Voice-Activated KFC Order Assistant Documentation

Table of Contents

1. **Introduction**
    - Purpose
    - Scope
    - Audience
2. **API Integration**
    - Technologies Used
    - API Endpoints
    - Request and Response Formats
3. **System Architecture**
    - Overview
    - Components
4. **Language Model (LLM) Implementation**
    - Model Selection
    - Training Data
    - Preprocessing Steps
5. **User Interaction Workflow**
    - User Interface Design
    - Voice Recognition
    - Order Processing Logic
    - Error Handling
6. **Deployment**
    - Environment Setup
    - Deployment Steps
    - Testing Procedures
    - Maintenance and Updates
7. **Conclusion**
    - Summary
    - Future Enhancements

1. Introduction

Purpose

The voice-activated KFC order assistant aims to streamline the process of placing orders at KFC drive-in locations using voice commands.

Scope

This documentation covers the integration of voice recognition, natural language understanding, and API development for seamless order processing.

Audience

- Developers
- Project Managers
- Stakeholders

2. API Integration

Technologies Used

- Python
- Flask
- SpeechRecognition
- Transformers (Hugging Face)
- Pandas

API Endpoints

- `/get_menu`: Retrieves the current menu items and prices.
- `/place_order`: Accepts order requests and calculates the total cost.

Request and Response Formats

- JSON format for both requests and responses.

Example:
json
Copy code

```
{
  "order": ["family deal", "classic zinger"]
}
```

3. System Architecture

Overview

The system consists of a Flask-based API server that integrates a language model for understanding user orders and processing them against a predefined menu.

Components

- **API Server**: Handles incoming requests and communicates with the language model.
- **Language Model**: Uses Hugging Face's transformers library to process natural language and generate responses.

4. Language Model (LLM) Implementation

Model Selection

- **GPT-2**: Chosen for its ability to generate coherent text based on input prompts.

Training Data

- Utilizes pre-trained weights and fine-tuning on specific restaurant order datasets.

Preprocessing Steps

- Tokenization of input text using GPT-2 tokenizer.

5. User Interaction Workflow

User Interface Design

- Voice-driven interface with speech recognition for input.
- Audio feedback using gTTS for response output.

Voice Recognition

- Utilizes the SpeechRecognition library to convert speech to text.

Order Processing Logic

- Matches user input against the menu items to compute the total cost.
- Handles special requests and additional items.

Error Handling

- Recognizes and handles unrecognized inputs gracefully.
- Provides clear prompts for user actions.

6. Deployment

Environment Setup

- Python virtual environment with required libraries installed.
- Flask server deployed on a cloud platform (e.g., Heroku, AWS).

Deployment Steps

- Clone repository, install dependencies, and configure environment variables.
- Start Flask server (`python api_server.py`).

Testing Procedures

- Unit tests for API endpoints and language model integration.
- User acceptance testing for voice recognition accuracy and order processing.

Maintenance and Updates

- Regular updates to menu items and prices.
- Monitoring for performance and scalability.

7. Conclusion

Summary

The voice-activated KFC order assistant provides a seamless ordering experience through voice commands, leveraging natural language processing technologies.

Future Enhancements

- Implement user accounts and order history.
- Expand menu customization options.
- Enhance natural language understanding capabilities.