Master 2: Computer Science

Specialization: Data Engineering and Web Technologies

Titled :

Hotel Recommendation System

- ***Realized by :***

FERADI MAROUANE	***FERADI ISSAM***	***SEDJAL MOHAMED AYMEN***

- ***Module coordinator :*** *Dr. F.Harrag*

Academic Year: 2024/2025

Thanks

First and foremost, we would like to express our gratitude to God for providing us with strength and guidance throughout the completion of this report.

We would also like to extend our sincere thanks to **:** Dr. F.Harrag, our report supervisor, for his valuable guidance, support, and availability throughout this work. His expertise and encouragement were instrumental in the successful completion of this report.

We dedicate this work to our promotion, specializing in IDTW for the collective effort, camaraderie, and shared knowledge that have enriched our academic experience. The collaboration and support from our peers have been invaluable in achieving this milestone.

By the members of the Group

CHAPTER 1

1 Introduction

1.1 Project Objective

The objective of this project is to develop an information retrieval system with the ability to perform various types of searches, including simple searches, exact proximity searches, combined searches with operators, and exact phrase searches. The system aims to efficiently retrieve documents based on user queries and improve search accuracy using tokenization, stemming, and proximity search algorithms.

1.2 Technologies Used

This project leverages several key technologies and libraries, including Python for the implementation, NLTK for natural language processing tasks such as tokenization and stemming, and custom algorithms for proximity and phrase search. Additionally, we utilized data structures like inverted indexes for efficient document retrieval and search operations

2 Preprocessing

The hotel data (KSA_DATA4.txt) is assumed to be in a text file, where each line represents a hotel with its ID, name, city, room type, and potentially other relevant information.

2.1 Tokenization:

Tokenization is the process of splitting a text into individual words or meaningful units, called tokens. In this project, we used the word_tokenize function from the NLTK library to efficiently split text strings into a list of words. This method ensures that each word in a given document is treated as a distinct token for further processing. By tokenizing the text, we can focus on analyzing individual words or terms, which is essential for subsequent stages of the information retrieval process.

2.2 Stemming:

Stemming is the technique of reducing words to their root or base form, allowing us to treat different forms of the same word as equivalent. For example, "running," "runner," and "runs" would all be reduced to "run." To achieve this, we used the PorterStemmer from NLTK, which is a widely-used algorithm for stemming words in English. The PorterStemmer efficiently reduces words to their root form, helping to improve the accuracy of document retrieval by grouping related terms together.

2.3 Stop Words Handling:

Stop words like "the," "and," or "is" are removed as they don't add meaningful information for search. Using NLTK's predefined stop words list, we filtered these out, improving the search system's accuracy by focusing on relevant terms.

2.4 Creating and Structuring the Index:

In this project, we created the **inverted index** by processing each term from the preprocessed text (after tokenization and stemming). For each term, we recorded the documents in which it appears and the specific positions of the term within those documents. The index is structured as a dictionary, where each key is a term, and the value is another dictionary containing document IDs and the positions of each term within those documents.

3 Search Operations

3.1 Index Loading from File:

The **load_index_from_file** function reads an index file and structures it for efficient lookups. Each term in the index is associated with a dictionary containing document IDs and the positions of the term within each document, enabling fast searches.

3.2 Boolean Search

Boolean search operations allow for flexible queries using logical operators:

- **AND Search**: Retrieves documents where all specified terms appear.
- **OR Search**: Returns documents where at least one of the specified terms appears.

Examples of Queries :

- Naseem AND gosaibi
- gosaibi OR Umm

3.3 Proximity Search

Proximity search allows for searches where terms must appear near each other:

- **Exact Proximity Search**: Retrieves documents where terms appear within a specific distance from each other.
  Example: "Naseem gosaibi" with a proximity distance of 1 will return documents where "Naseem" and "gosaibi" are adjacent.
- **Phrase Search**: Searches for an exact sequence of terms in the given order, ensuring that the terms appear together exactly as specified.

4 Main Program

4.1 Loading the Index:

The program starts by loading a positional index from a text file. This index contains information about the terms present in the documents and their positions within each document. This allows for efficient term-based searching.

4.2 Processing Queries:

The query file is loaded, and for each query, documents are ranked based on their relevance. The ranking is done using a search algorithm (e.g., TF-IDF or another weighting model).

4.3 Displaying Results:

Once the documents are ranked, the results are written to an output file, with each line containing the query ID, document ID, and relevance score. The top 5 results for each query are displayed along with their associated content, providing the user with a preview of the most relevant documents.

4.4 User Interaction

After processing all the queries, the program prompts the user to input a query manually and then displays the results for that query, including the most relevant documents.

CHAPTER 2

2.1 Introduction

Hotel Recommendation System leverages a combination of text analysis techniques, including TF-IDF vectorization and cosine similarity, to identify hotels similar to a user's input. This approach aims to provide personalized recommendations based on the characteristics of the input hotel, such as its name, city, star rating, and textual descriptions.

2.2 Methodology

1. **Data Collection and Preparation:**
   - A dataset of hotels (saudi_hotels.csv), including relevant information such as hotel name, city, star rating, room types, and textual descriptions (e.g., reviews, amenities).
   - The dataset was cleaned and preprocessed to remove noise and improve data quality. This included handling missing values, converting data types, and removing irrelevant characters.
2. **Feature Engineering:**
   - A new feature, "new_features," was created by concatenating relevant columns (City, Star_Rating, Type_of_room, Review_title).
   - Text preprocessing was applied to the "new_features" column, including tokenization, stop word removal, and stemming. This step aimed to extract meaningful keywords and improve the accuracy of the similarity calculations.
3. **TF-IDF Vectorization:**
   - The preprocessed text data was transformed into a matrix of TF-IDF values using the TfidfVectorizer from scikit-learn.
   - TF-IDF (Term Frequency-Inverse Document Frequency) weights terms based on their importance within a document and across the entire dataset, providing a more informative representation of the text.
4. **Cosine Similarity Calculation:**
   - The cosine similarity between the TF-IDF vectors of each hotel pair was calculated using the cosine_similarity function from scikit-learn.
   - Cosine similarity measures the cosine of the angle between two vectors, [1] providing a measure of their similarity.
5. **Recommendation Generation:**
   - A function was developed to generate hotel recommendations based on the cosine similarity matrix.
   - Given an input hotel, the function identifies the most similar hotels by selecting those with the highest cosine similarity scores.
   - The function returns a list of recommended hotels with their names, cities, star ratings, and similarity scores.

2.3 Results and Discussion

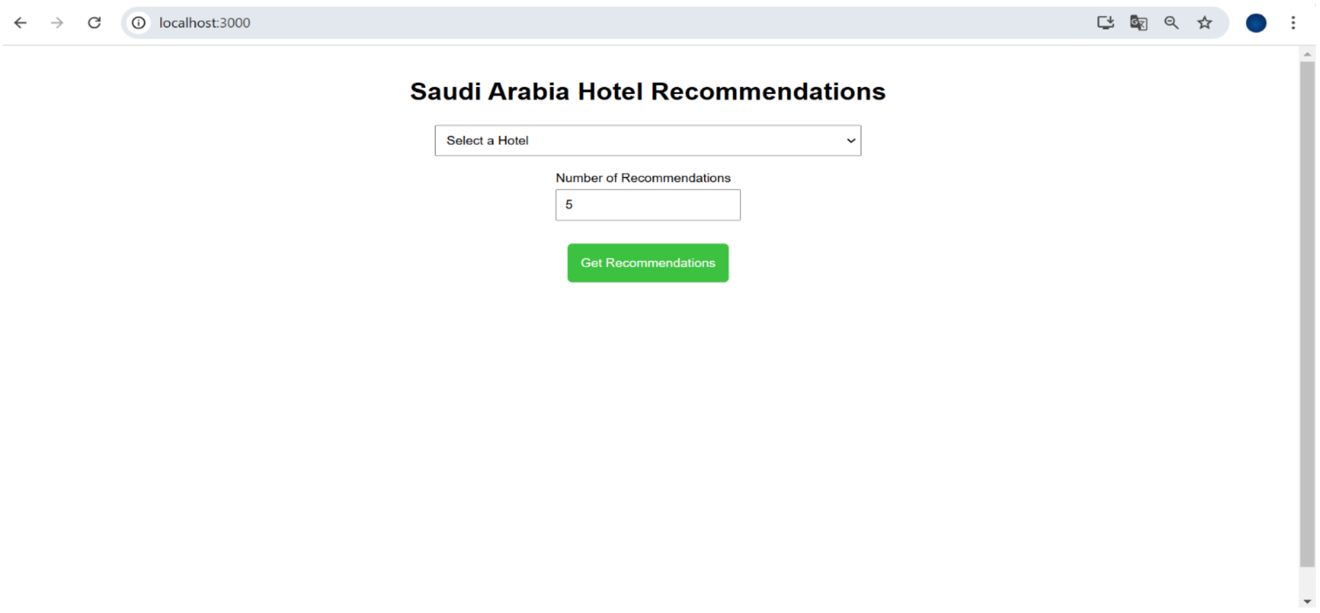
The developed recommendation system effectively identifies hotels similar to a given input. The use of TF-IDF and cosine similarity provides a robust approach to capturing the semantic relationships between hotels based on their textual descriptions and other relevant features.

The system can be further enhanced by:

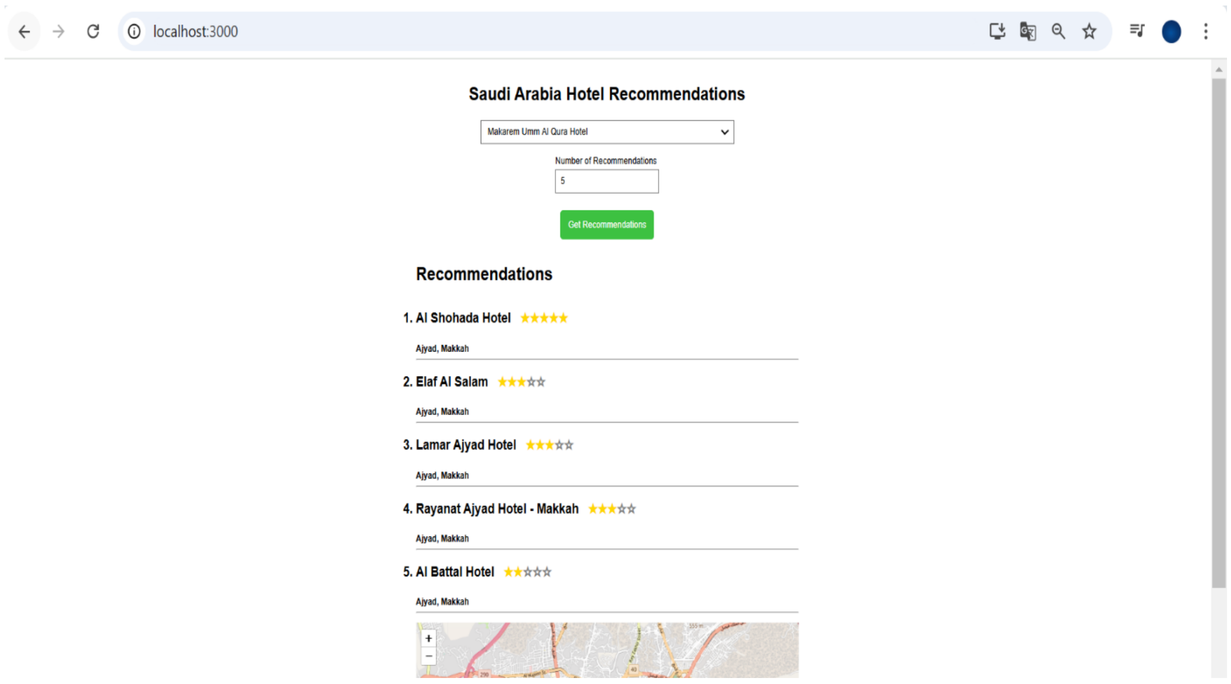
- **Incorporating user preferences:** Considering user-specific preferences, such as budget, travel dates, and preferred amenities.
- **Utilizing collaborative filtering:** Leveraging user ratings and reviews to identify similar users and recommend hotels they have enjoyed.
- **Integrating external data sources:** Incorporating data from external sources, such as review platforms and travel agencies, to enrich the recommendations.
- **Developing a user-friendly interface:** Creating an interactive interface for users to easily input their preferences and view recommendations.

2.4 application interface

Interface 1 :



Interface 2 :



Saudi Arabia Hotel Recommendations

Makarem Umm Al Qura Hotel

Number of Recommendations

5

Get Recommendations

Recommendations

1. Al Shohada Hotel ★★★★★

Ajyad, Makkah

2. Elaf Al Salam ★★★☆☆

Ajyad, Makkah

3. Lamar Ajyad Hotel ★★★☆☆

Ajyad, Makkah

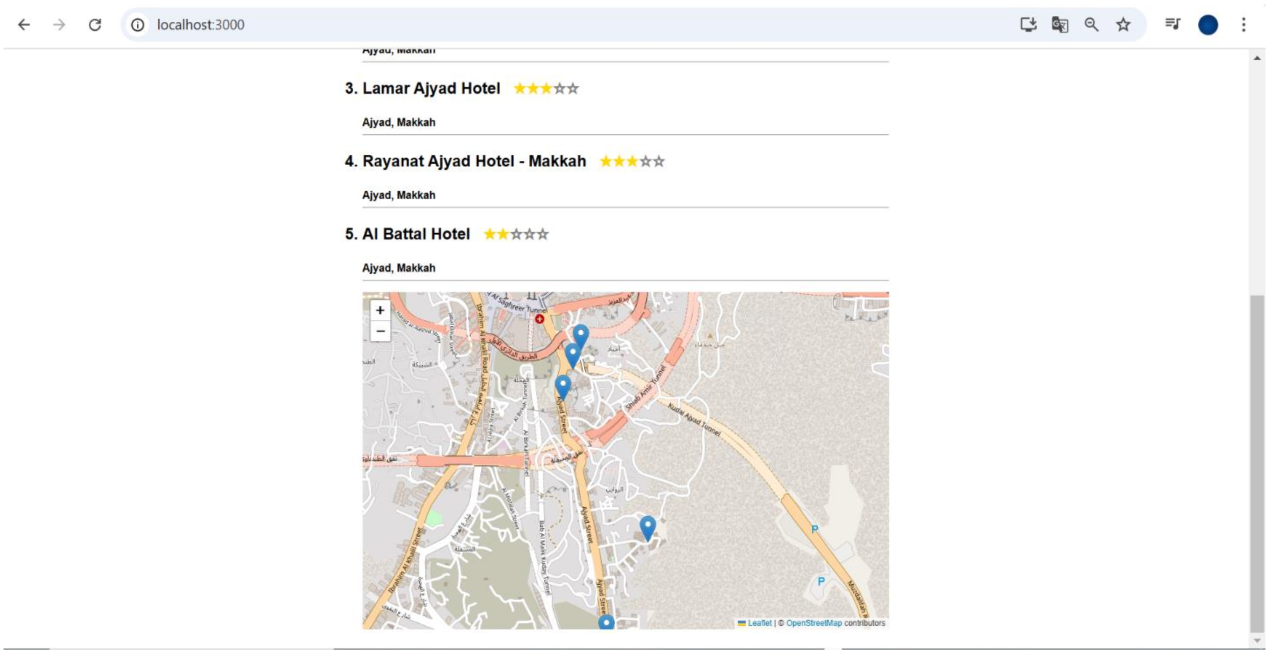
4. Rayanat Ajyad Hotel - Makkah ★★★☆☆

Ajyad, Makkah

5. Al Battal Hotel ★★☆☆☆

Ajyad, Makkah

Interface 3 :



3. Lamar Ajyad Hotel ★★★☆☆

Ajyad, Makkah

4. Rayanat Ajyad Hotel - Makkah ★★★☆☆

Ajyad, Makkah

5. Al Battal Hotel ★★☆☆☆

Ajyad, Makkah

. Conclusion

This report demonstrates the development of a hotel recommendation system based on text analysis techniques. The system effectively identifies similar hotels using TF-IDF and cosine similarity, providing a foundation for personalized travel recommendations. Future work will focus on incorporating user preferences and exploring more sophisticated recommendation algorithms to further enhance the system's performance.