LLM-Powered E-Commerce Product Recommendations: A Big Data Approach to Optimize Product Search

Nihar Patel

Email: npatel17@stevens.edu

BIG DATA TECHNOLOGY TERM PAPER

Abstract—In today's rapidly growing E-commerce ecosystem, personalized recommendations have become a key feature in enhancing user experiences. This paper presents a scalable architecture that leverages web-scraped e-commerce data and integrates it with a fine-tuned Large Language Model (LLM) to provide real-time product recommendations. The system gathers product data from various online sources, preprocesses it using big data tools, and deploys an LLM to interpret user queries to recommend the most relevant products further. Using tools like Selenium, Scrapy, and open API endpoints data was scrapped efficiently. Technologies such as Apache Airflow, PostgreSQL, Apache Kafka, PySpark, and AWS are utilized to handle data ingestion, storage, streaming, and processing at scale. The proposed architecture efficiently combines data scraping, natural language understanding, and big data processing to enable a seamless recommendation engine, providing users with personalized product suggestions based on their written inputs.

Keywords— E-commerce, recommendation, Large Language Model, Selenium, Apache Airflow, Apache Kafka, PySpark

1. Introduction

Personalized product recommendations have trans formed the way consumers shop online, allowing them to find products tailored to their specific needs with minimal effort. Traditional recommendation systems often rely on collaborative filtering or content-based methods, but recent advances in natural language processing (NLP) have opened the door to more intuitive and user-friendly solutions. By deploying a Large Language Model (LLM) trained on a wide variety of product data, it is possible to generate highly relevant recommendations based on user inputs in natural language. This paper introduces an end-to-end system integrating periodic data scraping from the web, data streaming and transformation and data storage techniques, to fine-tune LLM to create a robust product recommendation engine. The system enables users to describe their desired product in a sentence or a query and based on that as input, it recommends the most suitable product from the database by analyzing the details of every e-commerce product from the database. The architecture employs Apache Airflow to automate the data scraping process, PostgreSQL to store structured data, Kafka to store real-time streaming, PySpark to process large-scale data, and AWS to deploy cloud data. This combination allows for a scalable and efficient pipeline to handle large datasets and real-time queries.

2. Background

Recommendation systems are a class of algorithms designed to provide users with personalized suggestions, addressing the challenge of information overload in modern digital environments. These systems have been widely adopted across various industries, including e-commerce, entertainment, education, and healthcare, to enhance user satisfaction and engagement. Broadly, recommendation systems can be categorized into three types: collaborative filtering, content-based filtering, and hybrid methods.

2.1. Existing Methods

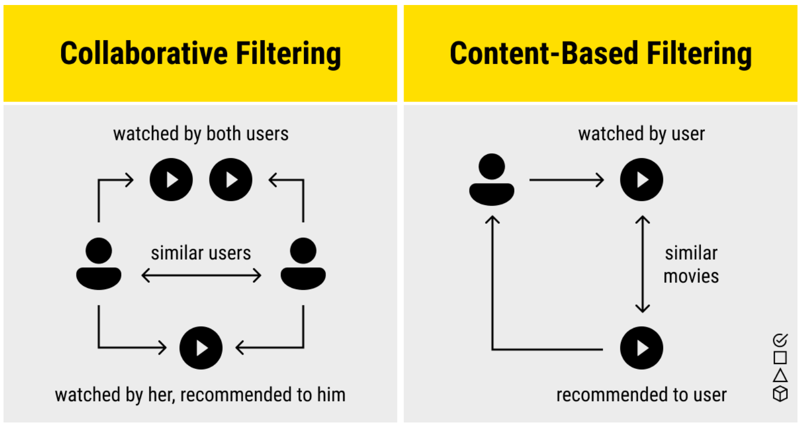
2.1.1. Collaborative Filtering. Collaborative filtering (CF) is one of the most widely used techniques in recommendation systems. It relies on user-item interaction data, assuming that users with similar preferences in the past will have similar preferences in the future. CF is further divided into two approaches:

• User-based Collaborative Filtering: Recommendations are generated based on the preferences of like-minded users [1]. While effective, this approach struggles with scalability as the number of users grows.

• Item-based Collaborative Filtering: This method, popularized by Amazon, focuses on item similarity rather than user similarity, offering better scalability and accuracy for large datasets [2].

Despite its popularity, CF faces challenges such as the cold-start problem, where new users or items with little interaction data are difficult to recommend.

2.1.2. Content-Based Filtering. Content-based filtering methods leverage the features of items and user profiles to make recommendations. These systems suggest items similar to those previously liked by the user, using algorithms like cosine similarity or decision trees [3]. The advantage of this approach lies in its ability to recommend niche items without relying on the preferences of other users. However, it is limited by its

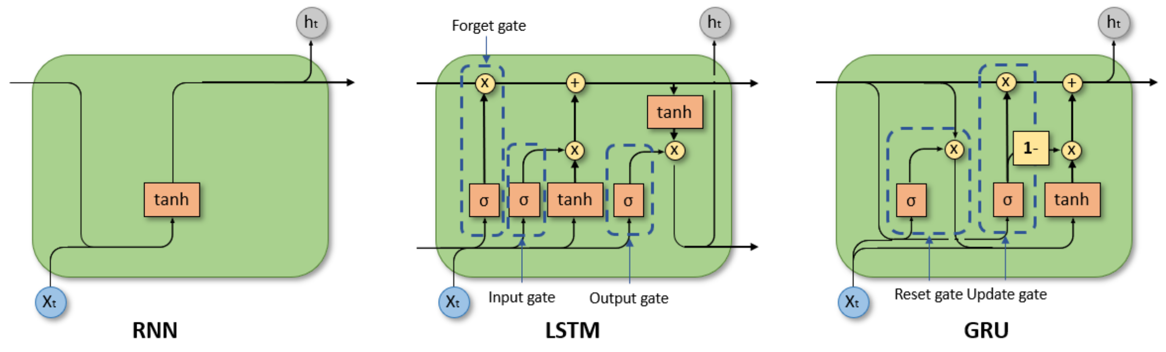


dependence on the availability and quality of item metadata, and it struggles to provide diverse recommendations due to the *serendipity problem*.

2.1.3. Hybrid Recommendation Systems. Hybrid methods combine collaborative and content-based filtering to overcome the limitations of each approach. They integrate multiple data sources and algorithms to enhance accuracy and robustness. For example, Netflix's recommendation algorithm combines user preferences with item metadata, using matrix factorization and deep learning models [4].

2.2. Context-Aware Systems using Deep Learning

In recent years, deep learning has revolutionized recommendation systems by enabling the extraction of high-level features from complex data types like images, text, and user interactions. Techniques such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) [Figure: 2] are increasingly used to model user preferences [5].



Context-aware recommendation systems further enhance personalization by incorporating additional contextual information, such as location, time, and device type, to refine suggestions [6]. These systems are particularly useful in mobile and IoT applications, where context plays a crucial role in user behavior.

While traditional recommendation systems rely heavily on explicit user feedback and structured data, the emergence of Large Language Models (LLMs) such as GPT and

BERT opens new possibilities for understanding unstructured user interactions. LLMs excel in capturing nuanced semantic relationships, making them promising tools for enhancing recommendations in domains such as conversational AI and personalized content delivery.

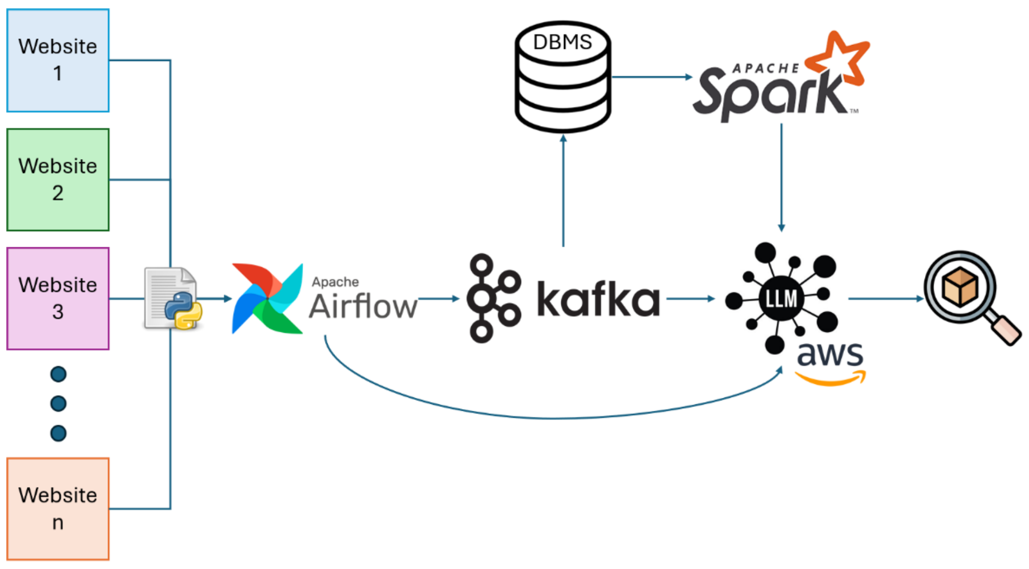
3. Methodology

First of all, we will see the proposed architecture and then we will cover the tools and technologies used in it.

3.1. Architecture

The architecture is designed to handle large volumes of e-commerce product data, process number of user queries in real time and deliver personalized product recommendations.

While architecture can be divided into the following processes:



1) Data Ingestion:

Web Scraping (Airflow): The system starts with Airflow orchestrating the scraping of product data from multiple e-commerce websites. Airflow schedules jobs that regularly collect product details, such as product names, prices, stock availability, size availability, product details such as product features, description, and materials used for manufacturing, and many other additional fields can be added to increase the knowledge gained for the product.

2) Data Storage and Management: PostgreSQL:

Scraped product data is stored in PostgreSQL, where it is organized and indexed for quick access. Historical data is maintained here, allowing for trend analysis and product comparison over time.

3) Real-Time Data Streaming: Kafka:

Kafka manages real-time streams of data updates from the scraped product details, such as price changes and availability status. This ensures that the product database and the LLM engine are always up-to-date.

4) Preprocessing and Embedding Generation: PySpark:

The product data is processed in bulk using PySpark. During this step, product descriptions are transformed into embeddings using natural language processing (NLP) techniques. These embeddings serve as the basis for matching user queries to relevant products.

5) Query Processing and Product Recommendations: LLM Deployment: When a user enters a product query (e.g., “I want red office shoes under $200”), the LLM converts

the query into a vector representation. This vector is compared against the precomputed product embeddings to identify the products most aligned with the user's intent.

6) Cloud Deployment: AWS: The entire system is deployed on AWS, with EC2 handling model inference and S3 storing both raw and processed product data. AWS Lambda is used for quick, serverless execution of real-time tasks, such as returning recommendations based on user input.

3.2. Tools & Technologies

It is very essential to learn which, why, and where to use each technology. So before diving deep into the project, we will get familiar with each tool used in this project.

1) Apache Airflow: Airflow is an open-source platform used to programmatically author, schedule, and monitor workflows. It is a crucial part of this system for automating the scraping of product data from multiple e-commerce websites. Airflow manages the entire scraping pipeline, ensuring that data is gathered at regular intervals and flows into the system without manual intervention.

2) PostgreSQL: PostgreSQL is a powerful open- source relational database used to store the scraped product data. As a structured SQL database, it offers robust querying capabilities, making it an ideal choice for storing, indexing, and retrieving product information (e.g., name, price, description, and availability).

3) Apache Kafka: Kafka is a distributed streaming platform that allows for real-time data ingestion and processing. In this project, Kafka streams updates from e-commerce websites—such as product price changes or availability fluctuations—and ensures that the LLM recommendation engine always has access to the most up-to-date data.

4) PySpark: PySpark, the Python API for Apache Spark, is used for large-scale data processing and manipulation. PySpark handles the preprocessing of the product data (cleaning, transforming, and feature extraction) and generates embeddings for product descriptions, which are then used by the LLM to match user queries to relevant products.

5) Large Language Model (LLM): The core of the recommendation engine is a fine-tuned LLM (e.g., GPT-3 or BERT), trained on product descriptions and user queries. The model interprets the user's natural language input and converts it into vector representations. It then compares these vectors with the preprocessed product embeddings to recommend the most relevant products.

6) Amazon Web Services (AWS): AWS is used for cloud deployment, providing scalable compute and storage resources. The LLM model is deployed on AWS EC2 instances, and processed product data is stored in AWS S3 for easy access. Additionally, AWS Lambda is used to trigger specific tasks in response to real-time queries, enabling quick and efficient processing of user requests.

References

[1] Jonathan L Herlocker, Joseph A Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 230–237, 1999.

[2] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item based collaborative filtering recommendation algorithms. In Proceedings of the 10th International Conference on World Wide Web, pages 285–295, 2001.

[3] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In The adaptive web, pages 325–341. Springer, 2007.

[4] Carlos A Gomez-Uribe and Neil Hunt. The netflix recommender system: Algorithms, business value, and innovation. ACM Transactions on Management Information Systems (TMIS), 6(4):13, 2015.

[5] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning-based recommender system: A survey and new perspectives. ACM Computing Surveys (CSUR), 52(1):1–38, 2019.

[6] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. AI Magazine, 32(3):67–80, 2011.