Daniel Brilliant, Sowmya Chinthapally, Ruth Iang

DATA690 Fall 2023

**RECOMMENDER SYSTEM**

**INTRODUCTION**

Recommender systems are a pivotal component in today's data-driven landscape, especially in domains like e-commerce, media streaming, and content platforms. They are designed to predict and present items or content that users are likely to be interested in, enhancing user experience and engagement. The development of these systems incorporates a variety of methodologies, each addressing different aspects of recommendation and user preference modeling.

This project focuses on Amazon product data. The aim is to determine the most effective method for delivering personalized recommendations based on our requirements. The study will employ two methods of collaborative filtering (user-based and singular value decomposition) and large-language models. The following analysis also discusses contemporary trends in NLP, where the focus is on understanding and generating human-like text, extracting meaningful insights from large text corpora, and enhancing the interaction between humans and machines through language. This reflects a shift from traditional rule-based systems to more dynamic, context-aware models that learn from vast amounts of data.

**LITERATURE REVIEW**

This paper thoroughly examines the evolution of recommender systems, which are designed to predict and present items or content tailored to users' interests, thereby enhancing user experience and engagement. The development of these systems involves a diverse array of methodologies, and this paper extensively explores various aspects of recommendation and user preference modeling.

One dominant methodology in recommender systems is collaborative filtering, which operates on the principle of user similarity. This approach relies on user-item interactions, assuming that users who agreed in the past will continue to agree in the future. Algorithms are employed to identify patterns and similarities among users and items, often utilizing techniques such as matrix factorization, which includes methods like Singular Value Decomposition (SVD). Collaborative filtering is widely appreciated for its ability to handle large and sparse datasets and its effectiveness in uncovering latent user preferences. However, it faces challenges with new items (the cold start problem) and sparse data scenarios.

In recent years, significant advancements in natural language processing have opened new frontiers for improving recommendation systems. A notable technology in this regard is Langchain, which holds the potential to significantly enhance recommender systems. It introduces innovative approaches to database interaction through 'SQLDatabaseChain,' enabling the integration of natural language processing capabilities with traditional database management systems. This integration facilitates a more intuitive and user-friendly interface for querying and analyzing data.

Traditional recommender systems often rely on limited data sources, resulting in generic or inaccurate recommendations. To address this limitation and personalize the user experience, richer product data is essential, capturing user preferences, product features, and evolving trends. The paper introduces the SERP API and LangChain as a compelling solution to this challenge. The SERP API provides access to vast amounts of product data from various online sources, including product pages, reviews, and social media discussions. Meanwhile, LangChain excels at processing and analyzing this data, extracting valuable insights, and enriching product profiles. This integrated approach aims to provide more accurate and personalized recommendations, ultimately elevating the overall user experience.

Additionally, the integration of OpenAI’s GPT with Gradio is explored in this paper. This powerful combination unleashes the potential for developing accessible and user-friendly interfaces, making advanced Natural Language Processing (NLP) capabilities readily available to a broader audience beyond expert users.

**METHODOLOGY**

**COLLABORATIVE FILTERING MODELS:**

1. **USER BASED COLLABORATIVE FILTERING:** This section details the implementation and analysis of a Collaborative Filtering recommendation system, focusing on the user-based approach. Collaborative Filtering is a technique that leverages the preferences of similar users to provide personalized item recommendations. The methodology involves creating a User-Item sparse matrix, calculating user-user similarity using cosine similarity, and predicting user ratings. The system further recommends items to users based on their predicted preferences.

The implementation steps are outlined in detail below.

* Data Preparation:

The dataset includes user ratings associated with specific products, along with user and product identifiers to construct a user-item sparse matrix with the pivot table function, capturing user-product interactions and filling missing values with zeros.

* User-User Similarity Calculation:

Cosine similarity is computed to establish the similarity between users based on their ratings for common products, and the resulting matrix is used to identify users with similar preferences.

* Neighbor Identification:

The function accepts a user-user similarity matrix and an integer 'n' indicating the desired number of neighbors to identify for each user. It outputs a DataFrame where each row represents a user, and the columns contain the indices of the users who exhibit the highest similarity. It provides a structured presentation of the top N similar users for each user in the collaborative filtering scenario.

* Rating Prediction:

This function takes into account the mean rating given by each user as a baseline. It adjusts this baseline by considering how much the user's actual ratings deviate from their mean rating. The predictions generated reflect a personalized estimation of how a user might rate items, considering their unique rating tendencies and preferences. This approach allows for nuanced and individualized predictions, enhancing the accuracy and relevance of the recommendation system.

* Recommendation Results:

Presenting the top-n items based on their predicted preferences.

An example of the output is provided below:

A screenshot of a computer program

Description automatically generated

1. **MATRIX FACTORIZATION BASED COLLABORATIVE FILTERING:** Matrix factorization (MF) is a technique for decomposing a matrix into two or more lower-rank matrices. In the context of collaborative filtering, MF can be used to decompose a user-item interaction matrix into two matrices: a user-feature matrix and an item-feature matrix. These matrices represent the latent features that users and items share. By projecting users and items onto these latent features, we can compute their similarity and make predictions about unobserved ratings. Singular value decomposition (SVD) is a specific type of MF that is commonly used for collaborative filtering. SVD decomposes a matrix into three matrices: a diagonal matrix of singular values, a left singular matrix, and a right singular matrix. The singular values represent the relative importance of each latent feature. The left and right singular matrices encode the relationships between users and items, respectively.

Let’s examine the details below:

* Singular Value Decomposition (SVD):

Apply Singular Value Decomposition to the user-item array to factorize it into three matrices: U (user matrix), Sigma (diagonal matrix of singular values), and Vt (item matrix). By choosing a specific number of latent factors (k), SVD decomposes the original matrix into lower-dimensional matrices.

* Construct SVD Prediction Matrix:

Reconstruct the user-item matrix using the dot product of the decomposed matrices (U, Sigma, Vt). The resulting 'svd\_prediction' matrix represents the predicted user-item interactions, capturing underlying patterns and latent factors.

* Recommendation Generation:

Utilize the 'svd\_prediction' matrix to recommend items for specific users. Provide recommendations for a predefined set of users along with the specified number of top recommendations.

Below is a sample output

A screenshot of a computer error

Description automatically generated

**Personalized Recommendations with LLMs**

**(i) SQL: A Powerful Tool for Personalized Recommendations, Now Enhanced with Langchain**

SQL databases excel at managing vast amounts of user data, including demographics, preferences, and behavior patterns. This data is the fuel for powerful recommendation systems, and SQL provides the tools to extract its full potential.

Langchain's SQLDatabaseChain elevates the power of SQL for building personalized recommendations. It simplifies complex tasks by integrating data sources and processing steps into a single flow. This streamlines recommendation pipeline management, unlocks deeper data insights, and delivers an exceptional user experience.

Combining Langchain with SQL unlocks a new era of personalized recommendations. It empowers you to tailor suggestions to individual users and adapt to their needs in real-time. This dynamic approach fosters user engagement and creates a more rewarding experience for everyone.

**(ii) SERP API based Product Data for Recommender Systems**

SerpAPI, a popular commercial API, provides access to search results from Google, Bing, Yahoo, and DuckDuckGo, encompassing all major search engines. By harnessing SERP API's data acquisition capabilities and LangChain's processing and analysis strengths, we can unlock new levels of personalization and accuracy in recommendations.

SERP API provides access to vast amounts of product data that can be used to enrich product profiles and personalize recommendations. By extracting features and specifications from product pages and reviews, detailed product profiles can be built to enable personalized recommendations based on user preferences. SERP API can also be used to scrape user reviews, social media posts, and other online content to extract implicit and explicit user preferences related to products, services, or topics. This information can be integrated into user profiles to improve recommendation accuracy.

**(iii) OpenAI Assistant for Personalized Product Recommendations using Structured Prompts and Gradio Interface**

The integration of advanced language models, particularly those from OpenAI's Generative Pre-trained Transformer (GPT) series, with user interfaces, marks a significant advancement in the fields of natural language processing (NLP) and human-computer interaction. A key development in this area is the creation of the "OpenAI Assistant," which showcases the practical application of these advanced language models in user-friendly interfaces. The OpenAI Assistant emphasizes the importance of effective prompt structuring, which is critical in leveraging the full potential of these models. In this setup, each prompt is meticulously structured into three parts: the role, the context, and the task. The role defines the AI's function, setting the tone and approach for the interaction. It could range from an advisor to a tutor, shaping the AI's responses. The context provides background information or the specific scenario related to the query, enabling the AI to understand the situation better and tailor its responses. The task is the actual query or command, directing the focus of the AI to generate specific outputs.

This structured approach to prompting enhances the interaction's effectiveness, ensuring that the AI comprehensively understands the nuances and intentions behind each query. Furthermore, the emergence of "Prompt Engineering" as a crucial skill in AI interactions signifies the growing importance of crafting efficient and effective prompts. Prompt Engineers design prompts that maximize the AI's capabilities, ensuring accurate and relevant responses, thereby optimizing the performance of AI models in practical applications.

The OpenAI Assistant fosters user trust and understanding by providing insightful explanations for its recommendations, tailored to individual preferences. This is achieved through prompts specifically designed to request explanations alongside suggestions. By incorporating user feedback, the assistant continuously adjusts and refines its recommendations to ensure they are increasingly relevant and personalized. Moreover, it elegantly addresses the cold-start problem by leveraging prompts to gather user information for initial recommendations, even with limited data, ensuring a seamless experience from the outset.

**RESULTS:**

For both the user-based collaborative filtering and matrix factorization-based collaborative filtering models, the results appeared to effectively recommend products of interest that compared favorably to prior reviews by users. A set of recommendations from the user-based collaborative filtering model gave a user who reviewed shampoos, shave products, and items from Jack Black’s product line recommendations for products in these categories. A set of recommendations from the matrix factorization collaborative filtering model gave a user who reviewed items in the Jane Iredale makeup line recommendations for other Jane Iredale products and makeups. Additionally, for these models, root-mean squared error (RMSE) was calculated based on how close predicted product ratings by user were to the actual ratings. These results showed both models with minimal RMSE values of 0.0131 (user-based collaborative filtering) and 0.00481 (matrix factorization-based collaborative filtering). While these methods anecdotally and mathematically are effective, the advancement into the paradigm of LLMs takes the basis of these systems and creates more logical and personalized recommendations that go beyond the capabilities of simple collaborative filtering.

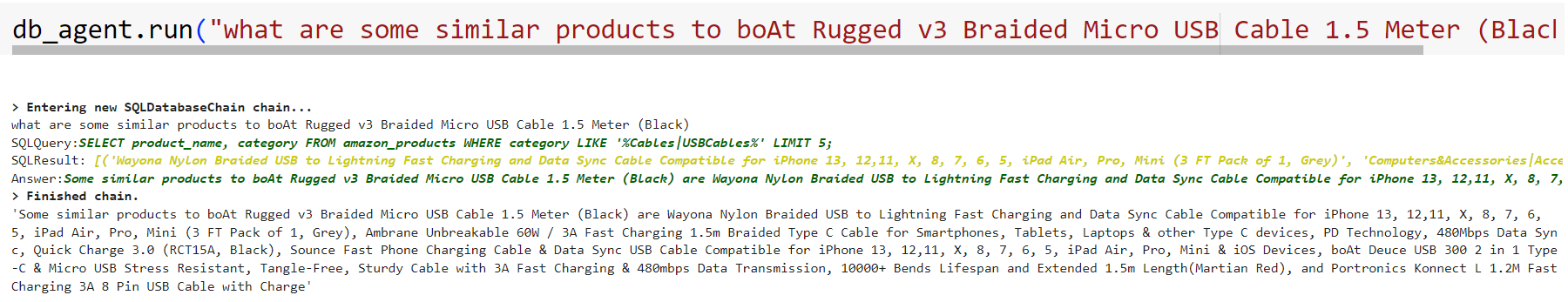
The effectiveness of SQLDatabaseChain in product recommendation was demonstrated through the seamless combination of SQL and large language models (LLMs). In this specific case, the system successfully identified similar products to a user query by leveraging the following capabilities:

(i) Efficient Data Access: SQL allows for the efficient storage and retrieval of product information.

(ii) Dynamic Queries: The LLM can dynamically generate SQL queries based on user queries, enabling the system to tailor recommendations to individual needs and preferences. In this instance, the LLM generated a query that identified products within the same category as the target product, ensuring relevant recommendations.

(iii) Personalized Recommendations: By combining user queries with information from the database, the system can personalize recommendations based on individual preferences and needs. This approach increases the likelihood that users will find the recommendations relevant and valuable.

The provided example output illustrates the system's ability to generate specific queries ("what are some similar products to boAt Rugged v3 Braided Micro USB Cable 1.5 Meter (Black)") and the corresponding SQL results, presenting similar products along with a transparent explanation.



This project also demonstrates the exciting potential of merging conversational AI with powerful technologies like OpenAI and Google Search to provide personalized product recommendations. It utilizes conversational AI agent, Mrkl, OpenAI's text-generation capabilities and Google Search results to generate relevant product recommendations. OpenAI's text-generation model "text-ada-001" is used for dialogue generation. SERP API is integrated to retrieve relevant product information. ChatOpenAI is used as the conversational AI agent framework. The user asks for recommendations for hair oils similar to Mielle Rosemary hair oil. The LLM identifies the user's intent to find product recommendations. It triggers a search action using SerpApi to find relevant hair oils based on the user's query. The LLM extracts key information from the search results, including product title, price, source, and rating. It analyzes the information and identifies products similar to Mielle Rosemary hair oil. The LLM selects three hair oils similar to the requested one. This implementation demonstrates the LLM's ability to understand user queries, perform relevant searches, and generate personalized product recommendations.

Below is a portion of output :



We have also explored the potential of OpenAI Assistant and structured prompts to generate personalized recommendations. We designed a structured prompt with three key components:

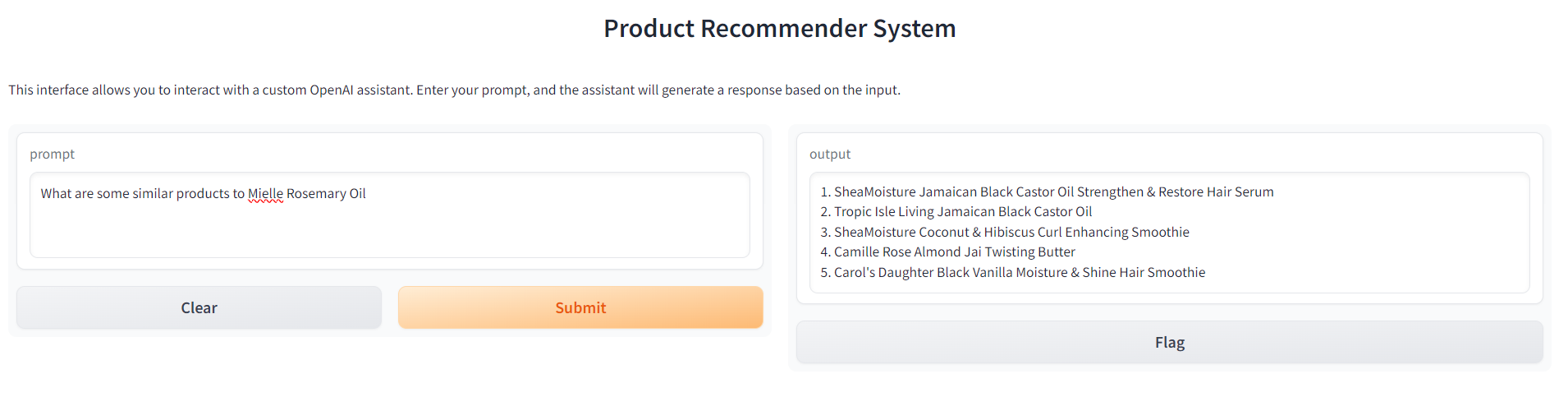
1. Role: The OpenAI Assistant assumed the role of a "Product Recommender," tasked with suggesting suitable alternatives to Mielle Rosemary Oil.

2. Context: The prompt incorporated information about Mielle Rosemary Oil, including its benefits and potential drawbacks. Additionally, it considered user preferences and desired outcomes to guide the recommendation process.

3. Task: The prompt explicitly defined the task, which was to generate a list of similar products to Mielle Rosemary Oil, taking into account the provided context and user preferences.

We presented the designed prompt to the OpenAI Assistant through a user-friendly interface, enabling natural language communication. The OpenAI Assistant then leveraged its knowledge of products and user preferences to generate a list of five alternative products that closely resembled Mielle Rosemary Oil. The generated recommendations were then analyzed to assess their relevance and suitability for the intended user. Additionally, user feedback was incorporated into the prompt for future iterations, allowing the OpenAI Assistant to continuously improve its recommendation accuracy and personalization. This approach demonstrates the effectiveness of combining OpenAI Assistant and structured prompts in generating personalized product recommendations. The structured approach ensures clear communication of user intent and preferences, while the advanced language processing capabilities of OpenAI Assistant enable the generation of relevant and tailored suggestions. This framework holds significant promise for enhancing user experience and satisfaction in the online product discovery and purchase process.

Below is a sample output:



**References:**

Balodi, T. (2020). *Review-based Recommendations System*. [Source code]. <https://www.analyticssteps.com/blogs/review-based-recommendations-system>

Das, A. (2019). *Amazon Recommendation System*. [Source code]. <https://github.com/theakshaydas/PGP-AIML/blob/master/CaseStudies/09-RE-Amazon.Product.Recommendation/Amazon%20Recommendation%20System.ipynb>

Khandelwal, R. (2023, October 20). The Next Generation of Learning Tools: LLM, LangChain, and Search Engines. Medium. <https://arshren.medium.com/the-next-generation-of-learning-tools-llm-langchain-and-search-engines-b12888dbcb6>

Ni, J. (n.d.). *Amazon review data (2018)* (Luxury Beauty, 5-core) [Data Set]. University of California, San Diego. <https://cseweb.ucsd.edu/~jmcauley/datasets/amazon_v2/>

Quaresma, F. (n.d). *Amazon Product Analysis & Recommendation System.* [Data Set]. Kaggle. <https://www.kaggle.com/code/flavioquaresma/amazon-product-analysis-recommendation-system>

Ramlochan, S. (2023, October 11). Using Large Language Models for Recommendation Systems. Prompt Engineering. <https://promptengineering.org/using-large-language-models-for-recommendation-systems/>

Rocca, B. (2019, June 2). Introduction to recommender systems. Medium; Towards Data Science. <https://towardsdatascience.com/introduction-to-recommender-systems-6c66cf15ada>