NLP - Voice Assistant

NLP Voice Assistant with GUI

End-of-semester project in the natural language processing class

Authors:
Brian Perret, Lenny Chiadmi-Delage

Acknowledgements

We would like to express our special thanks to Radosław Kycia. As our teacher for NLP classes, he taught us how to implement a chatbot using `Python`, which was fundamental to the success of this project.

Abstract

In this project, we developed a voice assistant with a graphical user interface (GUI) in Python. Our voice assistant can recognize spoken commands, process natural language inputs, and respond accordingly. This project combines state-of-the-art NLP techniques with a user-friendly interface to provide an interactive and intuitive user experience. The assistant is capable of performing tasks like answering predefined questions and opening applications such as WhatsApp and Microsoft Teams. This report outlines the purpose, scope, and methodologies of the project, followed by a detailed explanation of the theoretical background and practical implementation.

Introduction

2.1 Aim

The primary aim of this project was to create an interactive NLP voice assistant that can recognize and process spoken commands, respond appropriately, and perform specific tasks such as opening applications. The voice assistant was designed to provide a seamless and intuitive user experience by integrating advanced NLP techniques with a user-friendly GUI.

2.2 Scope

The scope of this project includes:

- Implementing a voice recognition system to capture and interpret spoken commands.
- Developing a text-to-speech system for the assistant to respond verbally.
- Creating a GUI to facilitate easy interaction with the assistant.
- Integrating functionalities to recognize and respond to predefined questions.
- Implementing commands to open specific applications (WhatsApp, Microsoft Teams).
- Testing the system on a Windows operating system.

2.3 Methodology

The methodology and tools used in this project include:

- **Programming Language**: Python
- **Voice Recognition**: `speech_recognition` library
- **Text-to-Speech**: `pyttsx3` library
- **NLP Techniques**: TF-IDF vectorization, cosine similarity using `scikit-learn`
- **GUI**: Flet library
- **Application Automation**: `AppOpener` library
- **Project Environment**: Virtual environment for dependency management

I. Theoretical Part

In this section, we discuss the theoretical foundations that were essential for solving the problem of creating a functional NLP voice assistant.

3.1 Natural Language Processing (NLP)

NLP is a field of artificial intelligence that focuses on the interaction between computers and humans through natural language. It involves several tasks such as speech recognition, language understanding, and language generation. In this project, we utilized NLP techniques to process and understand user queries and generate appropriate responses.

3.1.1 Speech Recognition

Speech recognition involves converting spoken language into text. We used the `speech_recognition` library in Python to capture and interpret spoken commands. This library provides access to several speech recognition engines, including CMU-Sphinx speech API.

3.1.2 Text-to-Speech

Text-to-speech (TTS) converts text into spoken language. We used the `pyttsx3` library, which is a text-to-speech conversion library in Python. It supports multiple TTS engines and allows customization of voice properties such as rate and volume.

3.1.3 TF-IDF Vectorization

Term Frequency-Inverse Document Frequency (TF-IDF) is a numerical statistic that reflects the importance of a word in a document relative to a collection of documents (corpus). We used TF-IDF to transform user queries and predefined responses into vector form, enabling efficient similarity computation.

3.1.4 Cosine Similarity

Cosine similarity measures the cosine of the angle between two non-zero vectors. It is used to determine the similarity between two vectors. In our project, cosine similarity was used to find the closest match between user queries and predefined responses.

II. Practical Part

In this section, we describe the practical implementation of the NLP voice assistant, present the results, and discuss our findings.

4.1 Implementation

4.1.1 Setting Up the Environment

We started by setting up a Python virtual environment to manage dependencies. The required libraries were installed using `pip`.

4.1.2 Voice Recognition and Text-to-Speech

The `speech_recognition` library was used to capture and convert spoken commands into text. The `pyttsx3` library enabled the assistant to respond verbally. The voice engine was configured to use a specific voice profile.

4.1.3 GUI Development

The GUI was developed using the Flet library, which provides a simple way to create web-based user interfaces. The main components of the GUI include a text field for user input and buttons to send commands or initiate voice recognition.

4.1.4 NLP Processing

We created a class, `ChatBotLogic`, to handle NLP tasks. The class uses TF-IDF vectorization to convert text into vectors and cosine similarity to match user queries with predefined responses. Specific commands to open applications were also implemented.

4.1.5 Integration

The GUI and NLP components were integrated to provide a seamless user experience. When the user speaks or types a command, the system processes the input and generates an appropriate response, which is then displayed and spoken.

4.2 Results

The NLP voice assistant successfully recognizes spoken commands, processes natural language inputs, and responds accordingly. It can perform specific tasks like opening WhatsApp and Microsoft Teams. The GUI facilitates easy interaction, making the assistant user-friendly.

4.2.1 Performance

The performance of the assistant was satisfactory during testing on a Windows system. The voice recognition was accurate, and the responses were appropriate for the predefined questions. The application-opening functionality worked as expected.

4.2.2 User Experience

The GUI was intuitive and easy to use. The text-to-speech responses added to the interactive experience, making the assistant feel more responsive and engaging.

Summary

The primary goal of developing an NLP voice assistant with a GUI was achieved. The assistant can recognize and process spoken commands, respond appropriately, and perform specific tasks. While the project was successful on a Windows system, future work could involve testing and optimizing the assistant for other operating systems. Additionally, expanding the range of commands and improving the NLP capabilities would further enhance the assistant's functionality.

Bibliography

1. `speech_recognition` library documentation: Speech Recognition Library
2. `pyttsx3` library documentation: pyttsx3 Library
3. `scikit-learn` documentation: Scikit-learn Library
4. `Flet` library documentation: Flet Library
5. `AppOpener` library documentation: AppOpener Library
6. `CMU Sphinx` library documentation: CMU Sphinx Library
7. Wikipedia article on TF-IDF: TF-IDF
8. Wikipedia article on Cosine Similarity: Cosine Similarity
9. Natural Language Toolkit (nltk) documentation: NLTK Library

By clearly defining the problem, leveraging suitable methodologies, and implementing an effective solution, we have created a functional and interactive NLP voice assistant that demonstrates the potential of combining NLP techniques with user interface.