DS565 - Generative AI-Driven Intelligent Apps Development

Project: ChatBot For SFBU

Course: DS565
Professor: Dr. Henry Chang

Presented by: Karan Shrestha (20087)

Project Overview

- **Objective**: Create a web-based chatbot with a conversational interface to assist users with inquiries and tasks.
- **Description**: A chatbot powered by natural language processing (NLP) and integrated with external data sources (e.g., documents, URLs) to provide intelligent, context-aware responses.

Project Workflow

Step 1: Data Loading

- Import data from PDFs, web URLs, and other sources.

Step 2: Vectorization

- Convert text into vector representations for efficient similarity search.

Step 3: Similarity Search

- Retrieve relevant information based on user queries.

Step 4: Generate Response/Speech to Text to Speecg

- Use a pre-trained language model (e.g., GPT) to formulate responses.

Step 5: Display Response

- Present the information in a user-friendly web interface.

Technical Architecture

Components:

- **Frontend**: HTML, CSS, and JavaScript for user interaction.
- **Backend**: Flask framework to handle API requests.
- **Vector Database**: FAISS or other libraries for efficient similarity search.
- **NLP Model**: OpenAI GPT or similar models for language understanding and generation.

Data Flow:

- User → Chat Interface → Backend API → Vector DB → NLP Model → Chat Interface->Speech to Text to Speech.

Data Sources

- **PDF Documents**: Uploaded by the user or administrator, processed for text extraction.
- **Web URLs**: Loaded dynamically for relevant content extraction.
- **YouTube Transcripts**: Audio content transcribed and used for Q&A.

Features of the Chatbot

- **Natural Language Understanding**: Recognizes and processes user input in natural language.
- **Contextual Responses**: Uses context from previous messages to maintain conversation flow.
- **Data Integration**: Pulls information from multiple sources like PDFs and web URLs.
- **Responsive UI**: User-friendly interface with real-time responses and typing indicators.
- **Loading Spinner**: Indicates when the chatbot is processing information.

Code

```python
# Function to load documents
def load_documents():
    docs = []
    pdf_loader = PyPDFLoader(file_path="data/sfbu-catalog.pdf")
    docs.extend(pdf_loader.load())
    with open('data/urls.txt', 'r') as file:
        urls = file.read().splitlines()
        url_loader = UnstructuredURLLoader(urls=urls)
        docs.extend(url_loader.load())
    return docs
```

```python
@app.route('/ask', methods=['POST'])
def ask():
    global chat_history
    question = request.json.get("question")
    inputs = {
        "question": question,
        "chat_history": chat_history
    }

    response = qa_chain.invoke(inputs)
    chat_history.append((question, response['answer']))
    return jsonify({"answer": response['answer']})
```

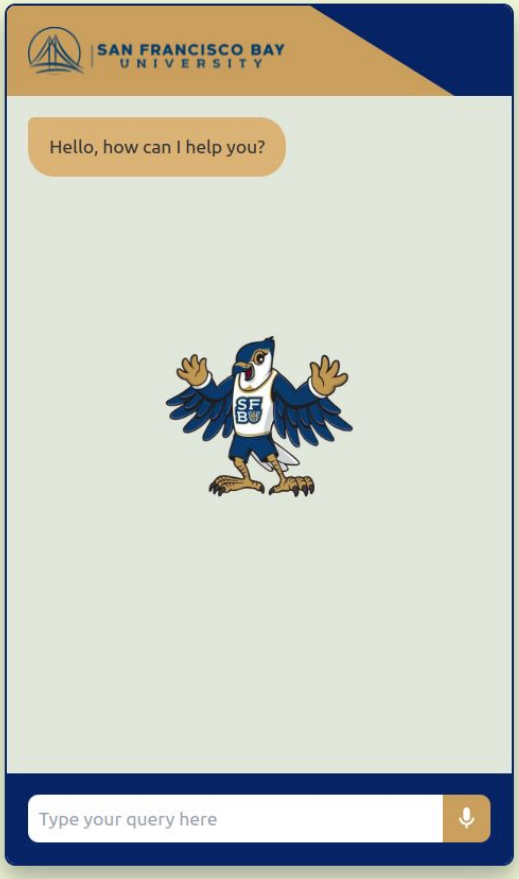
Code

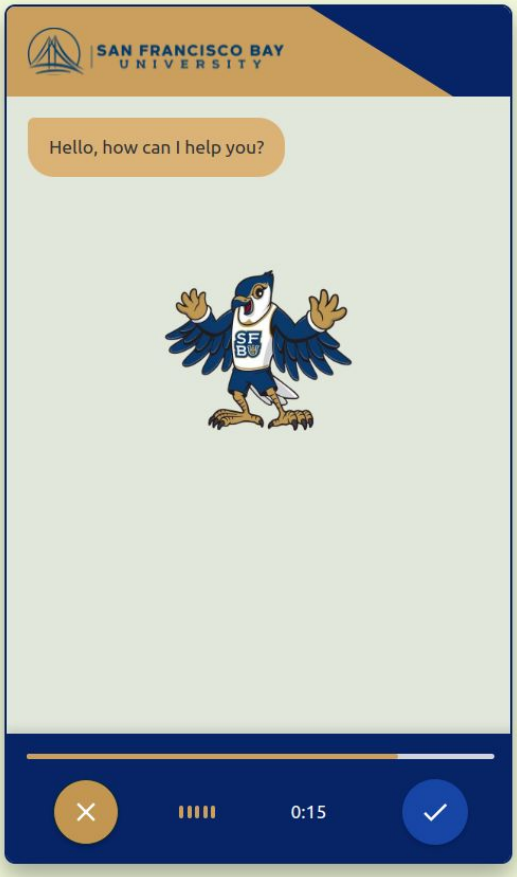
```
# Create vectorstore from documents
documents = load_documents()
vectorstore = FAISS.from_documents(documents, embedding_model)

# Use SimpleMemory instead of ConversationMemory
conversation_memory = SimpleMemory()

# Create conversational retrieval chain
qa_chain = ConversationalRetrievalChain.from_llm(
    llm=llm,
    retriever=vectorstore.as_retriever(),
    memory=conversation_memory
)
chat_history = []
```

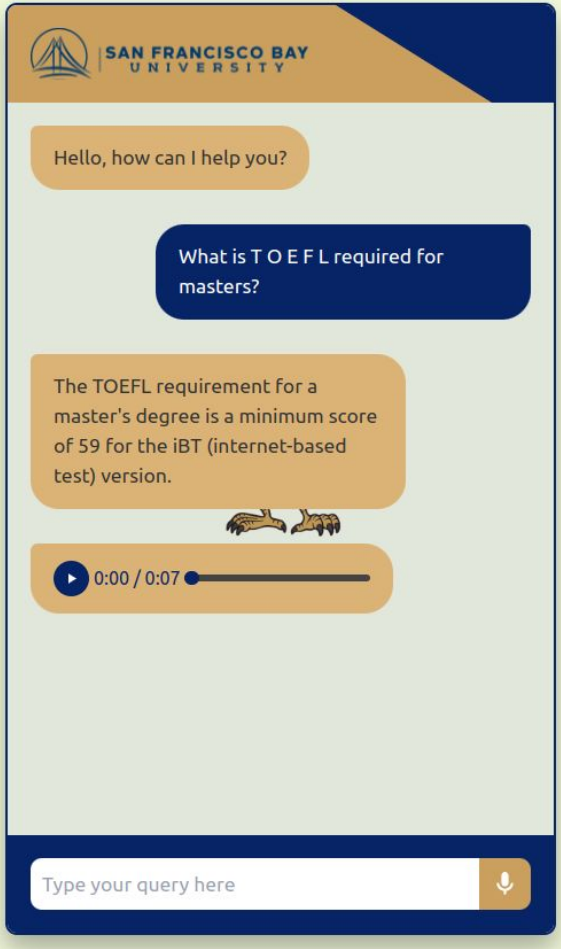
Code

```python
def generate_tts(text, output_dir="static/audio"):
    global previous_audio_file
    os.makedirs(output_dir, exist_ok=True)
    unique_filename = f"response_{uuid.uuid4().hex}.mp3"
    # unique_filename = f"response.mp3"
    filepath = os.path.join(output_dir, unique_filename)
    if previous_audio_file and os.path.exists(previous_audio_file):
        try:
            os.remove(previous_audio_file)
        except Exception as e:
            print(f"Error deleting file {previous_audio_file}: {e}")

    # Update the previous file tracker
    previous_audio_file = filepath

    with client.audio.speech.with_streaming_response.create(
        model="tts-1",
        voice="onyx",
        input=text,
    ) as response:
        response.stream_to_file(filepath)

    return filepath  # Return the full path of the audio file


```

```python
model = whisper.load_model("base")


def transcribe_audio(audio_file):
    try:
        # Resolve the absolute path to the `temp` folder
        base_dir = os.path.dirname(os.path.abspath(__file__))  # Get directory of `utility.py`
        temp_dir = os.path.join(base_dir, "temp")  # Resolve the temp directory path

        # Ensure the temp directory exists
        os.makedirs(temp_dir, exist_ok=True)

        # Save the audio file to the `temp` directory
        filename = secure_filename(audio_file.filename)
        temp_path = os.path.join(temp_dir, filename)
        audio_file.save(temp_path)
        transcription = model.transcribe(temp_path,fp16=False)
        return transcription['text']

    except Exception as e:
        return f"Error during transcription: {str(e)}"

```


SAN FRANCISCO BAY UNIVERSITY
Hello, how can I help you?
Type your query here


SAN FRANCISCO BAY UNIVERSITY
Hello, how can I help you?
0:15


SAN FRANCISCO BAY UNIVERSITY
Hello, how can I help you?
What is T O E F L required for masters?
The TOEFL requirement for a master's degree is a minimum score of 59 for the iBT (internet-based test) version.
0:00 / 0:07
Type your query here