Speech Recognition Final Project

Oral Ace 

English Speaking Practice Program Based on Speech Recognition and Large AI Models

Sutdents: Zhuobing Zhao, Yang Zhang, Xin Gao

Student Number: 2252750, 2253722, 2251356

Major: SoftWare Engineering, Tongji University

Teacher: Ying Shen

Time: 2024/12/27

Project Introduction
Content Analysis
1 Background and Needs Analysis
2 Related Work
Expected Outcomes
1 Recommendation Module
2 Learning Module
3 Practice Module
4 Record Module
Principles of Speech Recognition
1 Basic Principles
  1. 1 Sound Collection
  1. 2 Signal Processing
  1. 3 Feature Extraction
  1. 4 Acoustic and Language Models
  1. 5 Decoding and Text Output
  1. 6 Post-Processing
2 Key Technologies
  2. 1 Deep Learning in Speech Recognition
  2. 2 End-to-End Speech Recognition
3 Application in This Project
  3. 1 Audio Recording
  3. 2 Audio Upload and Conversion
  3. 3 Text Processing and Display
  3. 4 Error Handling and Optimization
4 Advantages of Whisper API
5 Implementation Challenges and Solutions
Design
1 Conceptual Design
2 Physical Design
Features
Technology Stack
System Architecture
Installation and Configuration
1 Prerequisites
2 Steps
Usage Instructions
Code Structure Explanation
1 1. `main.py`
2 2. `Speech2Text.py`
3 3. `Text2Speech.py`
4 4. Resource Files
API Documentation
1 OpenAI API
Frequently Asked Questions
1 Cannot Record or Sound Device Errors
2 API Call Failures or Slow Responses
3 Audio Playback Issues
4 Interface Display Issues or Crashes
Future Prospects

1. Project Introduction

This project aims to develop a speech recognition application to help users practice English speaking. Through preset dialogue scenarios, users can engage in simulated conversations. The system will recognize users' speech in real-time and convert it into text, enabling interaction with an AI assistant. Additionally, the system can convert the assistant's replies into speech, providing a more immersive learning experience.

2. Content Analysis

2.1. Background and Needs Analysis

With the acceleration of globalization and increasingly close international exchanges, English as an international language has become increasingly important. Currently, the English education system is gradually emphasizing comprehensive English proficiency to avoid the "mute English" phenomenon where learners can read and write but lack speaking skills. Oral exams have been incorporated into various English tests and everyday life, making fluent English speaking an indispensable skill in daily life and work.

To improve their speaking skills, more and more learners seek to provide themselves with an immersive speaking learning environment, creating opportunities to practice speaking, engage in conversations with native speakers, and learn pronunciation and intonation from their dialogue partners to enhance their oral abilities.

However, for many English learners, oral expression is often their weak point and a difficult area to improve. Traditional speaking learning methods, such as teaching by oral instructors, require highly qualified teachers, resulting in a scarcity of excellent instructors, large student-teacher ratios, and limited teaching time. This makes it impossible to achieve sufficient personalized teaching and oral improvement in the classroom. Additionally, considering the current situation, oral teaching, especially by foreign teachers, is expensive, bringing significant learning costs to learners. The "speaking buddy" approach is also limited by partners' proficiency and availability, making it difficult to achieve effective and continuous speaking practice.

With the rapid development of artificial intelligence technology, speech recognition, and AI large models provide new solutions for English speaking. We aim to develop an application based on these technologies to help oral learners practice speaking.

2.2. Related Work

We surveyed some existing speaking practice software on the market, including Duolingo, FluentU, Shanbay Listening and Speaking, Tomato English, and HelloTalk.

- **Duolingo**: The speaking practice primarily involves imitation reading. It is suitable for beginners, addressing learners' pronunciation and intonation issues. However, the difficulty of learning content cannot be customized, making it more suitable for learners with a certain foundation who need to undergo multiple cumbersome English

proficiency tests to align the learning content with their actual learning needs.

- **FluentU, Shanbay Listening and Speaking, Tomato English**: These applications offer speaking practices such as imitation reading, repetition, and dubbing, as well as paid live and recorded speaking classes. They help users accumulate commonly used phrases and expressions in different contexts but lack an immersive speaking dialogue environment. They cannot assess learners' conversational abilities and offer limited speaking practice scenarios.
- **HelloTalk**: Provides a platform to find native-speaking conversation partners but does not directly offer speaking learning and practice functions. Specific speaking practices require learners to communicate with conversation partners they find on the platform.

In summary, current popular speaking practice software lacks comprehensive functionality, focusing more on voice and intonation aspects without providing conversational speaking practice features. They fail to meet learners' conversational speaking needs and cannot effectively improve learners' conversational abilities. Additionally, although these applications inquire about learners' levels and goals during registration, they do not provide personalized learning content based on learners' levels. All learners receive the same recommended and practice content on the homepage, offering limited help for improving speaking skills.

3. Expected Outcomes

3.1. Recommendation Module

When users register, in addition to filling in basic information, the application will conduct a survey and assessment. The application will ask users about their learning purposes, including: practicing for speaking exams (such as middle school exams, college entrance exams, CET-4/6, TOEFL, IELTS, etc.), daily life scenarios, work scenarios, personal improvement, and others. The application will also perform a simple assessment of users' English levels, including vocabulary tests and pronunciation and intonation tests, combined with users' self-evaluation to obtain an initial assessment of their English proficiency.

Based on users' learning purposes and English levels, the application will recommend different learning content on the recommendation page. For example, if a user wants to improve their TOEFL speaking skills, the recommendation page will suggest TOEFL speaking course videos and entry points for TOEFL-type speaking dialogue practice functions.

3.2. Learning Module

In the learning module, the application primarily offers two modes: imitation reading and classroom.

- **Imitation Reading Mode**: Users can choose different content videos based on their learning purposes for imitation reading. For example, if a user wants to improve their speaking skills in daily life scenarios, they can choose videos related to daily life scenarios, such as shopping or sending packages, and imitate the dialogues in the videos to accumulate commonly used vocabulary and phrases. During imitation reading, the application will also score users' pronunciation to help them learn and improve better.

- **Classroom Mode**: The application provides three modes: short video knowledge points, video courses, and live courses. Users can choose to watch based on their needs. Short videos combine entertainment and learning, allowing users to learn a simple knowledge point within a few minutes. Video courses offer systematic speaking improvement courses, and live courses support real-time interaction, enhancing the interactivity and participation of learning.

3.3. Practice Module

In the practice module, users can choose different practice scenarios based on their learning purposes, including: practicing for speaking exams (such as middle school exams, college entrance exams, CET-4/6, TOEFL, IELTS, etc.), daily life scenarios, work scenarios, personal improvement, and others. In each scenario, users can engage in conversations with AI. After the conversation, AI will point out the errors users made during the dialogue, evaluate their current speaking level, and provide suggestions for improving their speaking skills.

3.4. Record Module

In the record module, the application automatically records the results of each practice session. Users can intuitively view the issues in their speaking, improvement suggestions, and changes in their speaking levels. Through records, users can better understand their learning progress and effectiveness, increasing their motivation and persistence in learning.

4. Principles of Speech Recognition

Speech recognition technology is the process of converting human speech signals into corresponding text, widely used in various voice interaction systems. In this project, speech recognition is one of the core features enabling users to practice speaking with AI. The following details the basic principles, key technologies, and specific applications of speech recognition in this project.

4.1. Basic Principles

A speech recognition system typically includes the following key steps:

1. **Sound Collection**
2. **Signal Processing**
3. **Feature Extraction**
4. **Acoustic and Language Models**
5. **Decoding and Text Output**
6. **Post-Processing**

4.1.1. Sound Collection

Sound collection is the first step in speech recognition, recording the user's speech input through a microphone to obtain raw audio signals. These signals are usually stored in digital formats like WAV or FLAC for subsequent processing. The choice of audio sampling rate and bit depth affects the recognition accuracy and system performance.

4.1.2. Signal Processing

Recorded audio signals often contain noise and unnecessary background sounds. Signal processing steps include:

- **Noise Reduction**: Removing background noise to improve the clarity of the speech signal.
- **Normalization**: Adjusting the amplitude of the audio signal to a uniform range, preventing overly loud or soft volumes from affecting recognition.
- **Silence Removal**: Detecting and removing silent segments in the audio to reduce unnecessary data processing.

4.1.3. Feature Extraction

Feature extraction converts audio signals into feature vectors suitable for machine learning models. Common feature extraction methods include:

- **Mel-Frequency Cepstral Coefficients (MFCC)**: Simulates the human auditory system, performing framing and Fourier transforms on the audio signal to extract spectral features.
- **Spectrogram**: Displays the frequency changes of the audio signal over time, helping capture the time-frequency information of speech.
- **Filter Bank Features**: Uses a set of filters to extract energy features from different frequency bands.

These features effectively represent the time-frequency characteristics of speech, providing strong support for subsequent model recognition.

4.1.4. Acoustic and Language Models

- **Acoustic Model**: Responsible for matching extracted feature vectors with corresponding phonemes or letters. Traditional acoustic models are based on Hidden Markov Models (HMM) and Gaussian Mixture Models (GMM). Modern acoustic models are often based on deep learning architectures such as Deep Neural Networks (DNN), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Transformers.
- **Language Model**: Responsible for understanding the grammatical and semantic relationships between words, predicting the probability of word sequences. Common language models include N-gram models and deep learning-based language models like BERT and the GPT series.

4.1.5. Decoding and Text Output

The decoder combines the outputs of the acoustic and language models to generate the most probable word sequence. This step typically involves optimization algorithms like the Viterbi algorithm or Beam Search to efficiently find the best match within a vast vocabulary space.

4.1.6. Post-Processing

Post-processing steps include spelling correction, punctuation restoration, and formatting to ensure the final output text is readable and accurate.

4.2. Key Technologies

4.2.1. Deep Learning in Speech Recognition

Deep learning technologies have significantly enhanced the accuracy and robustness of speech recognition. Key technologies include:

- **Convolutional Neural Networks (CNN)**: Excel at processing local features, suitable for extracting two-dimensional features like spectrograms.
- **Recurrent Neural Networks (RNN) and their variants (e.g., LSTM, GRU)**: Excel at handling sequential data, capturing temporal dependencies in speech signals.
- **Transformers**: Based on self-attention mechanisms, capable of parallel processing of sequence data, improving training efficiency and performance.

4.2.2. End-to-End Speech Recognition

Traditional speech recognition systems typically consist of multiple independent modules, whereas end-to-end speech recognition systems map audio signals directly to text using unified neural network models. This approach simplifies the process and enhances performance. Common end-to-end architectures include Connectionist Temporal Classification (CTC), Attention mechanisms, and Transformer models.

4.3. Application in This Project

This project uses OpenAI's Whisper API to implement speech recognition functionality. The specific application process is as follows:

4.3.1. Audio Recording

Users click the "Start Recording" button in the application interface to initiate audio recording. The application uses the PyAudio library to capture the user's speech input and save it as a WAV format audio file. Specific steps include:

- Initializing the PyAudio object and configuring recording parameters (e.g., sampling rate, number of channels, frame size).
- Starting the recording, capturing audio data in real-time, and storing it in a buffer.
- After the user stops recording, closing the audio stream and saving the audio file.

4.3.2. Audio Upload and Conversion

After recording, the application sends the audio file to OpenAI's Whisper API for processing. The Whisper model workflow includes:

- **Preprocessing**: Whisper performs noise reduction and normalization on the audio file to ensure input signal quality.
- **Feature Extraction**: The model extracts time-frequency features from the audio signal, such as MFCC and spectrograms.
- **Recognition and Transcription**: Using deep learning models, Whisper analyzes the extracted features to generate corresponding text transcription results.

After Whisper API returns the recognition results, the application displays the text content in the chat window for user reference.

4.3.3. Text Processing and Display

After receiving the recognition results from Whisper API, the application performs the following processing:

- **Text Display**: The recognized text content is displayed in the chat window of the user interface for easy viewing and confirmation.
- **Subsequent Processing**: The recognized text is used as input and passed to the GPT-4 model to generate intelligent replies, enabling conversational practice with AI.
- **Error Handling**: If errors occur during recognition (e.g., API call failure, network issues), the application prompts the user with appropriate error messages and suggests retrying or checking device settings.

4.3.4. Error Handling and Optimization

To ensure the accuracy of speech recognition and system stability, the project incorporates multi-layered error handling mechanisms during recording and conversion processes:

- **Device Detection**: Before recording starts, the application checks if the microphone device is properly connected and functioning, prompting users to perform necessary device checks.
- **Exception Catching**: During recording and audio processing, the application catches potential exceptions (e.g., audio device errors, file saving failures) and provides user-friendly error prompts.
- **Retry Mechanism**: For temporary errors, the application allows users to retry recording or API calls, ensuring process continuity.
- **Performance Optimization**: By optimizing audio sampling rates and frame sizes, the application balances recording quality and processing speed, enhancing overall user experience.

4.4. Advantages of Whisper API

Choosing OpenAI's Whisper API as the speech recognition solution offers the following advantages:

- **High Accuracy**: The Whisper model is trained on large-scale speech data, providing high accuracy and strong robustness to handle various accents and speech speeds.
- **Multilingual Support**: Supports speech recognition in multiple languages, catering to diverse user needs.
- **Real-Time Capability**: Whisper API has low latency, enabling real-time speech-to-text conversion and enhancing user interaction experience.
- **Ease of Integration**: Through API interfaces, Whisper can be easily integrated into existing applications, simplifying the development process.

4.5. Implementation Challenges and Solutions

In this project, achieving efficient and accurate speech recognition faces the following challenges, along with corresponding solutions:

- **Background Noise Interference**: Users may record in different environments with background noise. Solutions include incorporating noise reduction algorithms in the signal processing stage and selecting high-quality microphone devices.
- **Diverse Accents and Speech Speeds**: Users may have different accents and speech speeds, affecting recognition accuracy. The Whisper model, trained on large and diverse datasets, has good generalization capabilities to adapt to different speech inputs.
- **Real-Time Requirements**: Users expect immediate recognition results. By optimizing audio sampling rates and utilizing efficient API interfaces, the project ensures real-time speech recognition processes.
- **Error Handling**: Various errors may occur during the recognition process, such as network issues and API limitations. Establishing robust error handling mechanisms improves system stability and user experience.

Speech recognition technology plays a crucial role in this project. By integrating the advanced Whisper API, the project can efficiently and accurately convert users' speech inputs into text, providing a reliable foundation for subsequent intelligent dialogue and evaluation features. Combining deep learning technologies and an end-to-end speech recognition architecture not only enhances the performance of speech recognition but also simplifies the system development and maintenance processes. In the future, as speech recognition technology continues to advance, the project will further optimize recognition algorithms, improve user experience, and meet broader application needs.

5. Design

5.1. Conceptual Design

Through collection and analysis, user needs can be summarized as:

- Improve speaking skills in exams, daily life, work, etc.
- Enhance through conversational dialogue practice.
- Learning content matches users' English levels.
- Record users' learning progress.

To address these user needs, the application is designed as a desktop program, providing four functional modules: Recommendation, Learning, Practice, and Record, to meet user requirements. Users interact with the application primarily through clicking and voice or keyboard inputs. Users browse and select through clicking. In the speaking practice module, users need to use the keyboard to control the start and end of voice input after selecting a scenario.

The application's primary color scheme is set to blue, with a background color of #f7f9fc, and font colors of #000000, #ffffff, #2980b9, and #2c3e50. The fonts used are Cambria and Georgia. Blue is chosen as the primary color because it typically represents trust, reliability, and professionalism, helping users develop a sense of trust and immerse themselves in learning. The selected fonts are clear and readable, with variations in font style and size to differentiate content and prevent user fatigue and boredom from reading the same font for extended periods.

The application includes four functional components:

1. **User Management**: Handles user registration, login, and management of level and learning purpose information.
2. **Recommendation**: Recommends appropriate learning content based on users' learning purposes and English levels.
3. **Learning Content Management**: Manages and provides different learning resources, such as video courses and short video knowledge points.
4. **Dialogue**: Provides dialogue practice functions in different scenarios.
5. **Data Recording**: Records users' practice data.

The user's interaction model is as follows: Users enter the system through the registration and login interfaces. On the recommendation page, they see recommended learning content and click to enter the learning module. In the learning module, users choose imitation reading mode or classroom mode to learn. In the practice module, users select practice scenarios and use voice input to engage in dialogue practice with AI. In the record module, users view their practice records and progress.

5.2. Physical Design

Using this application involves functionalities such as registration and login, speaking practice, imitation reading learning, video course learning, recommendations, records, and personalized settings.

- **Registration and Login**: During registration, users set their avatars, accounts, and passwords by clicking. For subsequent personalized learning content recommendations, users need to select their learning purposes, current speaking levels, and expected speaking levels. The application also conducts speaking level

tests, including imitation reading and vocabulary tests, to assess users' levels.

- **Speaking Practice**: Provides different scenarios, such as speaking exam simulations and situational dialogue simulations. Users select scenario categories by clicking with the mouse, and the application randomly generates a scenario. Users use voice input to engage in spoken dialogue with AI, controlling the start and end of the conversation with mouse clicks and keyboard inputs. After the dialogue ends, AI generates evaluations and presents them through both spoken and text displays.
- **Imitation Reading Learning**: Offers different scenarios for users to practice imitation reading. Users use voice input to imitate reading and control the start and end of imitation reading with mouse clicks. After each imitation reading session, AI generates scores and points out areas for improvement. Users can also record commonly used phrases and sentences in specific scenarios for speaking practice.
- **Video Course Learning**: Provides a variety of video courses for users to learn. Users can choose short videos to learn simple knowledge points or opt for video courses for systematic speaking improvement. The application supports live courses, allowing users to choose more interactive learning methods.
- **Recommendation Function**: Based on users' learning purposes and speaking levels provided during registration, the application filters suitable learning content collections on the recommendation page, simplifying the process for users to find the desired functions.
- **Record Function**: Automatically records each practice session's results. Users can view their learning consistency through a calendar view, monitor changes in study time with line charts, and review each speaking practice's content, evaluations, and scores. Recording information is synchronized to the application's backend user data, updating users' current speaking levels.
- **Personalized Settings**: Records users' learning information provided during registration, including current status and learning purposes. Users can modify this information at any time, and the application will update the recommended content based on the latest information. If users feel that the current recommended content does not match their levels, they can retake the speaking level test to receive more appropriate learning content recommendations.

6. Features

- **Multiple Scenario Dialogues**: Offers various life scenarios, such as daily life, shopping, education, social life, and traveling abroad, allowing users to choose based on their needs.
- **Voice Recording and Recognition**: Users can record their speech using a recording button, and the system will convert it into text in real-time.
- **Intelligent Dialogue**: Integrates OpenAI's GPT model to provide intelligent dialogue responses, helping users practice English speaking.
- **Text-to-Speech**: The assistant's replies are converted into speech, making it convenient for users to listen and learn.

- **Evaluation and Feedback**: Evaluates the entire dialogue process, providing feedback and suggestions on grammar, vocabulary, fluency, and other aspects.

7. Technology Stack

- **Programming Language**: Python
- **Interface Framework**: PyQt5
- **Speech Processing**:
  - Recording: PyAudio
  - Speech Recognition: OpenAI Whisper API
  - Text-to-Speech: OpenAI TTS API, SoundDevice, SoundFile
- **Artificial Intelligence**: OpenAI GPT-4
- **Other Libraries**: qfluentwidgets, numpy

8. System Architecture

The system is primarily divided into the following modules:

1. **User Interface (UI)**:
   - Built using PyQt5, providing a user-friendly interaction interface.
   - Includes features such as scenario selection, recording controls, and dialogue display.
2. **Speech Recording and Recognition**:
   - Uses PyAudio for audio recording.
   - Recorded audio is converted to text via OpenAI Whisper API.
3. **Intelligent Dialogue**:
   - Utilizes OpenAI GPT-4 model to generate dialogue responses.
   - Dialogue content is displayed in real-time on the interface and played through the text-to-speech module.
4. **Text-to-Speech**:
   - Converts the assistant's text replies into speech using OpenAI TTS API and plays them to the user.
5. **Evaluation and Feedback**:
   - Evaluates the entire dialogue process and provides detailed feedback and improvement suggestions.

9. Installation and Configuration

9.1. Prerequisites

- **Operating System**: Windows 10 or higher, macOS, Linux
- **Python**: Version 3.9 or above

9.2. Steps

1. **Clone the Project**

```
git clone https://github.com/Notyourbing/OralAce.git
cd OralAce
```

2. **Create a Virtual Environment (Optional)**

```
python -m venv venv
source venv/bin/activate  # Linux/macOS
venv\Scripts\activate     # Windows
```

3. **Install Dependencies**

```
pip install -r requirements.txt
```

Example `requirements.txt` Content:

```
keyboard
openai
PyAudio
PyQt5
PyQt5_sip
PyQt-Fluent-Widgets
pyttsx3
SpeechRecognition
sounddevice
soundfile
numpy
```

> **Note**: On some operating systems, installing `pyaudio` may require additional steps or precompiled binaries. Please refer to the PyAudio Installation Guide for installation instructions.

4. **Configure API Keys**

In the `main.py`, `Speech2Text.py`, and `Text2Speech.py` files, ensure to replace `api_key` with your own OpenAI API key.

```
client = OpenAI(
    api_key="your_openai_api_key",
    base_url="https://api.chatanywhere.tech/v1"
)
```

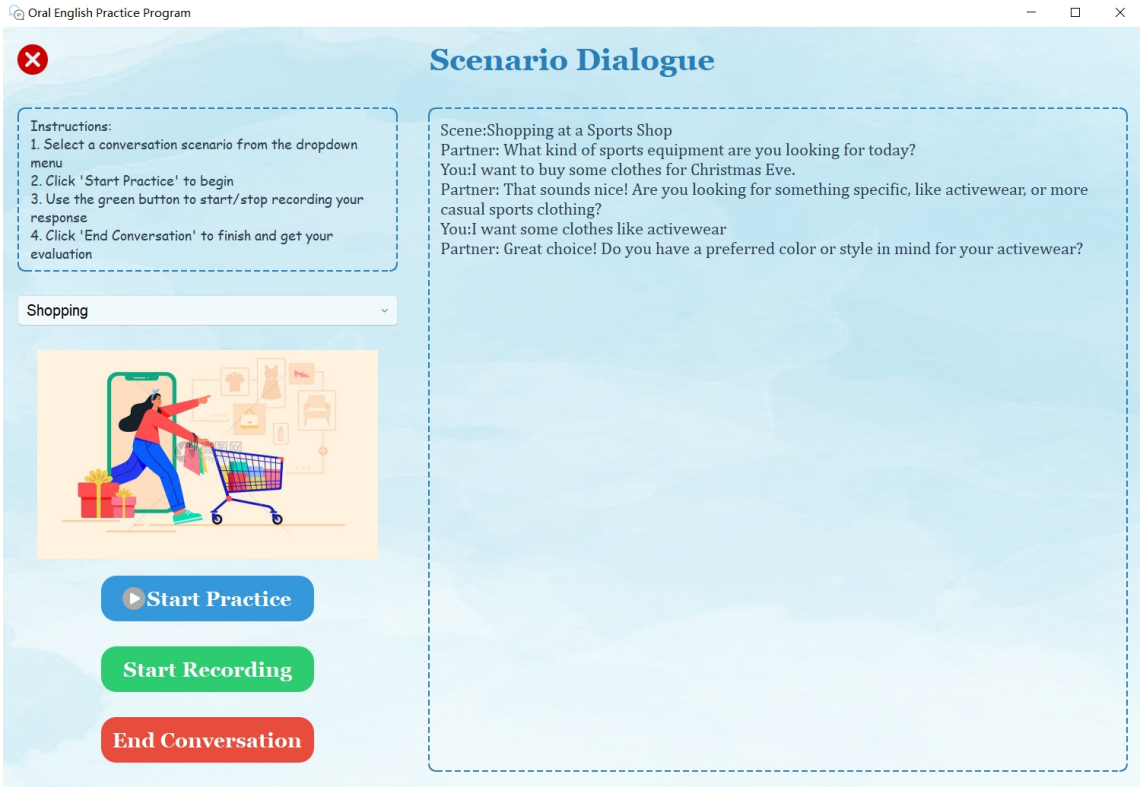
5. **Run the Project**

```
python main.py
```

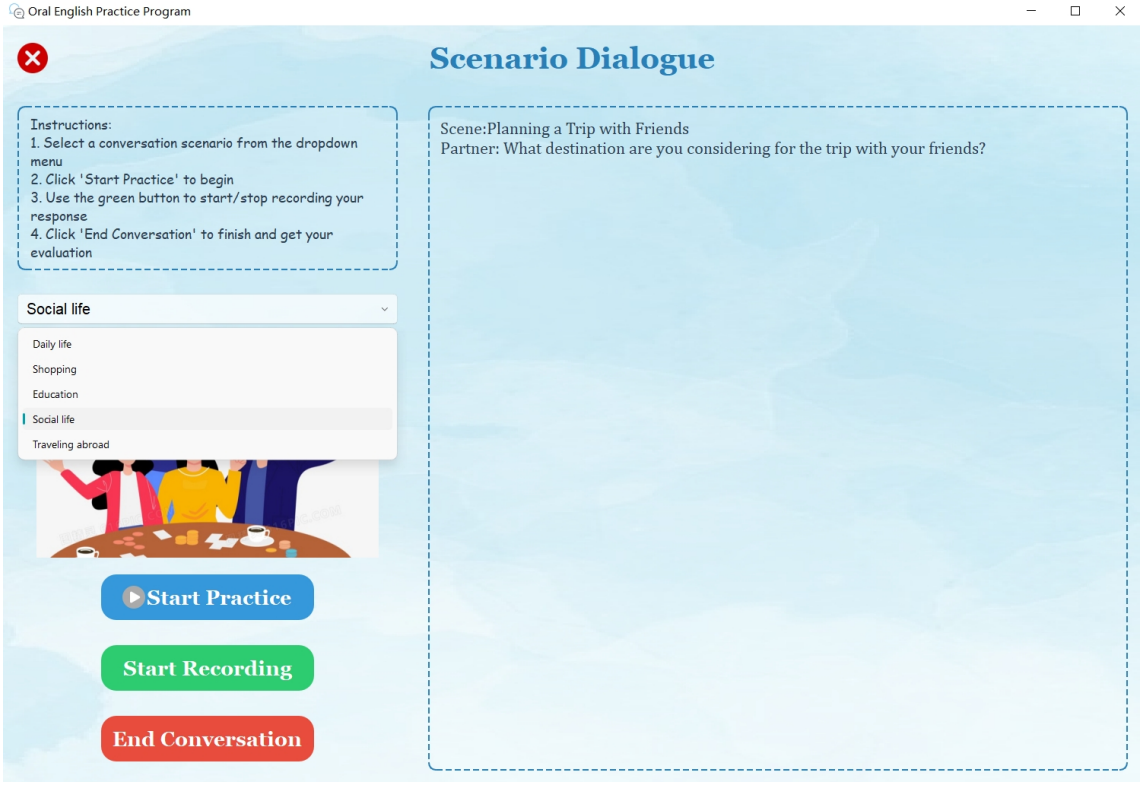
10. Usage Instructions

1. **Launch the Application**

   Run `main.py`, and the application window will appear.



2. **Select a Dialogue Scenario**



   Choose a dialogue scenario from the dropdown menu, such as "Daily Life," "Shopping," etc.

3. **Start Practice**

   Click the "Start Practice" button. The system will randomly select a specific dialogue context and display it on the interface.

4. **Recording and Dialogue**
    - Click the "Start Recording" button to begin recording. The system will listen and convert your speech into text.
    - After recording ends, the system will generate an intelligent reply and play it through text-to-speech.
    - Repeat this process to continue the dialogue practice.
5. **End Dialogue**

    Click the "End Conversation" button. The system will stop the dialogue loop and provide an evaluation and feedback on the entire dialogue process.

6. **Exit the Application**

    Click the close button at the top left corner to exit the application.

11. Code Structure Explanation

11.1. 1. `main.py`

The main program file responsible for building the user interface and handling user interaction logic.

- **Class `ChatApp`**:
    - Initializes interface elements, including titles, descriptions, scenario selection dropdowns, start buttons, recording buttons, exit buttons, chat display boxes, etc.
    - Handles user operations, such as starting practice, controlling recording, ending dialogue, etc.
    - Interacts with `Speech2Text.py` and `Text2Speech.py` to implement speech recognition and text-to-speech functionalities.

- Uses the OpenAI GPT-4 model to generate intelligent replies and perform dialogue evaluations.

11.2. 2. `Speech2Text.py`

Responsible for converting users' speech into text.

- **Function** `speech_to_text(chatApp)`:
  - Uses PyAudio to record user speech.
  - Converts the recorded audio to text using OpenAI Whisper API.
  - Returns the converted text.

11.3. 3. `Text2Speech.py`

Responsible for converting text into speech and playing it.

- **Function** `play_audio(audio_data)`:
  - Uses SoundDevice to play audio data.
- **Function** `text_to_speech(text)`:
  - Calls OpenAI TTS API to convert text into speech.
  - Starts a new thread to play the converted audio.

11.4. 4. Resource Files

- `image/`: Stores image resources required by the application, such as background icons, button icons, etc.

12. API Documentation

12.1. OpenAI API

The project uses several API services provided by OpenAI:

1. **ChatGPT (GPT-4)**
   - **Purpose**: Generate intelligent dialogue replies and dialogue evaluations.
   - **Endpoint**: `client.chat.completions.create`
   - **Parameters**:
     - `model`: Model name, e.g., `"gpt-4o-mini"`
     - `messages`: List of dialogue messages, including system messages, user messages, and assistant messages.
     - `max_tokens`: Maximum number of tokens for the reply.
2. **Whisper API**
   - **Purpose**: Convert users' speech recordings into text.
   - **Endpoint**: `client.audio.transcriptions.create`

- **Parameters**:
  - `model`: Model name, e.g., `"whisper-1"`
  - `file`: Audio file (in-memory WAV file).
  - `language`: Audio language, e.g., `"en"`.

3. **Text-to-Speech (TTS) API**
   - **Purpose**: Convert text replies into speech and play them.
   - **Endpoint**: `client.audio.speech.create`
   - **Parameters**:
     - `model`: Model name, e.g., `"tts-1"`
     - `voice`: Voice type, e.g., `"shimmer"`
     - `input`: Text content to convert.

> **Note**: Ensure the security of API keys. Do not expose API keys in public code repositories. It is recommended to manage API keys using environment variables or configuration files.

13. Frequently Asked Questions

13.1. Cannot Record or Sound Device Errors

Solution:

- Ensure the microphone is correctly connected and recognized by the system.
- Check if other applications are occupying the microphone device.
- Verify that PyAudio is correctly installed and compatible with the operating system.

13.2. API Call Failures or Slow Responses

Solution:

- Check if the network connection is stable.
- Ensure the OpenAI API key is correct and has sufficient call quotas.
- Check the status of OpenAI services to ensure the API is functioning properly.

13.3. Audio Playback Issues

Solution:

- Ensure that audio output devices (such as speakers or headphones) are correctly connected and set as the default output device.
- Check if SoundDevice and SoundFile libraries are correctly installed.
- Verify that the audio data format is compatible with the playback device.

13.4. Interface Display Issues or Crashes

Solution:

- Ensure all image resource files exist in the specified paths.
- Verify that PyQt5 is correctly installed and the version is compatible.
- Check console output for error messages to locate the issue.

14. Future Prospects

1. **Multilingual Support**: Extend support to speaking practice in other languages, such as Chinese, French, etc.
2. **User Account System**: Allow users to create accounts to save practice records and evaluation reports.
3. **Advanced Evaluation Features**: Introduce more comprehensive evaluation criteria, such as pronunciation accuracy and intonation.
4. **Real-Time Feedback**: Provide real-time voice feedback and improvement suggestions after users' recordings.
5. **Mobile Support**: Develop corresponding mobile application versions to enhance user convenience.
6. **Offline Mode**: Implement partial offline support for certain functionalities to improve application usability and privacy.

15. Conclusion

This project integrates speech recognition, natural language processing, and text-to-speech technologies to provide a comprehensive English speaking practice platform. By continuously optimizing and expanding, it aims to offer users a better learning experience and effectiveness.

If you have any questions or suggestions, please feel free to contact the project maintainers.

Authors: Zhuobing Zhao, Yang Zhang, Xin Gao
Contact Information: 252750@tongji.edu.cn
Project Homepage: https://github.com/Notyourbing/OralAce