Real-time Speech to Text to Speech : Building Your AI-Based Alexa

Gangzhaorige Li

github.com/gangzhaorige/ML-OPENAi-CustomerSupport/tree/main/Real-timeSpeechToTextToSpeech

Environment Setup (Python Libraries)

Requirements:

pydub

speech_recognition

whisper

gtts

click

openai

Initialize Open AI API Key

Create .env file and add your Open API Key

```
def init_api():
    with open(".env") as env:
        for line in env:
            key, value = line.strip().split("=")
            os.environ[key] = value

    openai.api_key = os.environ.get("API_KEY")
    openai.organization = os.environ.get("ORG_ID")
```

Main Idea

1. Recording Audio. (User voice from microphone)
2. Transcribing the Audio (Saving audio into queue)
3. Replying to User Request (For transcibed text)

Command line arguments

```
@click.command()
@click.option("--model", default="base", help="Model to use", type=click.Choice(["tiny", "base", "small", "
@click.option("--english", default=False, help="Whether to use the English model", is_flag=True, type=bool)
@click.option("--energy", default=300, help="Energy level for the mic to detect", type=int)
@click.option("--pause", default=0.8, help="Pause time before entry ends", type=float)
@click.option("--dynamic_energy", default=False, is_flag=True, help="Flag to enable dynamic energy", type=b
@click.option("--wake_word", default="hey computer", help="Wake word to listen for", type=str)
@click.option("--verbose", default=False, help="Whether to print verbose output", is_flag=True, type=bool)
```

Recording Audio

Record the Audio and enqueue for it for the processing.

```
def record_audio(audio_queue, energy, pause, dynamic_energy):
    r = sr.Recognizer()
    r.energy_threshold = energy
    r.pause_threshold = pause
    r.dynamic_energy_threshold = dynamic_energy

    with sr.Microphone(sample_rate=16000) as source:
        print("Listening...")
        i = 0
        while True:
            audio = r.listen(source)
            torch_audio = torch.from_numpy(np.frombuffer(audio.get_raw_data(),
            audio_data = torch_audio
            audio_queue.put_nowait(audio_data)
            i += 1
```

Transcribing Audio

Continuously transcribe audio, recognize the wake word, and enqueue the results.

```python
def transcribe_forever(audio_queue, result_queue, audio_model, english, wake_word, verbose):
    while True:
        audio_data = audio_queue.get()
        if english:
            result = audio_model.transcribe(audio_data, language='english')
        else:
            result = audio_model.transcribe(audio_data)

        predicted_text = result["text"]

        if predicted_text.strip().lower().startswith(wake_word.strip().lower()):
            pattern = re.compile(re.escape(wake_word), re.IGNORECASE)
            predicted_text = pattern.sub("", predicted_text).strip()
            punc = '''!()-[]{};:'"\,<>./?@#$%^&*_~'''
            predicted_text = predicted_text.translate({ord(i): None for i in punc})
            if verbose:
                print("You said the wake word.. Processing {}...".format(predicted_text))

            result_queue.put_nowait(predicted_text)
        else:
            if verbose:
                print("You did not say the wake word.. Ignoring")
```

Respond to the question by using Open AI API to generate a response audio.

```python
def reply(result_queue):
    while True:
        result = result_queue.get()
        data = openai.completions.create(
            model="gpt-3.5-turbo-instruct",
            prompt=result,
            temperature=0,
            max_tokens=150,
        )
        answer = data.choices[0].text
        mp3_obj = gTTS(text=answer, lang="en", slow=False)
        mp3_obj.save("reply.mp3")

        reply_audio = AudioSegment.from_mp3("reply.mp3")
        play(reply_audio)
        os.remove("reply.mp3")
```

How to run:
Run app.py by providing –wake word.

1. Use the wake word and followed by your question.
2. Automatically generates reply.mp3.
3. Plays the mp3.
4. Automatically deletes the mp3 file in the end.