Real-time Speech to Text to Speech : Building Your AI-Based Alexa - using Google's gTTS for Text-to-Speech

Emily Weng, 20016

Table of Contents

```
1   from pydub import AudioSegment
2   from pydub.playback import play
3   import speech_recognition as sr
4   import whisper
5   import queue
6   import os
7   import threading
8   import torch
9   import numpy as np
10  import re
11  from gtts import gTTS
12  import openai
13  import click
14
```

Install Required packages

- Install the needed package to run the code. If anything is missing, install it as well.

How to Run Code/Edits

```
def reply(result_queue):
    while True:
        result = result_queue.get()  # Get the user's input from the queue

        try:
            # Call the ChatCompletion API with the correct arguments
            data = openai.chat.completions.create(
                model="gpt-3.5-turbo",  # Specify the model
                messages=[
                    {"role": "system", "content": "You are a helpful assistant."},
                    {"role": "user", "content": result},
                ],
                temperature=0,  # Set temperature for response creativity
                max_tokens=150,  # Limit the length of the response
            )

            # Extract the assistant's reply
            answer = data.choices[0].message.content  # Correct way to access the reply content

            # Convert the text response to speech using gTTS
            mp3_obj = gTTS(text=answer, lang="en", slow=False)
            mp3_obj.save("reply.mp3")  # Save the audio response
            reply_audio = AudioSegment.from_mp3("reply.mp3")

            # Play the audio response
            play(reply_audio)

            # Clean up the temporary audio file
            os.remove("reply.mp3")

        except Exception as e:
            print(f"Error in generating or playing the response: {e}")
```

- data = openai.chat.completions.create
- answer = data.choices[0].message.content
- These needs to be changed to fit the current model version.

Enhance reply()

```
Tabnine | Edit | Test | Explain | Document | Ask
def reply(result_queue):
    while True:
        result = result_queue.get()  # Get the user's input from the queue

        try:
            # Check the cache first
            if result in cache:
                answer = cache[result]  # Retrieve the cached answer
                print(f"Retrieved from cache: {answer}")
            else:
                # Call the ChatCompletion API with the correct arguments
                data = openai.chat.completions.create(
                    model="gpt-3.5-turbo",  # Specify the model
                    messages=[
                        {"role": "system", "content": "You are a helpful assistant."},
                        {"role": "user", "content": result},
                    ],
                    temperature=0,  # Set temperature for response creativity
                    max_tokens=150,  # Limit the length of the response
                )

                # Extract the assistant's reply
                answer = data.choices[0].message.content.strip()  # ßCorrect way to access
                cache[result] = answer  # Store the result in the cache
                print(f"Cached new answer: {answer}")

            # Convert the text response to speech using gTTS
            mp3_obj = gTTS(text=answer, lang="en", slow=False)
            mp3_obj.save("reply.mp3")  # Save the audio response
            reply_audio = AudioSegment.from_mp3("reply.mp3")

            # Play the audio response
            play(reply_audio)

            # Clean up the temporary audio file
            os.remove("reply.mp3")
```

```
        except Exception as e:
            # Handle errors gracefully and provide a fallback response
            print(f"Error: {e}")
            fallback_choices = [
                "I'm sorry, I don't know the answer to that.",
                "I'm not sure I understand.",
                "Can you please rephrase the question?",
                "I need more information to help with that.",
            ]
            fallback_answer = fallback_choices[np.random.randint(0, len(fallback_choices))]
            print(f"Fallback answer: {fallback_answer}")

            # Convert fallback response to speech
            mp3_obj = gTTS(text=fallback_answer, lang="en", slow=False)
            mp3_obj.save("reply.mp3")  # Save the fallback audio
            reply_audio = AudioSegment.from_mp3("reply.mp3")

            # Play the fallback audio
            play(reply_audio)

            # Clean up the temporary audio file
            os.remove("reply.mp3")
```

This is done to get a better reply from gTTS

Run the code

Use this command to run the code in terminal:

```
python (file name).py --model base --english
--energy 300 \
 --pause 0.8 --dynamic_energy --wake_word
"hey computer" \
 --verbose
```

Output

```
Listening...
/Users/emilyweng/venv/lib/python3.11/site-packages/whisper/transcribe.py:132: UserWarning: FP16 is
not supported on CPU; using FP32 instead
  warnings.warn("FP16 is not supported on CPU; using FP32 instead")
Cached new answer: OpenAI is an artificial intelligence research laboratory consisting of the for-p
rofit OpenAI LP and the non-profit OpenAI Inc. It aims to ensure that artificial general intelligen
ce (AGI) benefits all of humanity. OpenAI conducts research in various areas of artificial intellig
ence and shares its findings with the public.
/Users/emilyweng/venv/lib/python3.11/site-packages/whisper/transcribe.py:132: UserWarning: FP16 is
not supported on CPU; using FP32 instead
  warnings.warn("FP16 is not supported on CPU; using FP32 instead")
```

```
Cached new answer: I am a helpful assistant here to provide you with information and assistance. Ho
w can I help you today?
/Users/emilyweng/venv/lib/python3.11/site-packages/whisper/transcribe.py:132: UserWarning: FP16 is
not supported on CPU; using FP32 instead
  warnings.warn("FP16 is not supported on CPU; using FP32 instead")
```

Conclusion

- Many parts of the code needed to be reviewed since it would change due to API version issue
- Reaction is a bit slow.

Github

https://github.com/emilywengster/sfbu/tree/effd2c0a27ed624d5599bc257a6aa94c1800c9c3/Machine%20Learning/ChatGPT/Customer%20Support%20System/Moderation%2C%20Classification%2C%20Checkout%20and%20Evaluation