval, task automation, and customer support, without the need for an internet connection. Additionally, by

integrating the OpenVINO toolkit, the application will ensure optimized performance and efficient inference of the neural language model, even on resource-constrained devices. Overall, the primary goal of this project is to empower users with a user-friendly and reliable Chat-Bot application that enhances productivity and user experience, while also contributing to the advancement of conversational AI technology.

Technical Knowledge

I am a second-year student at Chennai Institute of Technology. I am enrolled in a 4 year B.Tech course. My major is Computer Science Engineering(it's more of maths and Computers though). My specialization is in the field of Networking, Full Stack Development, and Artificial Intelligence. The courses that I have done include

- Web Development
- React - The Complete Guide 2024 (incl. React Router & Redux)

- Advanced Computing for Computer Engineers(A compressed version of important CS courses)
- Advanced Topics in Networking (CCNA v7)
- Computer Vision

I have done many algorithms and machine learning courses on Coursera.

Some of my previous projects in image processing and computer vision include -

- Anomaly Detection For Fraud detection system for a banking solutions
- Face recognition for Identifying the identity of the dead person

Am currently working on many interesting projects, including Cold-Recog, finding out innovative techniques for finding the

temporal resolution of the identity of Dead Person (Using Aadhar database) etc. Am also currently working on implementing the KNN algorithm. Read ahead in the personal motivation section for the reasons why I have already been working on implementing this technique .

I interned with IITM Pravartak Technologies Foundation in my third year. My work involved extensive use of UNIX and Linux . Pravartak involved working with teams across the globe. So I am comfortable working with people across multiple time zones and this shall not be a problem in the development process. I have worked with Shell Scripting and Networking in my advanced computing course. Based on my skills, I was selected to be Best Computer Science Badge in Linkedin (Over Top 3% in the world) for Chennai Institute of Technology, in which programming was taught in Workshop for Networking with Linux .

Programming languages I have previously worked with include C, Java, Python, Matlab, SQL, Bash, Docker, React Native,lua etc. I have built certain customization for my Linux Garuda (Arch based Distro).

Project

Project Abstract

The **Desktop Chat-Bot Application** project aims to develop a versatile and efficient cross-platform application that enables seamless communication between users and a chat-bot interface. The application will integrate advanced Natural Language Processing (NLP) techniques, allowing the chat-bot to comprehend natural language inputs and generate appropriate responses. The key features of the project include:

NLP Integration: The application will leverage state-of-the-art NLP models to process user queries and generate contextually relevant

responses. This integration will enable the chat-bot to understand user intent, extract key information, and provide accurate and helpful replies.

Offline Functionality: The chat-bot application will operate offline, eliminating the need for an internet connection. By storing necessary data and resources locally, users will be able to interact with the chat-bot seamlessly, regardless of internet availability.

Cross-Platform Compatibility: The application will be developed using cross-platform technologies, ensuring compatibility with various desktop operating systems such as Windows, macOS, and Linux. Users will have the flexibility to access the chat-bot interface from their preferred desktop environment.

User Interface Design: The application will feature an intuitive and user-friendly interface designed using modern web technologies. The interface will facilitate smooth interactions between users and the chat-bot, enhancing the overall user experience.

OpenVINO Integration: To optimize performance, the project will integrate the OpenVINO toolkit for efficient inference of the NLP models. This integration will ensure fast and reliable processing of user queries, even on resource-constrained devices.

Technical Details

Below we have explained the significant steps of the Project .

- We start by considering each of the possible ways of the approach independently for analysis

- We choose a suitable colour space in which we want to work, This could depend on the specific application that we are dealing with, though in the paper, authors have used NTSC color space for further operations.

```
1
2 import numpy as np
3
4
5 # Build the AI
6 class ChatBot():
7     def __init__(self, name):
8         print("----- starting up", name, "-----")
9         self.name = name
10 # Execute the AI
11 if __name__ == "__main__":
12     ai = ChatBot(name="Dev")
```

```
----- starting up Dev -----
```

```
import speech_recognition as sr
def speech_to_text(self):
    recognizer = sr.Recognizer()
    with sr.Microphone() as mic:
        print("listening...")
        audio = recognizer.listen(mic)
    try:
        self.text = recognizer.recognize_google(audio)
        print("me --> ", self.text)
    except:
        print("me -->  ERROR")
```

Activation Trigger: The function is designed to activate the chatbot upon receiving specific trigger phrases, such as "Hey Dev" or "Hello Dev". This activation trigger is essential for initiating the interaction between the user and the chatbot.

Speech Recognition: The function incorporates speech recognition capabilities to identify the trigger phrases spoken by the user. It listens for audio signals and processes them to detect the activation command.

Conditional Activation: Upon detecting the activation trigger, the function activates the chatbot, allowing it to receive and process subsequent user inputs. This conditional activation ensures that the chatbot remains dormant until prompted by the user.

Response Generation: Once activated, the chatbot processes the user's speech input and generates suitable responses based on predefined algorithms or machine learning models. These

responses may vary depending on the context of the conversation and the specific queries posed by the user.

Natural Language Understanding: The function employs natural language understanding techniques to interpret the user's speech input accurately. It analyzes the semantics and intent behind the user's words to generate relevant and coherent responses.

Output Delivery: After processing the user's input and generating suitable responses, the function delivers the output or response to the user through the appropriate channel, such as text-to-speech synthesis or displaying text on a graphical user interface.

Continuous Interaction: The function facilitates continuous interaction between the user and the chatbot by listening for additional input after delivering a response. This allows for a dynamic and conversational interaction experience, where the

chatbot can respond to follow-up questions or requests from the user.

```
#Those two functions can be used like this
# Execute the AI
if __name__ == "__main__":
    ai = ChatBot(name="Dev")
    while True:
        ai.speech_to_text()
        ## wake up
        if ai.wake_up(ai.text) is True:
            res = "Hello I am Dev the AI, what can I do for you?"
        ai.text_to_speech(res)
```

This can be used as an initializing screen

This is where the AI chatbot becomes intelligent and not just a scripted bot that will be ready to handle any test thrown at it. The main package we will be using in our code here is the Transformers package provided by HuggingFace, a widely acclaimed resource in AI chatbots. This tool is popular amongst developers, including those working on AI chatbot projects, as it allows for pre-trained models and tools ready to work with various NLP tasks. In the code below, we have specifically used the DialogGPT AI chatbot, trained and created by Microsoft based on millions of conversations and ongoing chats on the Reddit platform in a given time.

Code:

```
import transformers
nlp = transformers.pipeline("conversational",
                            model="microsoft/DialoGPT-medium")
#Time to try it out
input_text = "hello!"
nlp(transformers.Conversation(input_text), pad_token_id=50256)
```

Reminder: Don't forget to provide the pad_token_id as the current version of the library we are using in our code raises a warning when this is not specified. What you can do to avoid this warning is to add this as a parameter.

nlp(transformers.Conversation(input_text), pad_token_id=50256)

You will get a whole conversation as the pipeline output and hence you need to extract only the response of the chatbot here.

Code:

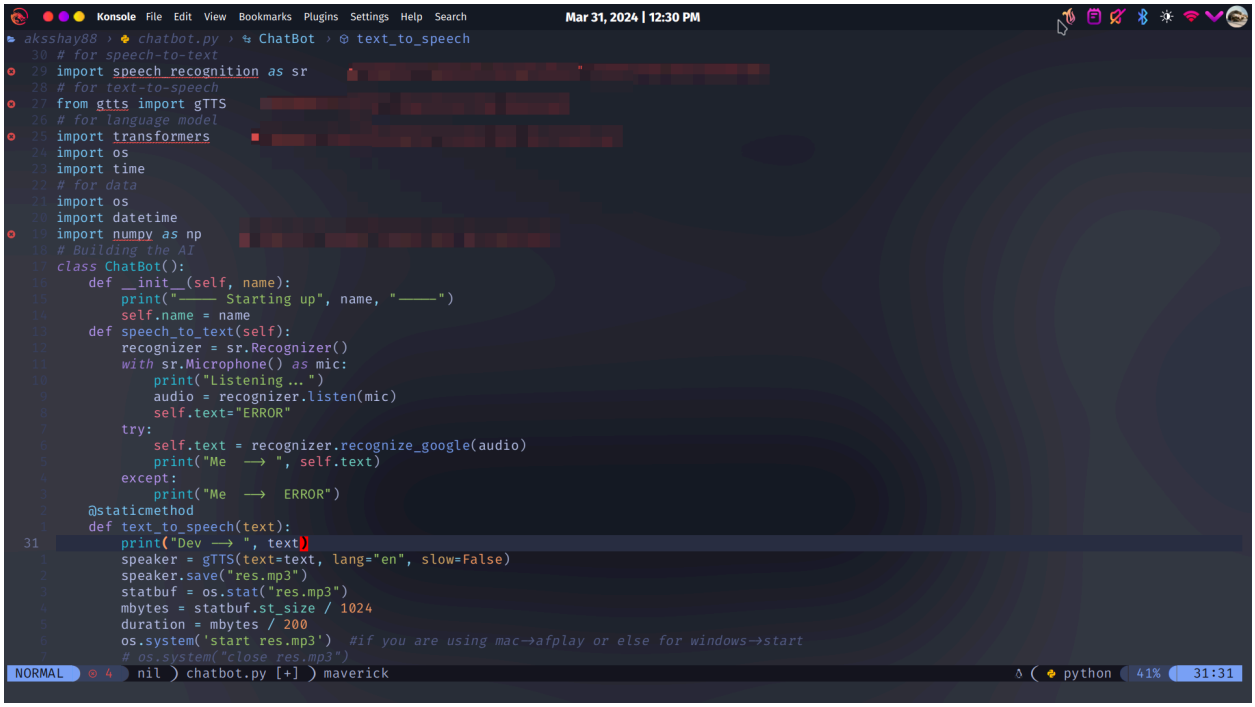
```
chat = nlp(transformers.Conversation(ai.text), pad_token_id=50256)

res = str(chat)

res = res[res.find("bot >> ")+6:].strip()
```

Finally, we're ready to run the Chatbot and have a fun conversation with our AI. Here's the full

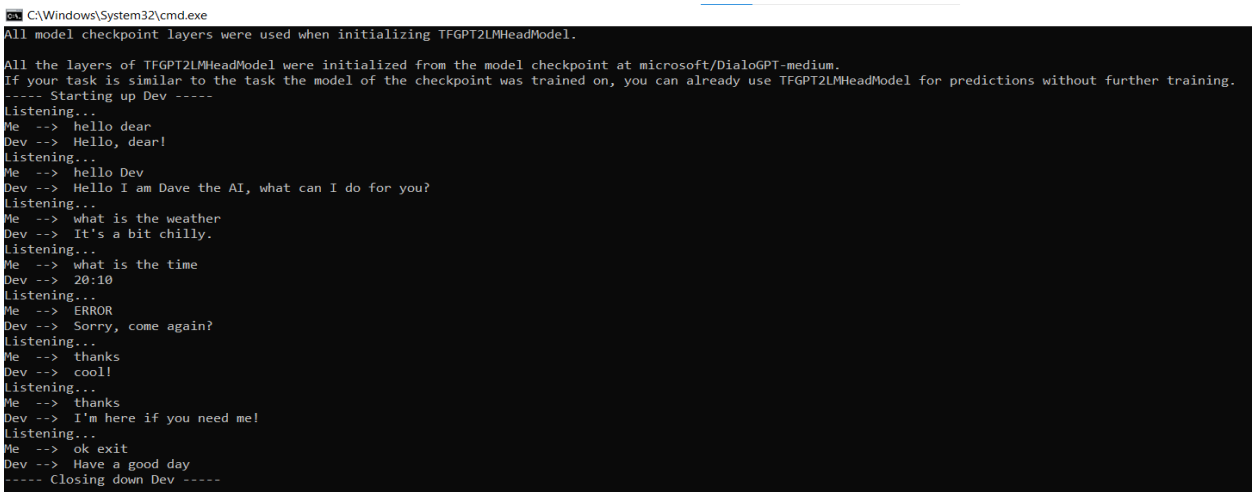
code:

```
----- starting up Dev -----
listening...
me --> Hello!
AI --> Hello :D
listening...
```

Great! The bot can both perform some specific tasks like a virtual assistant (i.e. saying the time when asked) and have casual conversations. And if you think that Artificial Intelligence is here to stay, she agrees:



Output:



Note: I have used google collab rather than my local machine or we can use Intel OneApi where I have 250$ credit points so that training time can be saved Obviously, Google is also there but the following lines will explain the issue. I used Python 3.9 as it had all the modules necessary and Python 3.6 and older versions will also work. Python 3.8 or the latest version might not have all the modules ported to match the version and hence I would suggest using Python 3.9 or older versions than 3.6.

To run a file and install the module, use the commands “python3.9” and “pip3.9” respectively if you have more than one version of Python for development purposes. “PyAudio” is another troublesome module and you need to manually

google and find the correct “.whl” file for your version of Python and install it using pip.

Pre-GSoC Preparation:

Prior to the official start of the Google Summer of Code (GSoC) project, the focus will be on laying the groundwork and preparing for the development of the Desktop Chat-Bot Application. **Here's the plan:**

Algorithm Development: Implement a prototype of the chat-bot algorithm as a precursor to the GSoC project. This involves designing and coding the core functionality of the chat-bot, including natural language processing (NLP) capabilities and response generation logic. This prototype will serve as a foundation for the subsequent development phases during GSoC.

Course Project Integration: Integrate the chat-bot algorithm into any relevant coursework or projects currently being undertaken. This will provide an opportunity to apply theoretical knowledge to practical implementation and gain valuable insights into the functioning and optimization of the chat-bot.

Completion Deadline: Ensure that the algorithm implementation is completed before the end of April, aligning with the timeline for the start of the GSoC project. Meeting this deadline will allow for a seamless transition into the official development phase and enable focused attention on refining and enhancing the chat-bot during GSoC.

Code Quality and Professionalism: Emphasize the importance of shipping professional-quality code by adhering to best practices in software development, such as writing clean and well-documented code, implementing efficient algorithms, and incorporating error handling mechanisms. This will set a strong foundation for the subsequent stages of the project.

Skill Enhancement: Utilize this pre-GSoC period to further enhance skills relevant to the project, such as familiarity with development frameworks and tools, proficiency in programming languages, and knowledge of NLP techniques. Engage in self-study, online courses, or practical exercises to deepen understanding and expertise in these areas.

By diligently completing these preparatory tasks, the stage will be set for a successful and productive engagement during the Google Summer of Code program, allowing for focused efforts on advancing the development of the Desktop Chat-Bot Application.

The project timeline outlines a structured plan for the development of the Desktop Chat-Bot Application, broken down into specific tasks and corresponding weeks:

1. **Week 1-2: Research and Setup:** The initial phase focuses on researching and familiarizing oneself with the Electron framework and OpenVINO toolkit. The development environment is set up during this period to prepare for subsequent tasks.

2. **Week 3-4: UI Design and Implementation:** In these weeks, the user interface (UI) for the Chat-Bot application is designed and implemented. The UI design aims to create an intuitive and user-friendly interface that enhances the overall user experience.

3. ****Week 5-6: NLP Integration and Testing**:** The pre-trained Neural Language Model is integrated into the application during this phase. Extensive testing is conducted to ensure the functionality and accuracy of the NLP model in understanding and generating responses to user inputs.

4. ****Week 7-8: OpenVINO Integration and Performance Testing**:** OpenVINO integration is undertaken to optimize the inference of the NLP model. Performance testing is conducted to evaluate the efficiency and responsiveness of the application, particularly in handling inference tasks.

5. ****Week 9-10: Error Handling and Testing:** Mechanisms for error handling are implemented to ensure the reliability and resilience of the application. Thorough testing is conducted to identify and address any issues or bugs, thereby enhancing the overall stability of the Chatbot.

6. ****Week 11-12: Documentation and Finalization:** The final weeks are dedicated to finalizing documentation, including installation instructions, usage guidelines, and technical details. Additionally, preparations are made for project

presentation and submission, ensuring that all deliverables are completed and ready for evaluation.

Personal Inspiration for the Project:

My passion for the Desktop Chat-Bot Application project originates from a desire to revolutionize user interaction paradigms and democratize access to intelligent conversational agents. This inspiration was ignited during my exposure to innovative technologies and experiences, such as my involvement in immersive workshops and interactions with cutting-edge research.

Throughout my journey, I've witnessed the transformative potential of conversational AI in enhancing productivity and facilitating seamless communication. However, I've also recognized the need for more accessible and user-friendly solutions in this domain. This realization was underscored by observing the challenges faced by individuals in navigating complex interfaces and accessing information efficiently.

Motivated by the opportunity to bridge this gap, I envisioned developing a desktop chat-bot application that prioritizes user experience and accessibility. By leveraging advanced NLP techniques and integrating offline functionality, I aim to empower users with a versatile and intuitive communication tool that enhances productivity and enriches their desktop computing experience.

My commitment to this project stems from a belief in its potential to transform how users interact with technology and my dedication to making it accessible to a diverse audience. By undertaking this project, I aspire to contribute to the advancement of conversational AI technology and empower individuals to harness the benefits of intelligent automation in their daily lives.