Zomato RAG Project Technical Report

1. System Architecture Overview

This project implements a Restaurant Information Retrieval Assistant using Retrieval-Augmented Generation (RAG) for Zomato menu data. The system consists of four main components:

Web Scraping Component

- Custom scraping modules (`menu.py` and `zommy.py`) to extract restaurant menus and listings from a city-specific URL
- Structured data collection with error handling and `robots.txt` compliance
- Data stored in a standardized format on MongoDB for efficient vector search retrieval

Knowledge Base Creation

- MongoDB vector database for storing embedded menu items
- Semantic search capabilities for natural language queries
- Contextual information preserved for detailed menu items

RAG-based Chatbot

- Query embedding using sentence transformers
- MongoDB-backed conversation history for contextual follow-ups
- HuggingFace model integration for response generation

User Interface

- Interactive chat interface for natural language restaurant queries
- Session management for persistent conversations

2. Implementation Details

Data Collection Methodology

- Target Data: Restaurant names, menu items with descriptions and prices, dietary information, average cost, ratings, etc.
- Storage Format: Structured data with appropriate indexing for efficient retrieval

Vector Database Implementation

- MongoDB with vector search capabilities
- Document structure optimized for menu information retrieval
- Embedding model: `nomic-ai/nomic-embed-text-v1`

Conversation Management

- Session-based conversation history stored in MongoDB
- Context-aware question rephrasing for follow-up queries
- Memory persistence across user sessions

Query Processing Pipeline

- User query embedding and optional rephrasing using conversation history
- Vector similarity search to retrieve relevant menu context
- Context-enhanced prompt construction
- Response generation with attribution to source information

3. Challenges and Solutions

Handling Varied Menu Formats

- **Challenge:** Handling 10k+ menu items efficiently
- **Solution:** Robust data processing pipeline with standardization techniques

Maintaining Conversation Context

- **Challenge:** Ensuring accurate multi-turn conversation
- **Solution:** MongoDB-based chat memory with session management and question rephrasing

Accurate Information Retrieval

- **Challenge:** Ensuring precision in response data
- **Solution:** Fine-tuned embedding model with optimized vector search parameters

Edge Case Handling

- **Challenge:** Responding to out-of-scope queries
- **Solution:** Graceful fallback responses

4. Future Improvements

- Enhanced scraping for broader coverage and auto-refresh
- Multi-modal support with images and visual elements
- Sophisticated comparison of restaurants and menu items
- Personalized recommendations through user preference modeling
- Optimization of retrieval and response generation speed

5. Technical Stack

- **Programming Language:** Python
- **Web Scraping:** Selenium (via custom modules)
- **Database:** MongoDB with vector search
- **Embedding Model:** `nomic-ai/nomic-embed-text-v1`
- **LLM Integration:** HuggingFace (`mistralai/Mixtral-8x7B-Instruct-v0.1`)
- **UI Framework:** Gradio

6. Installation and Usage

1. Clone the repository
2. Install dependencies: `pip install -r requirements.txt`
3. Configure MongoDB and HF connection settings in `.env`
4. Launch the application: `python app.py`

The application provides a natural language interface for querying restaurant information, with support for context-aware follow-up questions.

7. Conclusion

This project demonstrates the practical application of modern RAG techniques in the restaurant domain, significantly enhancing user interaction for menu information retrieval. The system integrates web scraping, vector search, and conversation memory to create a comprehensive and scalable assistant. It fulfills the project requirements while laying a foundation for further innovation.