# Technical Report: Final Project DS 5110: Introduction to Data Management and Processing

Team Member 1: Yaksh Shah(002310883)[1]
Team Member 2: Smit Chandi (002320719)[2]
Khoury College of Computer Sciences
Data Science Program
`shah.yak@northeastern.edu`[1]
`chandi.s@northeastern.edu`[2]

December 10, 2024

## Contents

## 1 Introduction

The real estate industry has undergone significant advancements with the integration of artificial intelligence (AI) and natural language processing (NLP) into property search systems. Traditional keyword-based search engines often struggle to understand user intent, limiting their effectiveness for descriptive or nuanced queries. Inspired by Zillow's NLP-powered search engine, which allows users to search properties using natural language, this project aims to create an intelligent recommendation system for Airbnb listings.

This system addresses a critical gap in property search by enabling users to input detailed and descriptive queries, such as *"a family-friendly house near the beach with a backyard."* Unlike traditional systems that rely solely on matching keywords, our approach focuses on understanding the broader context of user input to retrieve relevant and meaningful results. By combining the strengths of multiple search techniques, the system ensures accuracy, relevance, and flexibility in property recommendations, catering to a wide range of user needs.

In addition to the property recommendation engine, the project includes a dynamic and interactive Power BI dashboard. This dashboard empowers users to explore property information visually, allowing them to filter and analyze listings based on city and neighborhood selections. By presenting data in an intuitive and interactive manner, the dashboard enhances decision-making by providing clear insights into property distribution and availability.

Overall, this project represents a user-centric approach to modern property search systems. It bridges the gap between traditional keyword-based searches and the need for a more intuitive, descriptive search experience. By integrating advanced search capabilities with interactive visualization tools, the system not only meets user expectations but also sets a new standard for property recommendation platforms.

## 2 Literature Review

One notable example of existing work is Zillow's optimized search system, which employs a hybrid search technique to streamline property searches. Unlike traditional filtering methods where users apply filters sequentially (e.g., location, price, size), Zillow's approach allows users to search directly using natural language queries. This system leverages a combination of natural language processing (NLP) and machine learning algorithms to interpret user input and deliver relevant property results instantly. By bypassing the need for step-by-step filtering, this technique enhances user experience by making the search process faster and more intuitive.

### 2.1 Search Methodologies

In information retrieval, combining syntactic and semantic search techniques enhances the accuracy and relevance of search results. A hybrid search approach integrates traditional keyword-based methods with advanced semantic understanding, leveraging both the precision of syntactic analysis and the contextual depth of semantic interpretation.

### 2.1.1 Syntactic Search with BM25

The BM25 (Best Matching 25) algorithm is a widely used ranking function in information retrieval systems. It evaluates the relevance of documents based on the frequency of query terms within them, adjusted for document length and term frequency saturation. The BM25 score for a document is calculated as:

$$\text{score}(D, Q) = \sum_{i=1}^{n} \text{IDF}(q_i) \cdot \frac{f(q_i, D) \cdot (k_1 + 1)}{f(q_i, D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{\text{avgdl}})}$$

Where:

- $f(q_i, D)$ is the frequency of term $q_i$ in document $D$,
- $|D|$ is the length of document $D$,
- avgdl is the average document length in the collection,
- $k_1$ and $b$ are free parameters, typically set to 1.2 and 0.75, respectively.

BM25 effectively ranks documents by considering term frequency and document length, providing a robust foundation for syntactic search.

### 2.1.2 Semantic Search with Sentence-Transformers

To capture the contextual meaning of queries and documents, semantic search models like Sentence-Transformers are employed. These models map sentences and paragraphs into dense vector spaces, enabling the measurement of semantic similarity. For instance, the `multi-qa-mpnet-base-dot-v1` model encodes text into 768-dimensional vectors, facilitating efficient semantic search.

By leveraging both approaches, hybrid search systems offer more comprehensive and relevant search results, accommodating a wide range of user queries. This combination is particularly relevant for our project, as it allows us to optimize the search mechanism within the Airbnb Recommendation System to deliver more precise and contextually relevant property suggestions.

# 3 Methodology

## 3.1 Data Collection

We collected property data for nearly 69,000 Airbnb listings belonging to four major cities: Los Angeles, San Francisco, Oakland, and San Diego. The data was obtained from a third-party website. The dataset includes a comprehensive set of features for each listing, covering various aspects such as property details, host information, pricing, availability, and review scores.

### 3.2 Data Preprocessing

The data preprocessing steps performed on the Airbnb listing data include:

1. Dropping a set of predetermined columns that were deemed unnecessary for the analysis various host-related information.
2. Filling missing values in several columns:
   - Filling missing values in features like host_listings_count, no_of_bedrooms, and beds with 0.
   - Filling missing values in 'name', 'description', 'neighborhood_overview', 'host_name', and 'amenities' columns with an empty string.
   - Filling remaining NaN values using forward-fill.
3. Converting the 'amenities' column to a list data structure, handling cases where the string representation could not be parsed.
4. Filling missing values in the review score and reviews per month columns with the mean values.
5. Converting the 'host_listings_count', 'bedrooms', and 'beds' columns to integers.
6. Mapping the 'instant_bookable' column to boolean values.
7. Rounding the review score and reviews per month columns to 2 decimal places.
8. Cleaning the text in various columns by removing HTML tags and non-alphanumeric characters.
9. Adding a new 'City' column that assigns the corresponding city (Oakland, San Diego, San Francisco, or Los Angeles) based on the 'neighbourhood_cleansed' column.

### 3.3 Airbnb Property Analysis Dashboard

This Power BI dashboard provides a detailed analysis of Airbnb property data through various interactive visualizations. Below is a description of each visualization and its purpose:

- **Drill-Down Pie Chart: Distribution of Property and Room Types** This chart shows the proportion of different room types, such as entire homes, private rooms, shared rooms, and hotel rooms, to understand how accommodations are distributed across the neighborhood.
- **Bar Chart: Top 10 Expensive Neighborhoods** The bar chart ranks neighborhoods by average pricing, highlighting the most expensive areas for Airbnb accommodations and offering insights into premium property locations.

- **Scatter Plot: Relationship Between Pricing, Guest Ratings, and Accommodation Size** This plot visualizes the correlation between property pricing, guest ratings, and accommodation capacity. Each bubble represents a property, with its size indicating guest capacity, allowing for analysis of pricing trends and guest satisfaction.
- **Column Chart: Property Availability by Room Type** This chart displays the availability of properties for each room type, categorizing listings as available or unavailable. It provides an overview of supply across different room types.
- **Map: Geographic Distribution of Properties** An interactive map shows Airbnb properties based on city and neighborhood, allowing users to explore property locations geographically. This map helps analyze regional patterns and filter listings dynamically by specific areas.
- **Key Metrics Cards** The cards highlight important statistics, including average pricing and reviews per month, summarizing the overall trends for properties present in the neighborhood.
- **Interactive Filters** Dropdown filters for city and neighborhood enable users to refine the dashboard's content dynamically, allowing for detailed exploration of Airbnb listings based on location.

## 3.4 Airbnb Property Search Using Hybrid Search

### 3.4.1 Syntactic Search

Syntactic search, also known as lexical or keyword-based search, retrieves information by matching query terms directly with the terms in documents. This method does not consider the meaning or context of the words but emphasizes surface-level textual matching.

**BM25 Algorithm:** BM25 (Best Match 25) is a widely used ranking algorithm in syntactic search that enhances traditional term-frequency methods by considering the term saturation and document length. It assigns scores to documents based on:

- **Term Frequency:** Frequency of the query term in the document.
- **Inverse Document Frequency:** Importance of the term across the entire dataset (penalizing common terms).
- **Document Length Normalization:** Adjusts scores for varying document lengths.

**Advantages of BM25:**

- High precision for specific keyword-based queries.
- Efficient for structured data with clear query terms.

**Limitations:**

- Does not understand synonyms or user intent.
- Fails to capture relationships between terms and broader contextual meanings.

### 3.4.2 Semantic Search

Semantic search addresses the limitations of syntactic search by focusing on the meaning and intent behind a query. It uses natural language processing (NLP) and machine learning techniques to interpret the query and retrieve results that align conceptually rather than just lexically.

**Sentence-Transformers/multi-qa-mpnet-base-dot-v1 Model:** This state-of-the-art transformer-based model is fine-tuned for question-answering tasks, enabling it to encode text (queries and documents) into dense vector representations that capture their semantic meaning.

### 3.4.3 How Semantic Search Works:

1. **Text Embeddings:** Queries and documents are converted into vector representations using pre-trained models like `multi-qa-mpnet-base-dot-v1`.
2. **Similarity Measurement:** Cosine similarity measures the closeness between the query and document vectors.
3. **Retrieval:** Documents with the highest similarity scores are retrieved as results.

**Advantages of Semantic Search:**

- Captures synonyms, paraphrases, and context.
- Aligns well with broad or intent-based queries.

**Limitations:**

- May overlook exact keyword matches when relying solely on embeddings.
- Requires significant computational resources and specialized models.

### 3.4.4 Hybrid Search

Hybrid search combines the strengths of both syntactic and semantic approaches to overcome their individual limitations. It integrates lexical precision with semantic understanding to generate comprehensive search results.

### 3.4.5 How Hybrid Search Works:

1. **Query Execution:**
   - The query is processed using both syntactic (BM25) and semantic (`multi-qa-mpnet-base-dc` methods.
   - **BM25:** Ensures that documents with exact term matches are retrieved.
   - **Semantic Search:** Retrieves documents that are conceptually aligned with the query, even if the terms differ.

2. **Result Aggregation:**
    - Outputs from both methods are combined using techniques like ranking fusion or weighted averaging.
    - Documents scoring well in both syntactic and semantic searches are prioritized.
3. **Final Ranking:** A scoring system is applied to re-rank the results. This ensures that the output reflects both exact matches (for precision) and semantic matches (for recall).
4. **Output:** A unified list of results is presented to the user, balancing lexical relevance with contextual alignment.

**Advantages of Hybrid Search:**

- **Precision and Recall:** Provides exact matches for specific queries while capturing the broader intent.
- **Comprehensive Coverage:** Handles a wide range of user queries, from direct keywords to descriptive, intent-based inputs.
- **Improved User Experience:** Ensures relevant results even for complex or vague queries.

## 3.5 Application in the Project

In this project, the hybrid search methodology enhances the retrieval of Airbnb property listings based on user queries.

- **BM25 (Syntactic Component):** Prioritizes exact matches for critical terms like location, amenities, or property type.
- **Sentence-Transformers/multi-qa-mpnet-base-dot-v1 (Semantic Component):** Handles vague or descriptive user inputs, such as "a cozy family-friendly apartment near the beach."
- **Result Combination:** Outputs from BM25 and semantic search are aggregated and re-ranked, ensuring both precision and contextual relevance.

This hybrid approach enables a robust, flexible, and efficient recommendation system, aligning property listings with user preferences and descriptions effectively.

## 4 Results

Check Out the Demo Of Our Application
Check Out our Our Power BI Dashboard

## 5 Conclusion

The project demonstrates the transformative potential of artificial intelligence and natural language processing in the real estate industry, particularly for enhancing the property search experience on platforms like Airbnb. By combining hybrid search techniques, a user-friendly chatbot interface, and robust data analytics, the system simplifies property discovery, saving users valuable time and effort.

This innovative approach bridges the gap between user intent and search results, delivering tailored recommendations that align with individual preferences. Furthermore, the integration of real-time query refinement ensures adaptability, making the system highly dynamic and responsive.The project sets a benchmark for AI-driven solutions in property search, paving the way for scalable, intuitive, and impactful applications in the travel and hospitality sectors.

## References

[1] https://zillow.mediaroom.com/2023-01-26-Zillows-new-AI-powered-natural-language-search-is-a-first-in-real-estate

[2] https://insideairbnb.com

[3] Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. Found. Trends Inf. Retr. 3, 4 (April 2009), 333–389. https://doi.org/10.1561/150000001

[4] multi-qa-mpnet-base-dot-v1 model for Semantic Search

[5] Understanding Hybrid Search

[6] Hybrid Search Using Langchain and Pinecone Video

[7] Exploring AirBnB Listings with Semantic Search: A Qdrant and LLM Powered Approach

[8] Create an Amazing Power BI Dashboard in 17 minutes New York City Airbnb dataset

## A Appendix A: Code

Code for our chatbot:

```python
import os
import streamlit as st
import pandas as pd
from PropertySearch import search
from langchain_core.messages import AIMessage, HumanMessage

parent_dir = os.path.dirname(os.path.abspath('Airbnb_Listings_Data.csv'))
path_of_file = os.path.join(parent_dir, 'Airbnb_Listings_Data.csv')

property_listings_df = pd.read_csv(path_of_file,low_memory=False)


def display_results(properties):
    """Display recommended properties with details."""
    for recommended_property in properties:
        property_id = recommended_property.metadata['property_id']
        image_url = property_listings_df[property_listings_df['id']==property_id]['picture_url'].values[0]
        st.image(image_url, use_column_width=True)
        st.markdown(recommended_property.metadata["property_url"])
        st.markdown(f"**Description:** <br>{recommended_property.page_content}", unsafe_allow_html=True)
        st.markdown(f"**Accomodates:** {recommended_property.metadata['accommodates']}")
        st.markdown(f"**City:** {recommended_property.metadata['City_name']}")
        st.markdown(f"**Neighbourhood:** {recommended_property.metadata['neighborhood_name']}")
        st.markdown(f"**Amenities:** <br>{', '.join(recommended_property.metadata['imp_amenities'])}", unsafe_allow_html=True)
        st.markdown(f"**Price:** {recommended_property.metadata['price']}")
        st.markdown("---")  # Divider for visual clarity


if "chat_history" not in st.session_state:
    st.session_state.chat_history = [
        AIMessage(content="Hello, I am an Airbnb Property Search bot. How can I help you?"),
    ]
    st.session_state.property_metadata = {}
    st.session_state.prev_description = ""
    st.session_state.results = []

# Streamlit app configuration
st.set_page_config(page_title='Airbnb Property Search Bot', page_icon=' ')
st.title(':blue[Airbnb Property Search Bot]       ')
st.subheader('Your :green[AI Assistant] for finding ideal properties', divider='rainbow')
st.caption("Note: This bot can make mistakes. Please verify important information.")
```

```python

# Display chat history

for message in st.session_state.chat_history:
    if isinstance(message, HumanMessage):
        with st.chat_message("Human"):
            st.markdown(message.content)
    else:
        with st.chat_message("AI"):
            try:
                st.markdown(message.content)
                display_results(message.metadata['properties'])
            except:
                continue


# Accept user input
user_input = st.chat_input("Write a short description of what you are looking for in an Airbnb property.")

if user_input and user_input.strip():
    # Add user input to chat history
    st.session_state.chat_history.append(HumanMessage(user_input))
    with st.chat_message("Human"):
        st.markdown(user_input)

    try:
        # Determine if it's an initial query or a refined query
        if len(st.session_state.chat_history) == 1:  # Initial query
            print(user_input)
            st.session_state.property_metadata = search.call_gemini_api(user_input)
            docs = search.search_similar_properties(user_input,st.session_state.property_metadata)
            st.session_state.prev_description = user_input
        else:  # Refined query
            query_dictionary = {
                "prev_query": st.session_state.prev_description,
                "previous_metadata": st.session_state.property_metadata,
                "current_query": user_input
            }
            print(query_dictionary)
            query_prompt = search.generate_refined_prompt(query_dictionary)
            refined_data = search.call_gemini_api(query_prompt)
            print(refined_data)
            st.session_state.property_metadata = refined_data.get("updated_metadata", {})
            refined_query = refined_data.get("refined_query", user_input)
            docs = search.search_similar_properties(refined_query,st.session_state.property_metadata)
            st.session_state.prev_description = refined_query

        if docs:
            # Save results to session state
            st.session_state.results.append(docs)

            # Display the current results
            with st.chat_message("AI"):
                st.markdown("Here are the recommended properties based on your input:")
                display_results(docs)
            st.session_state.chat_history.append(
                AIMessage(content="Here are the recommended properties based on your input:",
                          metadata={"properties": docs}))
        else:
            with st.chat_message("AI"):
                st.markdown("No matching properties found. Please refine your query.")
            st.session_state.chat_history.append(
                AIMessage(content="No matching properties found. Please refine your query."))
    except Exception as e:
        st.error(f"An error occurred: {str(e)}")
```

Listing 1: Example Python Code