Retrieval-Augmented Generation for Airbnb-style Hotel Review Chatbot

Chu-Yun Chu (cc280)
Wuchi Wang (ww58)

1 Introduction / Problem

In recent years, the hospitality industry has seen a dramatic increase in user-generated content, especially in the form of customer reviews on platforms such as Booking.com and Airbnb. While these reviews are valuable for understanding guest experiences, the sheer volume and unstructured nature of the data make it difficult for users to extract meaningful insights. Travelers with specific needs—such as finding a hotel that is both clean and quiet—must often read through dozens or even hundreds of reviews to make an informed decision.

To address this issue, our project aims to develop an intelligent hotel review question-answering system that can retrieve and summarize relevant reviews based on a user's natural language query. We explore the paradigm of **Retrieval-Augmented Generation (RAG)**, a framework that combines dense retrieval with large language models (LLMs) to generate context-aware responses. In our case, the system first retrieves the top-k most relevant hotel review documents based on the semantic similarity between the user query and hotel content, and then passes those documents into an LLM to produce fluent, personalized answers.

By targeting a real-world scenario, hotel booking decision support for Airbnb-style accommodations, this system bridges the gap between unstructured textual data and actionable insights, enabling smarter, faster, and more satisfying user experiences.

2 Prior Work

Our final system in Task 3 builds upon the foundational components developed in Task 1 and Task 2.

In **Task 1**, we implemented a large-scale web scraping pipeline that collected hotel names and English-language reviews from Booking.com. For each hotel, we collected up to 1,000 reviews, resulting in a dataset of over 200,000 raw reviews across approximately 200 hotels in New York City. Reviews were stored in CSV format and preprocessed to extract the English full review field for downstream analysis.

In **Task 2**, we transformed the corpus into a retrievable knowledge base. Using the `stella_en_400M_v5` SentenceTransformer model, we embedded each hotel document (a concatenation of its reviews) into a high-dimensional vector space. We implemented a `DenseRetrievalExactSearch` class to perform semantic search using cosine similarity.

In **Task 3**, we integrated this pipeline with a generative language model (e.g., **DeepSeek-R1-Distill-Qwen-1.5B**) and added a summarization step using `facebook/bart-large-cnn`. The chatbot interface was implemented using Gradio for interactive natural language querying.

3 Model / Algorithm / Method

The system is based on a simplified Retrieval-Augmented Generation (RAG) framework but implemented in a modular and lightweight form.

3.1 Corpus Preparation

We first preprocessed hotel reviews collected from Booking.com. For each hotel, we retained only review lines containing the word `liked`, then concatenated them into a single document per hotel. Each document is identified by a hotel name and assigned a unique docID.

3.2 Dense Retrieval via Sentence Embeddings

We used the `all-MiniLM-L6-v2` SentenceTransformer to encode each hotel document into a 384-dimensional embedding vector. The same embedding model is used to encode user queries. Cosine similarity is computed between the query and each document vector to retrieve the top-k most semantically relevant hotels. This process is implemented using the PyTorch function `torch.nn.functional.cosine_similarity`.

3.3 Prompt Construction

Instead of retrieving exact text snippets, we extract a truncated portion of each relevant hotel's reviews and format them into a manually constructed prompt. The prompt follows this general format:

```
Please briefly introduce these 3 hotels and summarize their reviews with only
description.  Max 75 words each.
```

This is followed by a block of text for each hotel, such as:

```
Hotel Name:  The Ludlow
Reviews:  Very stylish hotel in trendy East Village...
```

3.4 LLM Model Comparison

We experimented with multiple open-source LLMs on Hugging Face to select one suitable for our chatbot pipeline. The models were evaluated based on generation quality, prompt compatibility, and runtime performance.

Table 1: LLM Model Evaluation Summary

Model	Max Tokens	Output Coherence	Comments
`TinyLlama-1.1B-Chat`	2048	Low	Incomplete answers, often cut off
`microsoft/Phi-3-mini-instruct`	2048	Moderate	Token overflow errors, not stable
`DeepSeek-R1-Distill-Qwen-1.5B`	16384	High	Smooth long-form generation

The final choice, `DeepSeek-R1-Distill-Qwen-1.5B`, balanced quality and inference time well. It supported longer prompts, returned more structured answers, and was relatively lightweight compared to full-scale instruction-tuned models.

We used the `deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B` model as a generic text generator. The final prompt is passed into the model using the `generate()` API with temperature 0.7, top-p 0.95, and a repetition penalty of 1.1. No fine-tuning or additional instruction tuning is applied. The output is parsed to remove formatting artifacts (e.g., asterisks, system prompt echoes).

3.5 Chatbot Interface

We used Gradio to deploy the system via a web-based user interface. Users can type in questions (e.g., "Which hotels are quiet and clean?"), and receive friendly, summarized hotel recommendations. The chatbot avoids answering non-hotel-related queries by hardcoding scope restrictions in the interface description.

4 Results & Findings

4.1 Demonstration

We queried `"Which hotels are quiet and clean?"` and received a structured response with three matching hotels, each listed with pros, cons, and a verdict.

4.2 Latency

- Retrieval: ∼1.2s
- Summarization: ∼4s per hotel
- LLM Generation: ∼10–15s

End-to-end latency is under 20 seconds on CPU.

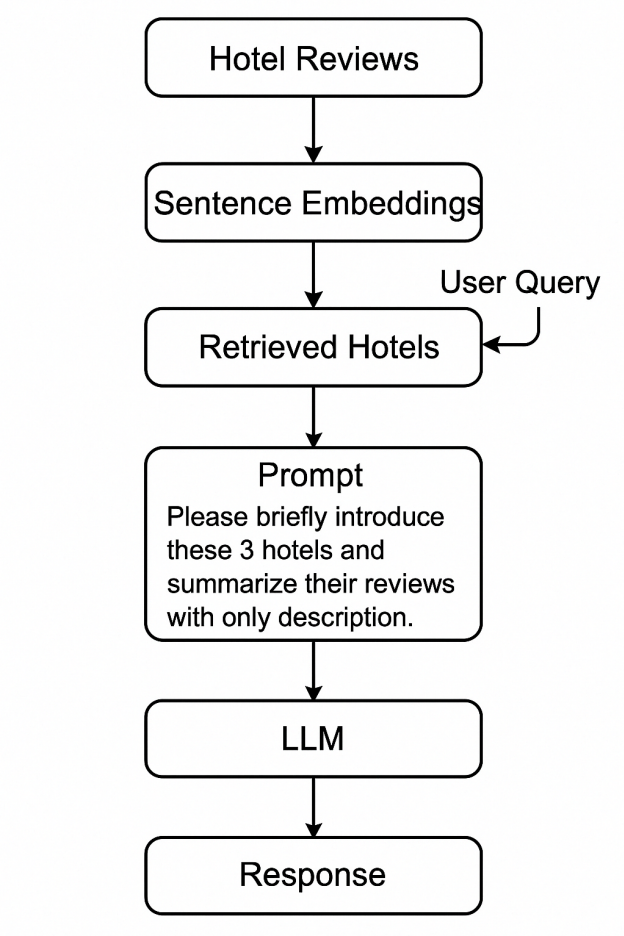

Figure 1: System Pipeline for Hotel Review Chatbot

4.3 Evaluation

Due to limited resources, we evaluated the system qualitatively. The chatbot responses were relevant, structured, and grounded in the reviews. However, small LLMs struggled with vague queries and summarization length was constrained by token limits.

4.4 Error Cases and Observations

We identified several limitations and edge cases during development:

- **Vague queries:** Inputs such as "good for families" or "fun" often led to generic or off-topic responses due to a lack of specific grounding in reviews.
- **Prompt length issues:** For hotels with longer reviews, the prompt occasionally exceeded the token limit, leading to truncated or incomplete generations.
- **Shallow reasoning:** While factual consistency was good, deeper inference (e.g., deducing "quiet" from subtle phrasing) was often missed without direct keywords.

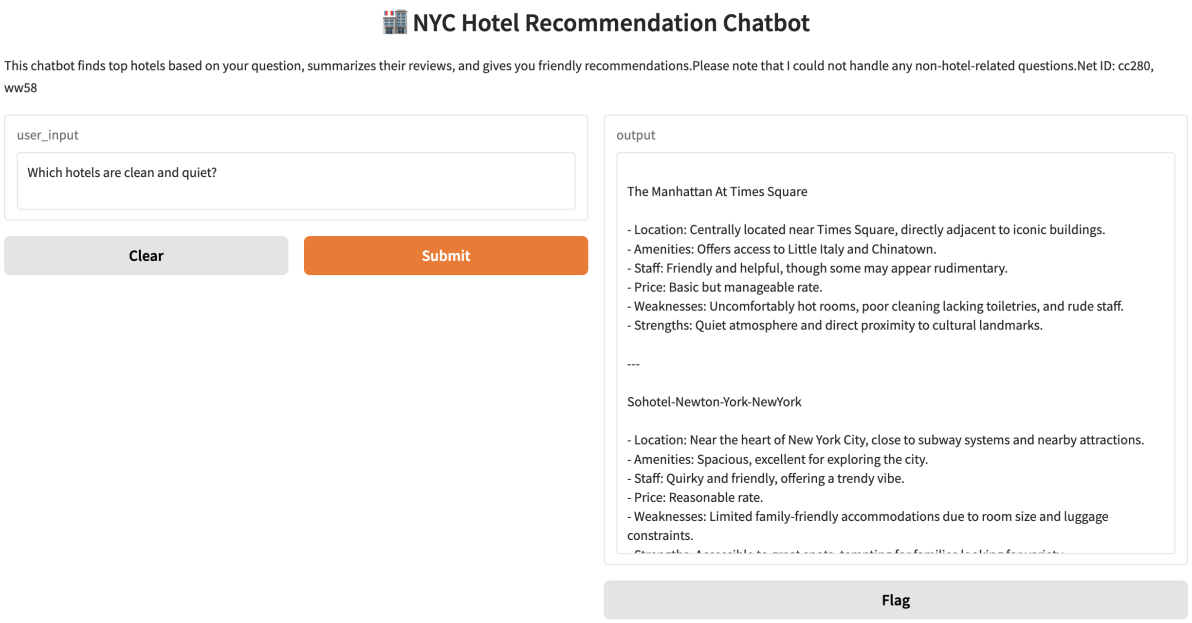

Figure 2: Sample Response Output for Hotel Review Chatbot

These highlight areas for improvement, including more robust prompt engineering and possibly integrating multi-turn clarification dialogue in future iterations.

5 Conclusion

We successfully implemented a domain-specific RAG-based chatbot system that recommends hotels based on user-defined preferences. The combination of dense retrieval, summarization, and LLM generation provides users with fast, meaningful hotel suggestions grounded in real reviews. Though limited by computational budget, the system lays a strong foundation for scalable, real-world applications in intelligent travel support.