SFBU Customer Support System - Speech to Text to Speech

Khoi Duong

Prof. Chang

CS589

3/30/2024

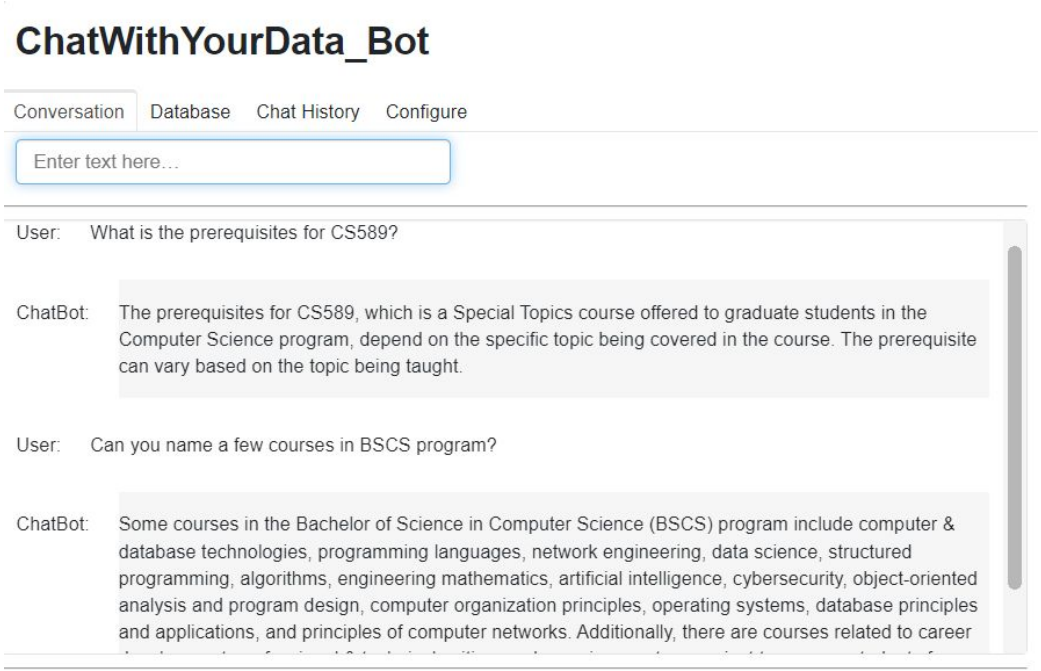
Table of contents

- Step 1: Implement SFBU Customer Support System - text
- Step 2: Implement Real-time Speech to Text to Speech : Building Your AI-Based Alexa - using OpenAI's TTS for Text-to-Speech
- Step 3: Enhance Step 2 by adding the features of the project implemented in Step 1.

7. SFBU Customer Support System - Speech to Text to Speech .
    - Process for the project implementation
        - Step 1: Implement SFBU Customer Support System - text
        - Step 2: Implement Real-time Speech to Text to Speech : Building Your AI-Based Alexa - using OpenAI's TTS for Text-to-Speech
            - Error: Concurrent Thread - Prachi Sethi, 2024 Spring
        - Step 3: Enhance Step 2 by adding the features of the project implemented in Step 1.
            - Hints: Two approaches on how to add the featurs
                - Option 1: Hard-coding the features on Step 2
                - Option 2: Using a library
                    - Instead of hard-coding the features on Step 2, a better idea is to implement the features as libraries which can be used for both Step 1 and Step 2.

Step 1: Implement SFBU Customer Support System - text

For step 1, please refer to the document about SFBU Customer Support System - text and follow the steps to complete



Step 2: Using OpenAI's TTS for Text-to-Speech

To implement OpenAI’s TTS, please refer to the document about [OpenAI TTS for Real-time Text-to-Speech] and follow the steps to complete

```python
from openai import OpenAI

client = OpenAI()

response = client.audio.speech.create(
    model="tts-1",
    voice="alloy",
    input="Hello world! This is a streaming test.",
)

response.stream_to_file("output.mp3")
```

```
(venv) PS D:\VS CODE\Python\CS589\TextSpeech> python app_openaiTTS.py --model base --english --energy 300
 --pause 0.8 --dynamic_energy --wake_word "hey computer" --verbose
D:\VS CODE\Python\CS589\TextSpeech\venv\lib\site-packages\pydub\utils.py:170: RuntimeWarning: Couldn't fi
nd ffmpeg or avconv - defaulting to ffmpeg, but may not work
  warn("Couldn't find ffmpeg or avconv - defaulting to ffmpeg, but may not work", RuntimeWarning)
Listening...
D:\VS CODE\Python\CS589\TextSpeech\venv\lib\site-packages\whisper\transcribe.py:115: UserWarning: FP16 is
 not supported on CPU; using FP32 instead
  warnings.warn("FP16 is not supported on CPU; using FP32 instead")
You did not say the wake word.. Ignoring
You said the wake word.. Processing ...
You said: who is the founder of OpenAI
The answer content: The founders of OpenAI are Elon Musk, Sam Altman, Greg Brockman, Ilya Sutskever, Wojc
iech Zaremba, John Schulman, and Chris Olah.
Transform the answer to mp3... Result will be in speech.mp3!
app_openaiTTS.py:123: DeprecationWarning: Due to a bug, this method doesn't actually stream the response
content, `.with_streaming_response.method()` should be used instead
  response.stream_to_file(speech_file_path)
```

Step 3: Enhance Step 2 by adding the features of the project implemented in Step 1.

```python
from openai import OpenAI
client = OpenAI()

sys.path.append('../..')
from dotenv import load_dotenv, find_dotenv
# read local .env file
_ = load_dotenv(find_dotenv())

openai.api_key  = os.environ['OPENAI_API_KEY']

def load_db(file, chain_type, k):
    # load documents
    loader = PyPDFLoader(file)
    documents = loader.load()
    # split documents
    text_splitter = RecursiveCharacterTextSplitter(
            chunk_size=1000,
            chunk_overlap=150)
    docs1 = text_splitter.split_documents(documents)
    # define embedding
    embeddings = OpenAIEmbeddings()
    # create vector database from data
    db = DocArrayInMemorySearch.from_documents(docs1,
            embeddings)
    # define retriever
    retriever = db.as_retriever(search_type="similarity",
            search_kwargs={"k": k})
    # create a chatbot chain. Memory is managed externally.
    qa = ConversationalRetrievalChain.from_llm(
        llm=ChatOpenAI(model="gpt-3.5-turbo", temperature=0),
        chain_type=chain_type,
        retriever=retriever,
        return_source_documents=True,
        return_generated_question=True,
    )
    return qa
```

scrollable

```python
# Define the cbfs class
class cbfs(param.Parameterized):
    chat_history = param.List([])
    answer = param.String("")
    db_query = param.String("")
    db_response = param.List([])

    def __init__(self, **params):
        super(cbfs, self).__init__(**params)
        self.panels = []
        self.loaded_file = "D:/VS CODE/Python/CS589/TextSpeech/2024Catalog.pdf"
        self.qa = load_db(self.loaded_file, "stuff", 4)
        self.recognizer = sr.Recognizer()
        self.recognizer.energy_threshold = 300
        self.recognizer.pause_threshold = 0.8
        self.recognizer.dynamic_energy_threshold = True

    def start_recording(self):
        with sr.Microphone(sample_rate=16000) as source:
            print("Listening...")
            audio = self.recognizer.listen(source)
            torch_audio = torch.from_numpy(np.frombuffer(audio.get_raw_data(), np.int16).flatten().astype(np.float32) / 32768.0)
        return torch_audio

    def transcribe_audio(self, audio_data):
        audio_model = whisper.load_model("base")
        result = audio_model.transcribe(audio_data, language='english')
        predicted_text = result["text"]
        return predicted_text

    def convchain_with_speech(self, event):
        # print(event)
        audio_data = self.start_recording()
        text = self.transcribe_audio(audio_data)
        return self.convchain(text)
```

```python
##############################################################
# Step 7.1.2.2: call_load_db function
##############################################################
def call_load_db(self, count):
    # init or no file specified :
    if count == 0 or file_input.value is None:
        return pn.pane.Markdown(f"Loaded File: {self.loaded_file}")
    else:
        file_input.save("temp.pdf")  # local copy
        self.loaded_file = file_input.filename
        button_load.button_style="outline"
        self.qa = load_db("temp.pdf", "stuff", 4)
        button_load.button_style="solid"
    self.clr_history()
    return pn.pane.Markdown(
        f"Loaded File: {self.loaded_file}")

##############################################################
# Step 7.1.2.3: convchain(self, query) function
##############################################################
def convchain(self, query):
    if not query:
        return pn.WidgetBox(pn.Row('User:',
            pn.pane.Markdown("", width=600)), scroll=True)
    result = self.qa({"question": query,
                      "chat_history": self.chat_history})
    self.chat_history.extend([(query, result["answer"])])
    self.db_query = result["generated_question"]
    self.db_response = result["source_documents"]
    self.answer = result['answer']
    self.panels.extend([
        pn.Row('User:', pn.pane.Markdown(query, width=600)),
        pn.Row('ChatBot:', pn.pane.Markdown(self.answer,
            width=600,
            styles={'background-color': '#F6F6F6'}))
    ])
    inp.value = ''  #clears loading indicator when cleared
    return pn.WidgetBox(*self.panels, scroll=True)
```

```python
##############################################################
# Step 7.1.2.4: get_lquest(self) function
##############################################################
@param.depends('db_query ', )
def get_lquest(self):
    if not self.db_query :
        return pn.Column(
            pn.Row(pn.pane.Markdown(f"Last question to DB:",
            styles={'background-color': '#F6F6F6'})),
            pn.Row(pn.pane.Str("no DB accesses so far"))
        )
    return pn.Column(
        pn.Row(pn.pane.Markdown(f"DB query:",
        styles={'background-color': '#F6F6F6'})),
        pn.pane.Str(self.db_query )
    )

##############################################################
# Step 7.1.2.5: get_sources function
##############################################################
@param.depends('db_response', )
def get_sources(self):
    if not self.db_response:
        return
    rlist=[pn.Row(pn.pane.Markdown(f"Result of DB lookup:",
        styles={'background-color': '#F6F6F6'}))]
    for doc in self.db_response:
        rlist.append(pn.Row(pn.pane.Str(doc)))
    return pn.WidgetBox(*rlist, width=600, scroll=True)

##############################################################
# Step 7.1.2.6: get_chats function
##############################################################
@param.depends('convchain', 'clr_history')
def get_chats(self):
    if not self.chat_history:
        return pn.WidgetBox(
              pn.Row(pn.pane.Str("No History Yet")),
               width=600, scroll=True)
    rlist=[pn.Row(pn.pane.Markdown(
        f"Current Chat History variable",
        styles={'background-color': '#F6F6F6'}))]
```

```python
#########################################################
# Step 7.1.2.6: get_chats function
#########################################################
@param.depends('convchain', 'clr_history')
def get_chats(self):
    if not self.chat_history:
        return pn.WidgetBox(
              pn.Row(pn.pane.Str("No History Yet")),
               width=600, scroll=True)
    rlist=[pn.Row(pn.pane.Markdown(
        f"Current Chat History variable",
        styles={'background-color': '#F6F6F6'}))]
    for exchange in self.chat_history:
        rlist.append(pn.Row(pn.pane.Str(exchange)))
    return pn.WidgetBox(*rlist, width=600, scroll=True)

#########################################################
# Step 7.1.2.7: clr_history function
#########################################################
def clr_history(self,count=0):
    self.chat_history = []
    return

#########################################################
# Step 7.1.2.8: text_to_speech function
#########################################################
def text_to_speech(self, verbose=True):
    text_to_speak = self.answer

    # Set the output filename (modify as needed)
    speech_file_path = Path(__file__).parent / "speech_SFBU.mp3"

    # Use OpenAI TTS API with desired voice and language
    response = client.audio.speech.create(
        model="tts-1",  # Adjust model if needed (check OpenAI documentation)
        voice="nova",  # Choose a voice from available options
        input=text_to_speak,
    )

    response.stream_to_file(speech_file_path)
```

```python
# Instantiate the cbfs class
cb = cbfs()

# Define the Panel GUI components
file_input = pn.widgets.FileInput(accept='.pdf')


#########################################################
# Step 7.2.2: Create buttons
#########################################################
button_load = pn.widgets.Button(name="Load DB",
      button_type='primary')
button_clearhistory = pn.widgets.Button(name="Clear History",
       button_type='warning')
button_clearhistory.on_click(cb.clr_history)
inp = pn.widgets.TextInput( placeholder='Enter text here…')
microphone_button = pn.widgets.Button(name='Speak to microphone', button_type='primary')
microphone_button.on_click(cb.convchain_with_speech)
bound_button_load = pn.bind(cb.call_load_db,
         button_load.param.clicks)
# Define a button to trigger text-to-speech conversion
button_tts = pn.widgets.Button(name="Convert to Speech", button_type="primary")
button_tts.on_click(cb.text_to_speech)


#########################################################
# Step 7.2.3: Create conversation
#########################################################
conversation = pn.bind(cb.convchain, inp)
#########################################################
# Step 7.2.4: Create jpg_pane
#########################################################
jpg_pane = pn.pane.Image('D:\VS CODE\Python\CS589\TextSpeech\BayleyBayhawk.jpg')
```

```
############################################################
# Step 7.2.5: Create tables
############################################################
tab1 = pn.Column(
    pn.Row(inp, microphone_button),
    pn.layout.Divider(),
    pn.panel(conversation, loading_indicator=True, height=300),
    pn.layout.Divider(),
    pn.Row(button_tts)
)
tab2 = pn.Column(
    pn.panel(cb.get_lquest),
    pn.layout.Divider(),
    pn.panel(cb.get_sources),
)
tab3 = pn.Column(
    pn.panel(cb.get_chats),
    pn.layout.Divider(),
)
tab4 =pn.Column(
    pn.Row( file_input, button_load, bound_button_load),
    pn.Row( button_clearhistory, pn.pane.Markdown(
        "Clears chat history. Can use to start a new topic" )),
    pn.layout.Divider(),
)

############################################################
# Step 7.2.6: Create dashboard
############################################################
dashboard = pn.Column(
    pn.Row(pn.pane.Markdown('# ChatWithYourData_Bot')),
    pn.Row(jpg_pane.clone(width=200)),
    pn.Tabs(('Conversation', tab1), ('Database', tab2),
        ('Chat History', tab3),('Configure', tab4))
)

# print(dashboard)
dashboard.servable()
```

Now, run the program with ***panel serve SFBU_CusSp_TTS.py***

```
(venv) PS D:\VS CODE\Python\CS589\TextSpeech> panel serve .\SFBU_CusSp_TTS.py
2024-04-07 02:00:51,076 Starting Bokeh server version 3.1.1 (running on Tornado 6.4)
2024-04-07 02:00:51,083 User authentication hooks NOT provided (default user enabled)
2024-04-07 02:00:51,086 Bokeh app running at: http://localhost:5006/SFBU_CusSp_TTS
2024-04-07 02:00:51,086 Starting Bokeh server with process id: 31908
d:\vs code\python\cs589\textspeech\venv\lib\site-packages\pydub\utils.py:170: RuntimeWarning: Couldn't find ffmpeg or
 avconv - defaulting to ffmpeg, but may not work
  warn("Couldn't find ffmpeg or avconv - defaulting to ffmpeg, but may not work", RuntimeWarning)
```

ChatWithYourData_Bot

Conversation | Database | Chat History | Configure

Enter text here... | Speak to microphone

User: Can you tell me a few subjects in the San Francisco Bay University?

ChatBot: At San Francisco Bay University, some of the subjects offered include Thinking, Public Speaking, Small Group Communication, Intercultural Communication, Modern American Literature, Calculus, Statistics, Physical Sciences, Physics, Introduction to Philosophy, Art Appreciation, Music Appreciation, Principles of Ethics, California History, Introduction to Sociology, Introduction to Psychology, and Emotional Intelligence.

User: Where is SFBU located?

ChatBot: San Francisco Bay University is located in Alameda, California.

Convert to Speech

ChatWithYourData_Bot

Conversation | Database | Chat History | Configure

Enter text here... | Speak to microphone

User: Can you tell me a few subjects in the San Francisco Bay University?

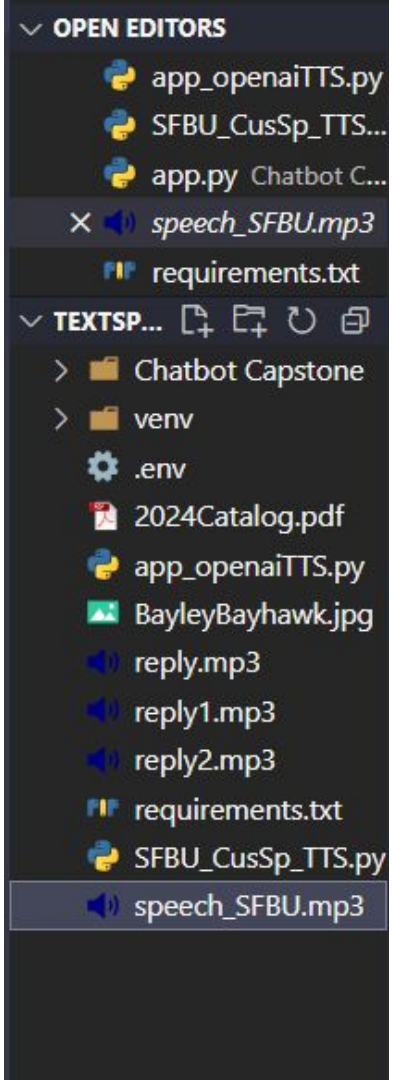
ChatBot: At San Francisco Bay University, some of the subjects offered include Calculus, Statistics, Physical Sciences, Physics, Introduction to Philosophy, Art Appreciation, Music Appreciation, Principles of Ethics, California History, Introduction to Sociology, Introduction to Psychology, Emotional Intelligence, Modern American Literature, Public Speaking, Small Group Communication, Intercultural Communication, and Thinking.

User: When does the summer trimester start and end?

ChatBot: I don't have the specific information about the start and end dates of the summer trimester at San Francisco Bay University. It would be best to check the university's official academic calendar or contact the university directly for this information.

Convert to Speech

The speech_SFBU.mp3 is created in the folder



References

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

Original repo: https://github.com/MynameisKoi/CS589/tree/main/TextSpeech

Source code: SFBU_CusSp_TTS.py

Result: speech_SFBU.mp3