Assistive Voice Assistant

Abishek Ajai Satnur and Charles Bruner
CS 4624: Multimedia, Hypertext, and Information Access
Professor: Edward A. Fox
Virginia Tech, Blacksburg VA 24061
April 30, 2024
Outline

● Project Description
● Requirements
● Design/Implementation
● Code Walkthrough
● Results
● Future Work
● Lessons Learned
Problem

Accessibility Challenges
Traditional drive-through systems fail to cater to customers with language barriers and hearing/speech impairments, leading to frustrations and communication gaps [1].

Client

Name
Levern Currie, Virginia Tech ISE doctoral student and CEO of Drivingo [2].

Objective
To develop an innovative all-in-one ordering kiosk that enhances accessibility using touch, hand gestures, and audio-based ordering.
Motivation

Focus
Refine the audio-based ordering component to be more accessible for customers with communication barriers.

Technology
Utilize conversational AI and NLP with GPT-4 Turbo to simulate human interaction.

Approach

Integration
Incorporate OpenAI’s GPT-4 to understand and process orders in multiple languages (French, English, German, Spanish, Chinese).

Goal
Ensure adaptability and efficiency for any fast-food chain, improving inclusivity and service efficiency in drive-throughs.
## Requirements

### Conversational Ordering System

**Objective**  
Accurately interpret complex customer orders with minimal repetition.

**Feature**  
Dynamic voice assistant for natural, conversational interactions.

### Multilingual Interface

**Support**  
Automatic language detection and response in English, Chinese, German, French, and Spanish.

**User Experience**  
Dynamic updates of menu displays based on detected language.

### Adaptive Communication Modes

**Modes**  
Support for voice, touch, and hand gestures.

**Accessibility**  
Seamless mode switching for an inclusive user experience.

### Integration and Scalability

**Integration**  
Must align with Drivingo’s existing infrastructure.

**Scalability**  
Adaptable to different fast-food chains and menu specifics.
Requirements Continued

Natural Language Processing and Machine Learning

Technology
Utilization of NLP and machine learning for precise order processing.

Development
Continuous learning from interactions to improve accuracy and adapt to new trends.

Testing and Model Training

Testing
Rigorous validation across languages and slang.

Training
Ongoing refinement based on real interactions to enhance system performance.

Expected Impact

Goal
Improve accessibility and efficiency in customer service interactions.

Benefit
Bridge communication gaps and cater to diverse customer needs.
Why we chose GPT-4 Turbo

- Ranked number one on LMSYS for chatbot purposes [3].
- The assistants API proved to be highly competent in threading of context in conversations and response management.
We conducted personal benchmark tests...

- By feeding GPT-4 turbo, GPT-3.5, LLama 2, Gemini, and Opus with 1000 restaurant related questions and checked to see what intent was detected [4, 5, 6].
- GPT-4 turbo did the best with 993 of those 1000 questions being detected for the right intent.

- We tested the language model on 1,000 questions, distributing 200 questions evenly across English, Mandarin, French, German, and Spanish to evaluate its understanding of various languages.
- GPT-4 turbo performed the best as a multilingual LLM.
Note:
Participation in this project required signing an NDA, restricting the disclosure of specific implementation details.
Client interacts with React frontend

"Hey! Can I get two fries?"

User message sent to backend

React

JSON containing reply and order state

Flask server

DynamoDB database

Python pre-trained ML model for gesture ordering

OpenAI LLM for speech ordering (NLU/NLP)

Menu information

Request info about menu

(1) User message - "Hey can I get two fries"

(2) Order Management that's taken place

(1) User intent

(2) Natural response
OpenAI Assistants

OpenAI claims that its’ assistant API can be used to “build AI assistants within your own applications”. While this is true, it is also true that it is currently not reliable by itself.

However we have found a few functionalities of this API extremely promising.

The first is function calling - which we can use to trigger action cycles based on user intent. The second is for natural response generation.

For our project we have combined these two functionalities along with our own backend for managing orders to create a reliable AI voice assistant for drive-through ordering. The accuracy and speed of this model will only improve over time, as the OpenAI API improves.
"Hey! Can I get two fries?"

User message sent to backend

Hey! Can I get two fries?

JSON containing reply and order state

Flask server

JSON returned with intent and arguments

Hey! Can I get two fries?

Message to undergo NLP passed

OpenAI LLM for speech ordering (NLU/NLP)

Text information

Menu information

Request info about menu

DynamoDB database
"Hey! Can I get two fries?

User message sent to backend

JSON containing reply and order state

"food_type": "fries",
"quantity": 2,
"size": "small"

JSON returned with intent and arguments

OpenAI LLM for speech ordering (NLU/NLP)

DynamoDB database

React

Flask server

Pauses 1.5 seconds
"Hey! Can I get two fries?"

User message sent to backend

React

JSON containing reply and order state

Flask server

JSON returned with intent and arguments

OpenAI LLM for speech ordering (NLU/NLP)

Message to undergo NLP

Query for "fries" in food_type

Returns "fries"

Menu information

Query for "fries" in menu

Request info about menu

DynamoDB database

Pauses 1.5 seconds

Adds {"food_type": "fries", "quantity": 2, "size": "small"} to order list
User message sent to backend

React

JSON containing reply and order state

Flask server

"Your order of two small fries has been added to the order. Is there anything else I can help you with?"

"Your order with intent and arguments"

Two small fries have been added to the order. Let user know. Message to undergo NLP passed.

OpenAI LLM for speech ordering (NLU/NLP)
Hey, it's Drvingo here! I am here to help with your order! Please get started with your order for today.

You: Yo yo yo what's up. Can I get 2 fries if y'all got that?

Using existing thread with ID: thread_zVifneKOWazKU1lsJUuY1gxl
Waiting...
Waiting...
Waiting...
Waiting...
Waiting...
Waiting...
Waiting...
Waiting...
Waiting...
Waiting...
Waiting...
Waiting...

Here is the state of the global order manager: {
  'friesmedium': {
    'food_type': 'fries',
    'quantity': 2,
    'food_size': 'medium'
  }
}
Waiting...
Waiting...
Waiting...
Waiting...
Waiting...
Waiting...
Waiting...

Assistant: Yo yo yo! I've added 2 medium-size fries to your order. Anything else I can get for you today?

You: ✗
It can also do that in different languages!

It works in 5 different languages - Chinese, French, Spanish, English, and German!

I'm confused about what else to order. What do you recommend?
这里是我们的菜单，你可以选择一些美味的食物

1. **全天早餐** - 包括鸡蛋、培根、制成汉堡，适合任何时间享用。
2. **动脉阻塞者汉堡** - 双层牛肉饼、培根、奶酪和特制酱料的巨型汉堡。
3. **培根汉堡** - 经典牛肉饼顶部加上脆培根、车打奶酪以及新鲜生菜和番茄。
4. **简单佳人汉堡** - 含单层牛肉饼、生菜和番茄，配上海绵酱和芥末。
5. **Heavenly Heaven** - 宵夜宵，配特调酱和新鲜香草。
6. **火锅尼乌肉** - 火锅泥丸肉，配蔬菜和特制酱料。
7. **新疆肉夹馍** - 夹馍中的肉，配特制酱料。
8. **奶酪** - 低脂奶酪，配特制酱料。

Nǐ kěyǐ xiǎnzé yīxiē kàn qǐlái yǒu rén dì měishi, huǒzhè rúguò nǐ yúyào gèng duō jiǎnyì, wǒ kěyǐ bāngzhù nǐ xuǎnzé! Xūyào bāngmáng diǎn shénme ma?

Here is our menu with some delicious food for you to choose from:

1. **All-Day Breakfast** - Includes eggs, bacon, and burgers, perfect for any time.
2. **Artery Clogger Burger** - A giant burger with a double beef patty, bacon, cheese and special sauce.
3. **Bacon Burger** - A classic beef patty...
It can also handle unrelated questions well!
Hey, it's Drivingo here! I am here to help with your order! Please get started with your order for today.

You: Tell me a food related joke

Assistant: Why don't eggs tell jokes? Because they might crack up! 😬

If there's anything else I can help with, just let me know! Maybe a menu peek or help with an order?

You: Nah I'm good. I think you're pretty cool, can we be friends instead of me having to order food?
You: As my friend can you give me some life advice? Do you think AI like you will replace humans like me from jobs like this drive through?

Assistant: I’m designed to assist and enhance experiences, not replace the unique and irreplaceable qualities that humans bring to every interaction. In roles like drive-through services, technology like AI can handle routine tasks, making processes more efficient. However, it can’t replicate human empathy, intuition, and genuine personal connections that are crucial in service industries.

As for life advice, embracing technology as a tool for growth and efficiency while nurturing the skills that are uniquely human, like creativity, emotional intelligence, and critical thinking, might be a beneficial approach. Leveraging technology rather than fearing it could open new opportunities and ways to excel in your career and personal life. Always try to stay adaptable and open to learning—it’s a valuable way forward!
OpenAI Assistant Creation

- Create an OpenAI client with your personal OpenAI account key.
- Create an assistant for your client. Once an assistant is created it can be reused.
- Create a thread for your conversation within the assistant. Manage the thread as required.
```python
OPEN_API_KEY = os.getenv("OPEN_API_KEY")
client = openai.OpenAI(api_key=OPEN_API_KEY)
ASSISTANT_ID = os.getenv("ASSISTANT_ID")
THREAD_ID = os.getenv("THREAD_ID")
MODEL = os.getenv("MODEL")

# Get or Create Assistant - part of making assistant (2)

def get_or_create_assistant():
    global ASSISTANT_ID
    if ASSISTANT_ID != "None":
        try:
            assistant = client.beta.assistants.retrieve(ASSISTANT_ID)
            print(f"Using existing assistant with ID: {ASSISTANT_ID}"
            return assistant
        except openai.error.OpenAIError:
            print("Assistant ID not found, creating a new assistant."

    assistant = client.beta.assistants.create(
        name="Driving Employee Testing!",
        instructions='''### Validate Order: ...
        tools=[{"type": "retrieval"},
        model=MODEL,
        file_ids=[file.id]
    )
    ASSISTANT_ID = assistant.id
    print(f"New assistant created with ID: {ASSISTANT_ID}. Please update your .env file accordingly."
    return assistant
```
def get_or_create_thread():
    global THREAD_ID
    # print(THREAD_ID)
    if THREAD_ID != "None":
        try:
            thread = client.beta.threads.retrieve(THREAD_ID)
            print(f"Using existing thread with ID: {THREAD_ID}"")
            return thread
        except openai.error.OpenAIError:
            print("Thread ID not found, creating a new thread."")
    thread = client.beta.threads.create()
    THREAD_ID = thread.id
    print(f"New thread created with ID: {THREAD_ID}. Please update your .env file accordingly.")
    return thread
# Assistant req processing (3)
# Runs whatever message is present in the current thread
# This is where I need to handle assistant FUNCTION HANDLING
# Then it passes the thread_id to run assistant function where the run is called
#

def run_assistant(thread_id):
    run = client.beta.threads.runs.create(
        thread_id=thread_id,
        assistant_id=ASSISTANT_ID,
    )

    while run.status != "completed":
        time.sleep(0.3)
        print("Waiting...")
        #Retrieve run and check if tool_calls are present
        run = client.beta.threads.runs.retrieve(thread_id=thread_id, run_id=run.id)
        if(run.status == "requires_action"):
            # print(f"run.required_action: {run.required_action}")
            # Assuming 'run.required_action' is the object provided
            required_action = run.required_action  # Access the required_action attribute of run
            submit_tool_outputs = required_action.submit_tool_outputs  # Access submit_tool_outputs of required_action

            #For every required action in tool_calls we process it
            for tool_call in submit_tool_outputs.tool_calls:
                # Assuming there could be multiple tool_calls but interested in the first one
                function_name = tool_call.function_name  # Access the name of the function in the first tool call
                # print("Required Action Function Name: ", function_name)
                if function_name == "add_to_orders_list":
                    tool_calls = run.required_action.submit_tool_outputs.tool_calls
Your Order:

- Large x 4 = $23.20
- Large x 2 = $9.64
- Medium x 6 = $34.32

Total: $67.16

Please select a category to order

- Burgers
- Desserts
- Drinks
- Fries

Finish Order
Future Work

- Improve speed of menu related questions (data retrieval takes longer than desired).
- Building on our successful integration of voice recognition, the next step could be to develop and implement voice response capabilities for our assistant by using a TTS engine.
- Implement TTS engine that returns speech output in a lower latency. So far our research has led us to learn about Python libraries we can use for this like Mistral [7] and Fast Whisper [8].
Lessons Learnt

- Enhanced System Architecture: Adapted pre-existing full-stack application to integrate OpenAI’s technology, ensuring smooth functionality of the new AI-driven voice component.
- Client and Team Collaboration: Kept communication lines open with the client and within our team, aligning our work with client expectations and project goals.
- Agile Approach in Problem-Solving: Utilized agile methods to swiftly respond to challenges and feedback, improving our ability to develop a user-centric voice assistant for drive-through ordering. Example - switching from Rasa AI [9] bot to using OpenAI.
Acknowledgements

We would like to thank our client and Drivingo’s CEO, Levern Currie (lcurrie@vt.edu), for this amazing opportunity to help make ordering food at restaurants more accessible to all people. We are also grateful that the existing codebase we expanded upon was written well, and was clear and simple to build upon. Working with Drivingo throughout the Spring 2024 semester was a great pleasure, and we would not have been able to learn as much as we did throughout this project without the continued feedback and support from Levern Currie.
References


