

PERSONALIZED PROPERTY MATCHING USING RAG FOR SPECIAL NEEDS

Data Analysis and Visualization 094295

Authors:

Itay Bachar - 206948218

Sewar Hino - 323016485

Adir Toledano - 313535379

A series of five parallel blue lines of varying lengths, slanted diagonally from the bottom-left towards the top-right, located on the right side of the page.

Technion
Spring 2023-2024

Abstract:

In This project we develop a Retrieval-Augmented Generation (RAG) system to improve property recommendations for Airbnb users by leveraging synthetic question-answer (QA) pairs for evaluation. Given the lack of ground truth data in the Airbnb dataset, we generated synthetic QA pairs based on Airbnb property descriptions. This dataset enhances the reliability of the RAG system in retrieving relevant properties. Our system, demonstrated through a prototype application, enables users to query property features like accessibility, infrastructure, and amenities, providing tailored property recommendations. The results confirm the RAG system's efficacy in returning relevant listings, showing promise for application across similar recommendation domains.

Introduction:

In recent years, personalized recommendation systems have become essential for platforms like Airbnb, where users increasingly seek properties that meet specific, often unique, requirements. Beyond standard filters like location, price, and number of rooms, users frequently have preferences tied to accessibility, specific amenities, or infrastructure needs—such as wheelchair accessibility, the presence of elevators, or the availability of essential services like internet or kitchen facilities. Despite this growing demand for personalization, traditional recommendation systems struggle to adequately address these nuanced user needs, often limiting the user experience to generic listings that fail to capture detailed requirements.

This project aims to bridge this gap by developing a Retrieval-Augmented Generation (RAG) pipeline tailored to offer property recommendations that reflect diverse user preferences. The RAG system leverages a unique approach to evaluating user-specific requirements through the creation of synthetic question-answer (QA) pairs, addressing the critical challenge of a lack of annotated data to train and evaluate the model's ability to retrieve and match properties based on detailed user queries.

The project also includes a demo application where users input specific property requirements in natural language, and the RAG system recommends listings that align with these detailed queries. This user interface enables an interactive experience, allowing users to type queries like “looking for a wheelchair-accessible property with an elevator” and receive relevant recommendations, demonstrating the RAG system's practical application.

To achieve this, we focused on three main areas of user interest: accessibility features, specific infrastructure (such as elevators), and basic amenities. We used a large

language model (LLM) to generate synthetic QA pairs, which enables the RAG system to be evaluated on its ability to accurately retrieve listings based on real user-like questions. This synthetic QA dataset provides a robust foundation for fine-tuning the model and assessing its ability to deliver relevant, contextually appropriate responses.


Ultimately, this project contributes to the field of recommendation systems by expanding the capabilities of Airbnb's recommendation framework. Through a more nuanced approach and an interactive user interface, we aim to improve user satisfaction by providing recommendations that address specific needs, creating a more inclusive and user-centric platform experience. This methodology has potential applications beyond Airbnb, offering valuable insights for enhancing recommendation systems across various industries where personalized, context-aware recommendations are vital.

Methodology:

[Dataset and Preprocessing:](#)

We used an Airbnb dataset hosted on Hugging Face, containing property descriptions, amenities, and numerical attributes such as price and guest capacity. The data preprocessing focused on the following key tasks:

1. Data Cleaning: Irrelevant columns like `weekly_price` and `monthly_price` were removed to streamline the dataset.
2. Handling Missing Values: Missing numerical values, such as in `bedrooms` and `beds`, were filled with the mean value. Text fields with missing values were replaced with placeholders to avoid skewing model training.
3. Data Type Validation: Data types were standardized (e.g., ensuring `first_review` and `last_review` were in datetime format).
4. Exploratory Data Analysis (EDA): Using histograms and KDE plots, Adir conducted EDA to identify outliers and understand feature distributions. This step provided insight into data patterns, such as the distribution of `price` and `number_of_reviews`.

 Refer to Appendix for visualization samples.

[QA Dataset Generation:](#)

Since ground truth values for QA were absent, we generated a synthetic QA dataset using Cohere's large language model (LLM) API. This dataset creation process included:

1. Sample Selection: 500 Airbnb listings were sampled to capture a broad spectrum of property features.
2. Topic-Based Prompting: The QA pairs covered topics like accessibility, specific infrastructure (e.g., elevator), and basic amenities (e.g., internet, kitchen). Iterative prompts were crafted to ensure topic relevance.
3. Validation and Refinement: LLM outputs were parsed and validated for format consistency and topic alignment. Challenges with LLaMA 3 led to a transition to Cohere's LLM, which provided better control over topic specificity and output coherence.

The final QA dataset, stored in a CSV, serves as a ground truth reference, essential for evaluating the RAG model's response quality.

[RAG System Development:](#)

The RAG system consists of embedding-based retrieval and a text generation model:

1. Embedding Generation: Property descriptions were embedded using the `all-MiniLM-L6-v2` model from the SentenceTransformers library. The embeddings were indexed with FAISS for fast similarity-based retrieval.
2. Text Generation: A language model (`distilgpt2`) was used to generate responses, summarizing retrieved properties in natural language.
3. QA-Enhanced Response Generation: the model utilized the previous part's QA pairs in the response generation process, integrating relevant QA pairs to improve the response context.
4. Evaluation: The system was evaluated using precision, recall, and F1-score. Synthetic queries were matched against the ground truth QA pairs to measure response relevance.

Experiments:

[Experimental Setup:](#)

The experimental setup was designed to evaluate the RAG system's performance in retrieving and generating responses that meet specific user requirements, with a focus on accessibility and amenity preferences. Our primary goals in this evaluation were to measure retrieval accuracy, assess the relevance of generated responses, and identify areas for improvement in user-specific recommendations. The experiments were structured as follows:

1. Synthetic Queries:

To simulate realistic user interactions, we developed a series of synthetic queries that focused on distinct property requirements. For example, queries included specific phrases such as “wheelchair-accessible property with an elevator” and “quiet neighborhood with fast internet access and kitchen amenities.” These queries tested the system's ability to accurately retrieve listings that matched highly detailed and specific preferences. By structuring queries around the main topics—accessibility, infrastructure, and basic amenities—we ensured a comprehensive assessment of the system's ability to handle diverse user needs.

2. Response Generation and Re-Ranking:

After the initial retrieval of relevant properties, we utilized the text generation component to produce customized responses for each query, providing users with natural language summaries of suitable listings. Additionally, a re-ranking mechanism prioritized the retrieved listings based on user-defined preferences, such as accessibility or specific infrastructure features. This approach helped to further refine the results, allowing us to evaluate the added value of re-ranking in improving response relevance for the user.

3. Evaluation Metrics:

Quantitative metrics were essential for evaluating retrieval accuracy and the quality of generated responses. We calculated:

- Precision: The proportion of retrieved listings that were relevant to the query, reflecting the system's ability to provide accurate results.
- Recall: The proportion of relevant listings that were successfully retrieved, assessing the system's comprehensiveness in meeting user needs.

- F1-Score: A balanced metric that considers both precision and recall, giving a single performance measure that captures retrieval quality.

Alongside these metrics, qualitative evaluations were conducted by analyzing the relevance and clarity of the generated responses. Each response was reviewed to ensure it accurately summarized key property features based on the user's requirements. This combined approach of quantitative metrics and qualitative assessment provided a thorough evaluation of the system's effectiveness.

4. Synthetic QA Dataset as Ground Truth:

The synthetic QA dataset generated by Itay's Cohere model served as ground truth, allowing us to establish labeled data against which the RAG system's output could be evaluated. For each query, we compared the retrieved results and generated responses with the synthetic QA pairs, verifying that the RAG system could match user queries with appropriate answers drawn from actual property descriptions. This synthetic dataset enabled us to measure the system's accuracy in a controlled environment, which would otherwise be challenging due to the lack of annotated data for real-world user queries.

5. User Interaction Simulations via Demo Application:

To assess the system's real-world utility, we conducted simulations through the demo application interface. Users entered queries in natural language, and the RAG system provided ranked recommendations based on the FAISS retrieval and re-ranking processes, followed by the generation of descriptive responses. This simulation helped us observe the user experience firsthand, ensuring that the RAG system could respond effectively to complex, nuanced requests. Insights from these simulations informed potential improvements, such as expanding the QA dataset to cover additional user scenarios.

Summary of Experimental Results:

Our experimental results demonstrated the RAG system's capacity to meet specific user requirements, particularly for accessibility-focused and infrastructure-specific queries. Precision and recall scores highlighted the system's strength in accurately retrieving relevant properties, while qualitative assessments confirmed the coherence and relevance of generated responses. However, some limitations were identified, particularly in handling rare or unique property features. Future work will focus on expanding the QA dataset and refining the retrieval model to improve response accuracy for these cases.

Results:

The RAG system demonstrated high retrieval accuracy, achieving a precision of 1.00, recall of 0.67, and F1-score of 0.80, indicating effective retrieval of relevant properties. Additionally, responses were contextually aligned with user needs, especially for accessibility-related queries.

Discussion:

Our project demonstrates the potential of a Retrieval-Augmented Generation (RAG) system for improving personalized property recommendations in the Airbnb domain. By integrating a synthetic QA dataset as ground truth, we successfully addressed gaps in traditional recommendation systems, enabling responses tailored to user-specific needs such as accessibility and amenities. The Cohere-based QA generation enhanced the quality of ground truth, and the FAISS-based retrieval proved efficient for handling large datasets, contributing to the overall relevance and scalability of our approach.

One of the project's unique aspects is the development of a demo application interface, which allows users to type in specific property requirements and receive tailored recommendations based on the RAG system's output. This interactive application highlights the RAG system's practical utility, offering users an intuitive way to find listings that align with specific, detailed preferences.

Key insights from this project include the value of generating synthetic QA pairs, which allowed us to address the absence of labeled data and provide an evaluation framework tailored to real-world user queries. However, we encountered limitations related to API rate limits and cost constraints with the Cohere model, which restricted the volume of QA pairs generated. Addressing these challenges in future work could include exploring self-hosted LLMs to minimize API-related constraints or fine-tuning the retrieval model with domain-specific embeddings to enhance relevance further.

Potential improvements for future work involve expanding the synthetic QA dataset to cover a wider range of user scenarios, thus increasing the system's robustness and ability to respond to complex, nuanced queries. Additionally, embedding more specific domain language into the retrieval model could enhance the RAG system's alignment with user expectations, making it a more powerful tool for personalized recommendations in both the hospitality sector and other domains.

Appendix:

GitHub Repository: <https://github.com/ItayBachar1/DataAnalysisLab.git>

1. ****Visualizations****: EDA plots (histograms, KDE, scatter plots) of features like `price`, `bedrooms`, and `number_of_reviews`.

	_id	minimum_nights	maximum_nights	\
count	5.555000e+03	5555.000000	5.555000e+03	
mean	1.664386e+07	5.564356	1.382776e+06	
min	1.170800e+04	1.000000	1.000000e+00	
25%	8.376039e+06	1.000000	5.900000e+01	
50%	1.711345e+07	2.000000	1.125000e+03	
75%	2.461229e+07	3.000000	1.125000e+03	
max	3.295874e+07	1250.000000	2.147484e+09	
std	9.622558e+06	22.613861	5.256920e+07	

	last_scraped	calendar_last_scraped	\
count	5555	5555	
mean	2019-03-01 12:48:43.420342016	2019-03-01 12:48:43.420342016	
min	2019-02-11 05:00:00	2019-02-11 05:00:00	
25%	2019-02-18 05:00:00	2019-02-18 05:00:00	
50%	2019-03-07 05:00:00	2019-03-07 05:00:00	
75%	2019-03-08 05:00:00	2019-03-08 05:00:00	
max	2019-03-11 04:00:00	2019-03-11 04:00:00	
std	NaN	NaN	

	first_review	last_review	\
count	4167	4167	
mean	2016-11-27 16:08:28.855291648	2018-08-15 20:18:09.416846592	
min	2009-10-27 04:00:00	2012-01-06 05:00:00	
25%	2015-12-08 17:00:00	2018-08-08 04:00:00	
50%	2017-04-02 04:00:00	2019-01-02 05:00:00	
75%	2018-04-24 04:00:00	2019-02-15 05:00:00	
max	2019-03-10 05:00:00	2019-03-11 04:00:00	
std	NaN	NaN	

	accommodates	bedrooms	beds	number_of_reviews	bathrooms	\
count	5555.000000	5555.000000	5555.000000	5555.000000	5545.000000	
mean	3.505851	1.411712	2.071454	27.606481	1.291163	
min	1.000000	0.000000	0.000000	0.000000	0.000000	
25%	2.000000	1.000000	1.000000	1.000000	1.000000	
50%	3.000000	1.000000	2.000000	5.000000	1.000000	
75%	4.000000	2.000000	3.000000	32.000000	1.000000	
max	16.000000	20.000000	25.000000	533.000000	16.000000	
std	2.297819	1.041473	1.617764	49.798376	0.702265	

	price	security_deposit	cleaning_fee	extra_people	\
count	5555.000000	5555.000000	5555.000000	5555.000000	
mean	278.766157	509.430424	94.074801	22.791899	
min	9.000000	0.000000	0.000000	0.000000	
25%	70.000000	100.000000	35.000000	0.000000	
50%	129.000000	500.000000	94.074801	0.000000	
75%	280.000000	509.430424	100.000000	20.000000	
max	48842.000000	39228.000000	2000.000000	2346.000000	
std	842.215531	1260.577280	93.792601	69.331511	

	guests_included	\
count	5555.000000	
mean	1.747435	
min	1.000000	
25%	1.000000	
50%	1.000000	
75%	2.000000	
max	16.000000	

2. **References**: Key papers, blogs, and resources that informed each stage of the project.