الجمهورية الجزائرية الدمقراطية الشعبية

People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

University August 20 1955 of Skikda

Faculty of Sciences

Department of Computer Science

End-of-study thesis for the academic Master's degree in computer science

Specialty

Networks and Distributed Systems (RSD)

Theme

Filter Bubble effect in recommender system: case study in e-commerce

Realized by

BOULOUDNINE Hayem

LALAOUI Foued

Supervised by

Dr. NABET Aicha

2024-2025

Acknowledgments and Dedication

All praise is due to Allah by whose grace and guidance this work has been completed, we pray that he accepts it and makes it beneficial for us and us and for others.

We are sincerely grateful to our parents for their unconditional love, prayers, and constant support throughout this journey. Their presence has been a true source of strength.

We also extend this gratitude to our siblings, loved ones, and friend, whose presence and support meant a great deal us along the way.

Our appreciation goes to our supervisor, Dr. Nabet Aicha, for her valuable guidance, expertise, and continuous support, which played a vital role in shaping the direction of our research.

We also thank the faculty members of the Faculty of Technology at the University of 20 August 1955, Skikda, for their efforts and dedication to academic excellence.

Finally, we dedicate this project to everyone striving to develop intelligent technologies that enhance user experience in the field of e-commerce. We hope that our work on recommender system serves as meaningful contribution to this dynamic domain and inspires further innovation.

Abstract

In the rapidly growing e-commerce landscape, recommender systems have emerged as a core component of creating personalized user interactions, but this can lead to unintended consequences, such as "filter bubbles," where users only see repeated content with similar qualities determined by their historical data. Filter bubbles can damage long run user satisfaction by limiting diversity.

This project presented the development of a real-time hybrid recommender system combining collaborative filtering (using ALS) with content-based filtering (using TF-IDF and cosine similarity), primarily to reduce the filter bubble effect while optimizing relevance and diversity of recommendations.

The methodology consists of two main phases:

- An offline phase, where models are trained and evaluated using a large-scale Amazon dataset.
- An online phase, where the trained system is integrated into a Django-based e-commerce platform that dynamically updates recommendations in response to real-time user interactions.

The results of the experiment show substantial improvements in precision, recall, and recommendation diversity. It was also shown that the system used in the experiment is adaptable to user behavior as needed, which improves engagement and user satisfaction with the recommendations.

This work demonstrates the efficacy of a hybrid and adaptive solution to solving modern recommendation challenges, specifically as it relates to providing recommendations with a balance of personalization and content exploration and diversity in dynamic real-time environments.

Keywords: Recommender system, Filter bubble effect, Hybrid recommender engine, Real time recommender system

Résumé

Le domaine du e-commerce est en pleine expansion, les systèmes de recommandation sont devenus un élément essentiel de la personnalisation des interactions utilisateur. Cependant, cela peut entraîner des conséquences inattendues, comme l'apparition des effets bulles, c'est dure les utilisateurs ne voient que du contenu dans la même catégorie présentant des qualités similaires, déterminées par leurs données historiques. Ces bulles peuvent nuire à la satisfaction utilisateur à long terme en limitant la diversité.

Ce projet présente le développement d'un système de recommandation hybride temps réel combinant le filtrage collaboratif (par ALS) et le filtrage basé sur le contenu (par TF-IDF et similarité cosinus) avec une phase de diversification et une mise à jour de recommandation en temp réel pour réduire l'effet de bulles tout en améliorant l'expérience utilisateur.

L'approche adoptée comporte deux phases :

- **Phase 1 hors ligne** : consiste à créer un moteur de recommandation hybride selon une architecture en cascade puis l'enrêner et le tester sur une large dataset d'Amazon ;
- **Phase 2 en ligne** : permet d'intégrer l'API du moteur hybride dans une plateforme e-commerce conçue avec Django permettant des mises à jour de recommandations en temps réel et un enrichissement du modèle.

Les résultats du test montrent des améliorations en termes précision et de diversité des recommandations. Ce projet montre qu'une solution hybride avec une recommandation temps réel peut diminuer l'effet de bulle et améliore la diversité du contenu pour les utilisateurs.

Mots clés : Système de recommandation, effet de bulle, moteur de recommandation hybride, Système de recommandation temps réel

Table of contents

Table of figures

Table of figures

General introduction

In the digital age, the recommendation systems have become an integral part of providing experiences dedicated to users in many fields such as electronic news and social networks, especially in the field of e -commerce, as these systems aim to predict the user preferences and suggest appropriate products or resources, through analyzing the user behavior, characteristics of products, and data structure.

With the rapid development of e-commerce platforms and the emergence of a huge number of products, there is an urgent need for intelligent recommendation technologies that operate in real time. Although traditional systems such as collaborative queuing and content-based filtering have proven their effectiveness, they still suffer from several challenges, such as the cold-start problem and data sparsity. Most existing works have mainly focused on addressing the cold-start problem; however, a more critical issue is the "filter bubble" — a phenomenon that occurs when algorithms repeatedly suggest the same type of content to a user based on their past behavior, reinforcing their current preferences and reducing their exposure to other diverse and innovative options. Although this reduction may seem beneficial at first, it can later lead to reduced user satisfaction.

In this project, we propose a recommender system based on cascade hybrid recommendation engine. The first stage, we use a collaborative filtering to identify relevant products based on the preferences of similar users, while the second stage, content filtering is selected to reorder results according to product characteristics and individual user preferences, such as category, brand, etc. at the end of the process a diversification step is planned. This sequential integration enhances both the accuracy and diversity of recommendations and allows the system to adapt to user interactions. In the context of e-commerce, where user interests change rapidly and product diversity is highly variable, this cascade hybrid model provides a flexible and scalable solution that combines personalization and discovery with reel time recommendation. Our approach is composed of two phases: an offline phase using to construct a hybrid model with diversification techniques, and an online phase that performs real-time recommendations all with the goal of mitigating the filter bubble problem and enhancing user satisfaction.

This work has been structured into four main chapters:

Chapter 1: Recommender System

This chapter is a review of the fundamental components of recommender system technologies, its types (collaborative, content-based, hybrid), and their importance in e-commerce.

Chapter 2: The Filter Bubble Effect in Recommender Systems

This chapter review the filter bubble effect, why it is a problem for diversity, and possible means of reducing its detrimental effects in recommendation systems.

Chapter 3: Our Approach

This chapter explains our proposed real-time hybrid recommendation approach, and the specific ways we shall increase personalization and reduce repetitions through algorithmic combination.

Chapter 4: Implementation of our approach

This chapter discusses the real-world development of system in two stages (offline and online), including the various tools, and evaluates the systems of real-time capabilities.

Chapter 1

Recommender systems

1.1 Introduction

The Sway that customers obtain goods and services has been completely transformed by the development of the internet. These days, e-commerce sites provide a wide range of goods, from electronics and books to travel services and clothing. Although there are many options available to consumers due to this abundance, users may find it difficult to find products that suit their unique needs and preferences due to information overload.

Recommender systems (RS) have become indispensable tools in the digital marketplace as a solution to this problem. By providing individualized product recommendations based on user behavior, preferences, and past data, these systems improve the shopping experience and make decision-making easier. Recommender systems assist users in efficiently navigating large product catalogues by prioritizing and filtering pertinent information.

In this chapter, we will review the fundamentals of recommender systems in the context of e-commerce. We will investigate several approaches used in recommender systems, such as collaborative filtering, which relies on users' interactions with items and similarities between users or items; content-based filtering, which recommends items based on product characteristics and user past preferences; and hybrid methods, which combine collaborative and content-based filtering techniques to leverage the strengths of each and mitigate weaknesses. Hybrid methods are among the most effective, providing more accurate and reliable recommendations by integrating multiple sources of information. We will also discuss the benefits of using recommender systems, such as increased sales and enhanced customer engagement, along with the associated challenges, such as algorithmic biases and data privacy issues.

1.2 What is a Recommender system?

A recommendation system is an artificial intelligence or AI algorithm, usually associated with machine learning, that uses Big Data to suggest or recommend additional products to consumers. These can be based on various criteria, including past purchases, search history, demographic information, and other factors. Recommender systems are highly useful as they help users discover products and services they might otherwise have not found on their own.

Recommender systems are trained to understand the preferences, previous decisions, and characteristics of people and products using data gathered about their interactions. These include impressions, clicks, likes, and purchases. Because of their capability to predict consumer interests and desires on a highly personalized level, recommender systems are a favorite with content and product providers. They can drive consumers to just about any product or service that interests them, from books to videos to health classes to clothing [1][2] [3].

1.3 Objects of Recommender system?

- **Personalization:** Personalize contents and product recommendations based on user's personal preferences, tastes, and behaviors;
- **Information filtering:** Filter out irrelevant items to reduce information overload and only show what is significant to the user;
- **Improving user experience:** Make the user experience by making it easier, more effective, and more enjoyable for the user to find useful content;
- **Increasing interaction:** Encourage users to interact more with the platform (i.e., watch more video, read more articles, buy more products);
- **Increase sales or retention (for businesses):** Achieve business goals, like increasing conversion rates, avg. order value, or user retention by recommending better products.
- **Discoverability:** help users discover new or lesser-known products that align with their preferences, not just popular products;
- **Efficient decision-making:** Reduce friction to help users make faster, better decisions (i.e., when picking a movie, book, or product);
- **Customer trust & satisfaction:** Maintain long-term relationships by providing continuously related and satisfying recommendations [4], [17], [18].

1.4 Challenges of the recommender systems

- **Initial data sparsity**
    - New User: When the system has no history of a new user's preferences.
    - New Item: When the system has a new item that has yet to have any interactions, therefore cannot be recommended.
- **Data Sparsity**
    - In big systems, users generally only rate a limited subset of items, which results in a very sparse user-item matrix that complicates the learning of preferences.
- **Privacy Issues**
    - The data captured and used to represent user behavior raises issues regarding data safety, user knowledge of how data is collected and used, and ethical tracking.
- **Over-Specialization / Filter Bubble**
    - Recommenders can create filter bubbles by essentially only recommending content that is a version of what has been viewed before, and limit access to dissimilar information and views (a challenge that our work aims to address).
- **Evaluation Challenges**
    - Offline metrics such as RMSE or precision are not necessarily useful indicators of real user satisfaction in real-life. Online A/B experimentation is better but expensive and slow [20].

1.5 Types of recommender systems

Now that we have defined recommender systems, their objective, usefulness, and the driving force behind recommender systems, in this section, we introduce different types of popular recommender systems in use [5].

1.5.1 Collaborative filtering recommender systems

Collaborative filtering recommender systems are basic forms of recommendation engines. In this type of recommendation engine, filtering items from a large set of alternatives is done collaboratively by users' preferences. The basic assumption in a collaborative filtering

recommender system is that if two users shared the same interests as each other in the past, they will also have similar tastes in the future. If, for example, user A and user B have similar movie preferences, and user A recently watched Titanic, which user B has not yet seen, then the idea is to recommend this unseen new movie to user B. The movie recommendations on Netflix are one good example of this type of recommender system. Figure 1 shows an example of collaborative filtering used by Netflix platform.

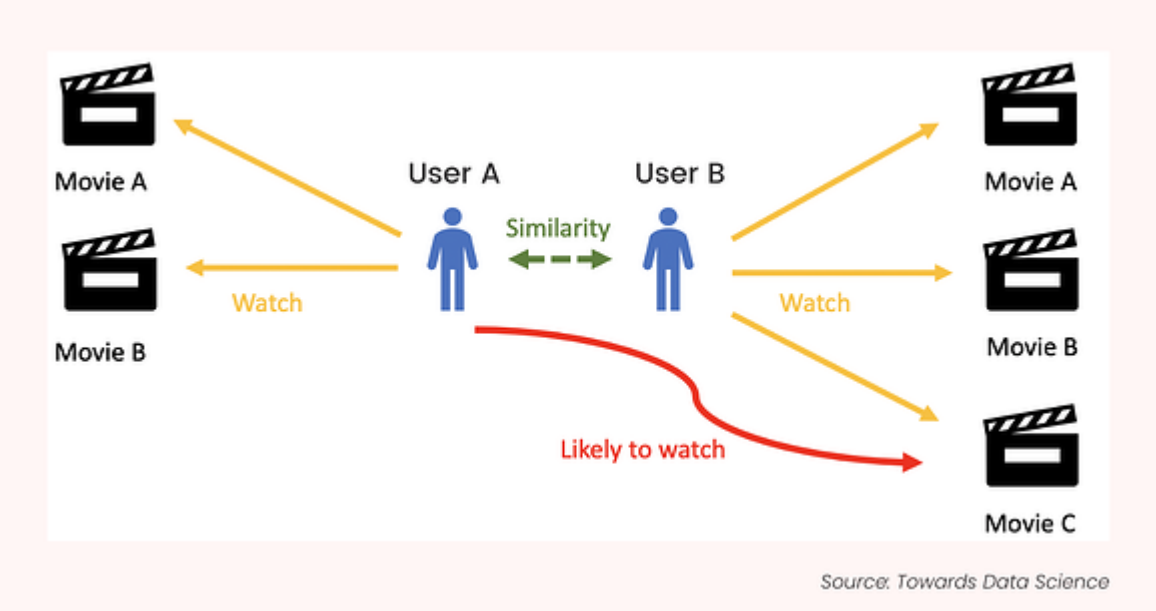


Figure 1 Netflix's use collaborative filtering [3]

There are two types of collaborative filtering for recommender systems:

- **User-based collaborative filtering**

  In user-based collaborative filtering, recommendations are generated by considering the preferences in the user's neighborhood. User-based collaborative filtering is done in two steps:

  - Identify similar users based on similar user preferences, for example, by calculating the similarity score using mathematical methods such as cosine similarity, Pearson correlation, or Euclidean distance, which are measures that help determine how close users' ratings of some items are.

  - Recommend new items to an active user based on the rating given by similar users on the items not rated by the active user.
- **Item-based collaborative filtering** in item-based collaborative filtering, the recommendations are generated using the neighbourhood of items. Unlike user-based collaborative filtering, we first find similarities between items and then recommend non-rated items which are similar to the items the active user has rated in past. Item-based recommender systems are constructed in two steps:
  - Calculate the item similarity based on the item preferences.
  - Find the top similar items to the non-rated items by active user and recommend them.

While building collaborative filtering recommender systems, we will learn about the following aspects:

- How to calculate the similarity between users?
- How to calculate the similarity between items?
- How recommendations are generated?
- How to deal with new items and new users whose data is not known?

The advantage of collaborative filtering systems is that they are simple to implement and very accurate. However, they have their own set of limitations, such as the Cold Start problem, which means, collaborative filtering systems fails to recommend to the first-time users whose information is not available in the system [5].

1.5.2 Content-based recommender systems

In collaborative filtering, we consider only user-item-preferences and build the recommender systems. Though this approach is accurate, it makes more sense if we consider user properties and item properties while building recommendation engines [5].

we use item properties and user preferences to the item properties while building content-based recommendation engines.

As the name indicates, a content-based recommender system uses the content information of the items for building the recommendation model. A content recommender system typically contains a user-profile-generation step, item profile-generation step-and model-building step to generate recommendations for an active user. The content-based recommender system recommends items to users by taking the content or features of items and user profiles. As an example, if you have searched for videos of Lionel Messi on YouTube, then the content-based recommender system will learn your preference and recommend other videos related to Lionel Messi and other videos related to football.

In simpler terms, the system recommends items similar to those that the user has liked in the past. The similarity of items is calculated based on the features associated with the other compared items and is matched with the user's historical preferences. Figure 2 shows an example of content-based filtering.

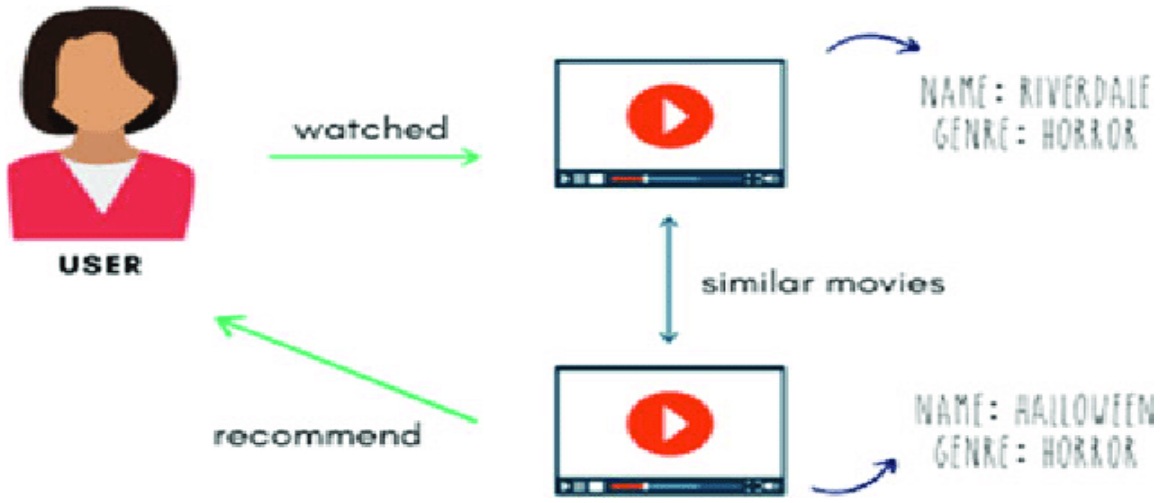


Figure 2 Example of content-based filtering.

While building a content-based recommendation system, we take into consideration the following questions:

- How do we choose content or features of the products?
- How do we create user profiles with preferences similar to that of the product content?
- How do we create similarity between items based on their features?
- How do we create and update user profiles continuously?

1.5.3 Core Function of Content-Based Recommenders

1. Item Profiling

- Each item (e.g., article, movie, product) is analyzed to extract key features—such as keywords, genre, brand, or metadata—and represented as a feature vector [43].
- For example, a movie might be represented by genre flags (action =1,comedy = 0), or a music track by its artist, style, tempo, etc.

2. User Profiling

- The system builds a profile for each user based on the items they have interacted with. The preferences are cumulated, creating a user feature vector, for example by averaging or using TF-IDF weights.

3. Similarity Computation

- Once the user profile has been created the system will compute the similarity between the user preference vector and the item vectors - usually by calculating cosine similarity.

4. The items which are the most similar (highest cosine score) to the user profile "win".

5. Filtering & Ranking

- Items are ranked by similarity score. The top matches are recommended to the user.

For example

- Suppose a user enjoys movies that are action and science fiction. These preferences are encoded in their profile (e.g., [1, 0, 1] for action, romance, science fiction).
- New movies are also vectorized. Say Movie A is [1, 0, 1] and Movie B is [0,1, 1].
- The system recommends Movie A because it has a higher cosine similarity with the user a better match of features [43].

While content-based recommendation systems achieve accurate results by matching item characteristics with user preferences, they are limited by the richness of user and item data. In addition, they suffer from major issues such as the cold-start problem and the tendency to reinforce existing user preferences, which may lead to a filter bubble effect. To overcome these limitations

and leverage the strengths of both content-based and collaborative filtering, hybrid recommender systems have emerged as a more comprehensive and robust solution

1.5.3 Hybrid recommender systems

This type of recommendation engine is built by combining various recommender systems to build a more robust system. By combining various recommender systems, we can replace the disadvantages of one system with the advantages of another system and thus build a more optimized system. For example, by combining collaborative filtering methods, where the model fails when new items don't have ratings, with content-based systems, where feature information about the items is available, new items can be recommended more accurately and efficiently [2].

For example, if you are a frequent reader of news on Google News, the underlying recommendation engine recommends news articles to you by combining popular news articles read by people similar to you and using your personal preferences, calculated using your previous click information. With this type of recommendation system, collaborative filtering recommendations are combined with content-based recommendations before pushing recommendations. Figure 3 shows an example of hybrid engine for recommender system.

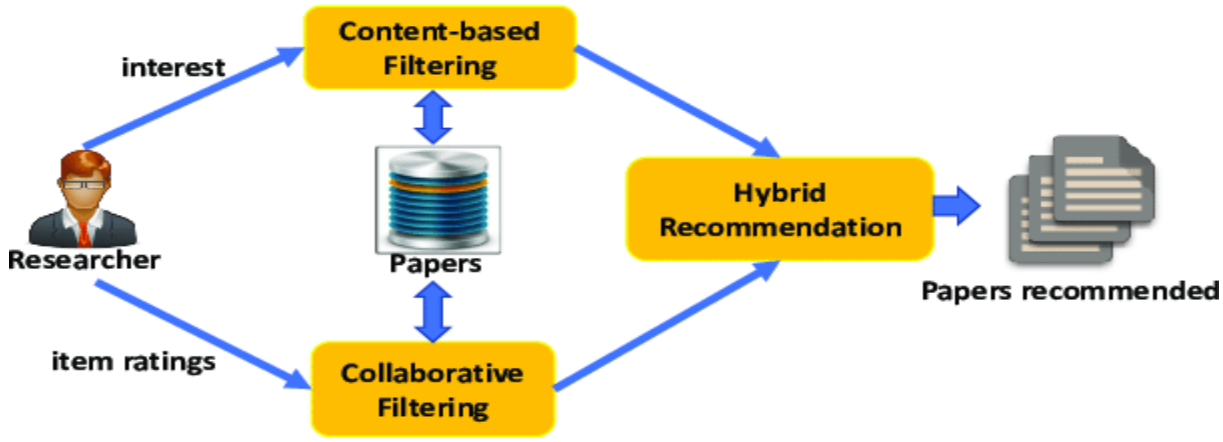


Figure 3 Example for hybrid engine.

Before building a hybrid model, we should consider the following questions:

- What recommender techniques should be combined to achieve the business solution?
- How should we combine various techniques and their results for better predictions?

The advantage of hybrid recommendation systems is that they increase the efficiency of recommendations compared to using only one technology. They also provide a balanced mix of recommendations to the user, whether at the personal level or at the "neighborhood" level. They are divided into several types, the most important of which are [15]:

- **Weighted Hybrid:** Recommendations from each algorithm are calculated and then combined using pre-defined weights, which express the importance of each recommendation source in the final result [15].
    - For example, on a platform like Amazon, the system can allocate 70% of recommendation results based on what similar users have purchased (collaborative filtering), and 30% based on the similarity of product features (content-based). If the content algorithm shows that a particular product is similar to what the user has previously purchased, but is not popular among others, it will be given less weight in the final result [15].
- **Switching Hybrid:** One algorithm is chosen from a set of algorithms, depending on a specific situation, user type, or expected recommendation quality.
    - Example: In a new movie streaming app, if the user is new and hasn't yet rated or interacted with the movie, an algorithm based on the movie's genre or rating (such as action or comedy) is used. After the user begins interacting, the system gradually transitions to using recommendations based on the behavior of similar users [15].
- **Mixed Hybrid:** Recommendations generated by multiple algorithms are presented to the user at the same time, without prior merging, giving them a variety of suggestions.
    - For example, on an e-commerce platform like Noon or Jumia, you might find two different sections: "Suggested products based on your previous purchases" (Collaborative Filtering) and, on the same page, "Similar products to this product" (Content-Based). The user sees both recommendations together and chooses based on their preference.

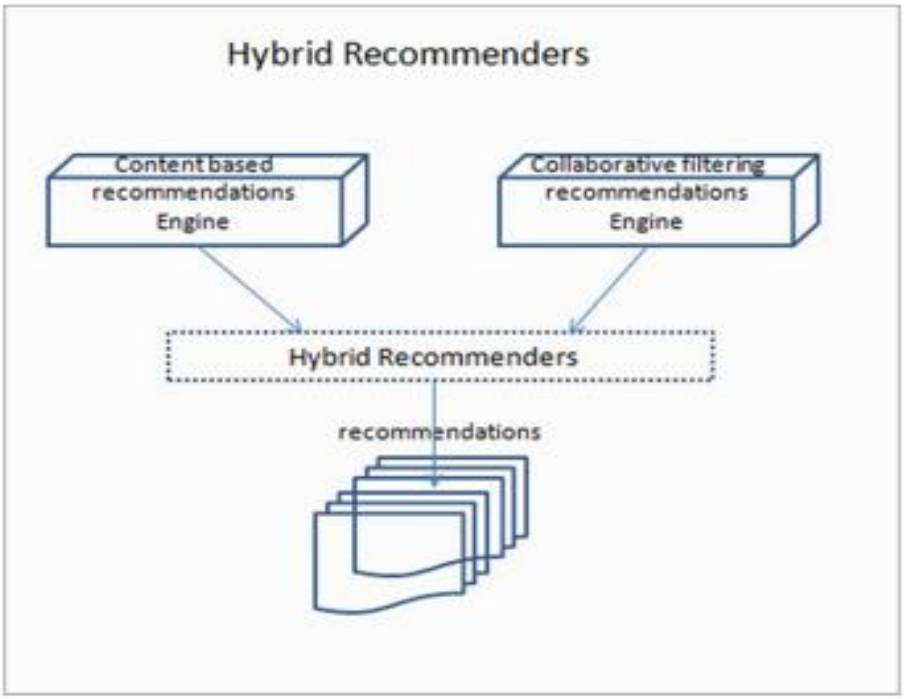


Figure 4 Mixed Hybrid recommender system

- **Cascade Hybrid:** A first algorithm is used to filter or sort the results, and then the results are passed to a second algorithm for further optimization.
    - Example: On a clothing website, a list of products is first selected based on the behavior of other users who purchased the same item (collaborative filtering). These products are then ranked based on characteristics such as the current user's preferred color or brand (content-based filtering). The product that best matches their preferences appears higher in the rankings [15].

(For this project, we adopted a cascade hybrid model, due to its ability to leverage the strengths of both collaborative filtering and content filtering in a sequential manner).

While this type of system mitigates the filter bubble effect, the continued evolution of user preferences and real-time needs requires a system capable of reacting instantly to these changes. This is where real-time recommendation systems become a strategic solution for supporting continuous personalization and intelligent adaptation to the user.

1.6 Real-time recommender systems

In recent years, recommendation systems have become an essential part of the digital user experience, especially on e-commerce platforms, where they provide immediate and relevant suggestions based on real-time user interactions. This performance relies primarily on the power of big data, which provides user data such as browsing history, previous purchases, and interaction preferences, allowing recommendation systems to generate accurate and effective recommendations.

Effective recommendation systems must be reliable, scalable, and respond in real-time to a large number of users. To achieve this, emerging technologies such as Apache Spark have emerged, which rely on in-memory processing and support live data streaming to generate recommendations quickly and accurately.

An example of this is the seemingly simple "You may also like" feature on shopping websites, which is driven by a complex system that processes live user interactions and adapts to changing multi-criteria preferences. Real-time recommendation systems go beyond analyzing historical data; they rely on reading current patterns and providing instant recommendations that enhance the user experience and increase loyalty [5].

1.6.1 How to build a recommender system?

Building a recommendation system involves several integrated stages that form a closed cycle, where the results are used to continuously feed and refine hypotheses. The following figure illustrates this cycle, and we will explain each stage separately below [2]:

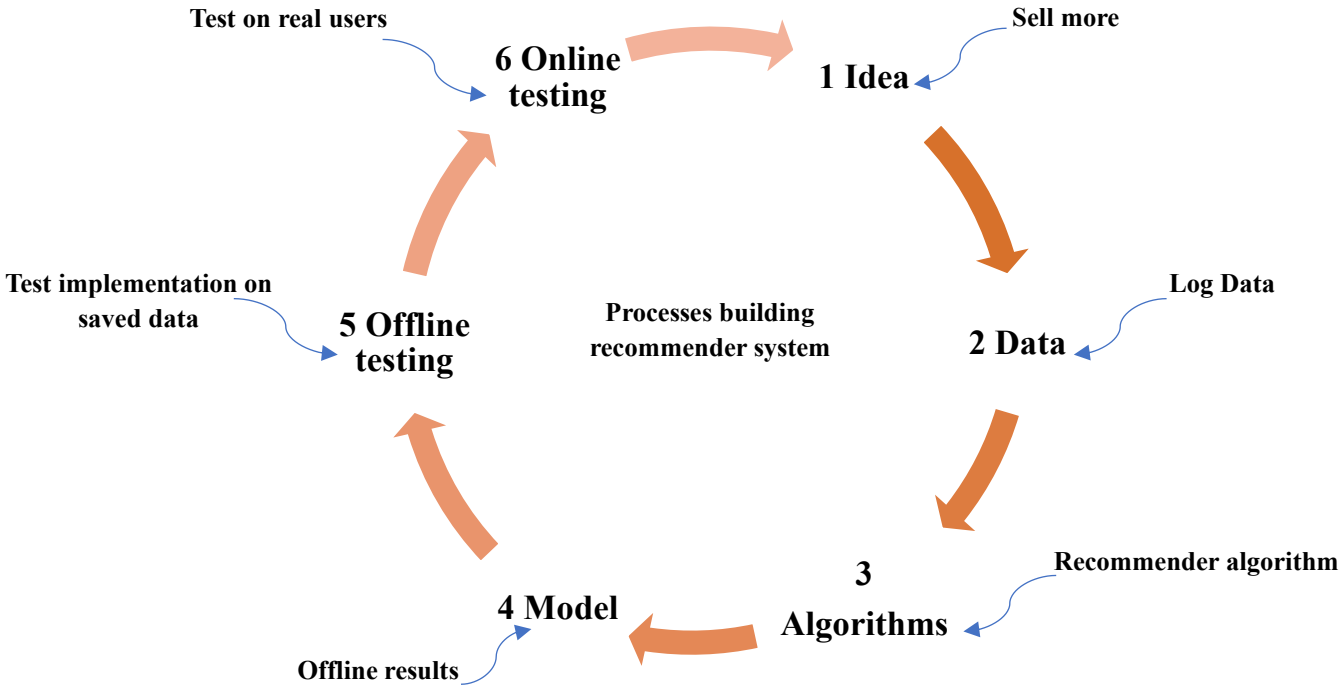


Figure 5 General processes to build a recommender system [2]

- **Idea:** The idea is initially defined as a basic idea or hypothesis upon which the system is based. This is often based on a business goal such as "increasing sales" or "improving user experience." This hypothesis determines the overall direction of the system. For example, "Customers who purchased a particular product might be interested in similar products. "
- **Data Collection:** After defining the idea, the next step is to collect the necessary data, such as browsing history, reviews, purchases, or product-related characteristics. This data constitutes the raw material upon which the system will rely. Example: Extracting user purchasing behavior data from an e-commerce platform's history.
- **Algorithms:** In this step, an appropriate recommendation algorithm is selected, such as collaborative filtering, content-based filtering, or a hybrid model. The selection should preferably be based on the nature of the data and the system's objectives. For example, use collaborative filtering if user interaction data is available.

- **Model**: The recommendation model is trained using the collected data, depending on the chosen algorithm. At this stage, patterns and preferences are extracted to generate recommendations later.
- **Offline Testing:** Before deploying the model, it is tested on historical data to simulate user behavior and evaluate performance. Metrics such as precision, recall, and coverage are relied upon.
- **Online Testing**: After successful offline testing, the model is tested in a real-world environment with actual users. A/B testing is used here to measure the impact of recommendations on user behavior, such as the number of clicks or purchase rate.

One of the key challenges facing recommender system developers is how to accurately evaluate the performance of these systems. A system may appear effective in terms of its architecture or algorithms, but it may not actually achieve user satisfaction or provide high-quality recommendations.

1.6.2 How is performance measured in evaluating recommendation systems?

Evaluating the performance of recommendation systems is a key step in measuring the effectiveness of the system in providing appropriate suggestions to users. Evaluation methods vary depending on the type of system and the purpose of the recommendations. Among the most commonly used metrics in this field are accuracy, precision, and recall, especially in recommendation cases where the interaction between the user and the item is categorized (for example, whether the user will like the product or not):

- **Accuracy**: is the proportion of correct predictions (either positive or negative) out of the total of all predictions made by the model.
  - In recommender systems, accuracy reflects how well the system's predictions are generally accurate [36,37].
- **Precision**: Positive accuracy measures the percentage of correct recommendations out of all recommendations that the system has identified as appropriate or relevant.
  - This metric reflects the system's "confidence" in the recommendations it provides. In other words: Of all the items the system deemed relevant to the user, how many were actually relevant? [36,38]

- **Recall:** Recall is the percentage of suitable items that the system was able to suggest out of all the actually suitable items.
    - Recall measures how well a system can "discover" as many items as possible that are of interest to the user. This metric is important in situations where missing a relevant item would be undesirable (such as a recommendation for urgent offers or important content) [36][38].

Recommender systems have become increasingly present across various domains such as healthcare (to suggest treatments or diagnoses), education (to personalize learning paths), and fashion (to recommend styles or outfits). However, their role becomes particularly crucial in e-commerce platforms, where the abundance of products and intense competition make intelligent, personalized recommendations essential for enhancing user experience and driving sales.

1.6.3 Recommender systems in e-commerce

In the world of e-commerce, competition is fierce, and users are faced with countless products. This is where a recommendation system comes in as a smart tool that guides users and helps them make decisions quickly, without the hassle of searching. For this reason, choosing a recommendation system for e-commerce is a strategic and important choice, and its importance lies in:

- Improving the personal user experience: Users find products that suit their tastes and needs without having to search, which increases their comfort and loyalty to the site. For example, Amazon, Zara, Noon... provide you with "suggested products" based on your behavior.
- Directly increase sales: The system suggests complementary or alternative products, increasing the likelihood of purchase. Example: "Customers who bought this product also bought..."
- Reducing cart abandonment: Making appropriate product recommendations encourages customers to finish their purchases.
- Filtering and guiding the user through the huge number of products: Without recommendations, the user can get lost among thousands of products.

- Leveraging data and turning it into value: E-commerce sites have a treasure trove of data. Recommendation systems are what give it meaning and turn it into profits [9,10].

Prominent real-world examples of the use of recommendation systems include:

Amazon: The platform relies on advanced recommendation algorithms to analyses user behavior and purchasing history, suggesting relevant products through sections such as "Frequently Bought Together" and "Customers Who Bought This Item Also Bought." Both collaborative filtering and content are used to provide effective personalized recommendations [39].

Alibaba: It uses machine learning techniques to match users with products they may be interested in, based on clicks, browsing history, and even user behavior in the current session. Geographic and temporal data are also used to improve recommendation results [40].

Netflix (although not a commerce platform, but a practical example): It provides movie and series recommendations to users based on their previous viewing history and other users' interactions [41].

Noon and Jumia: These platforms combine general recommendations (bestsellers) with personalized recommendations based on user history, increasing the likelihood of a visit converting into an actual purchase [42].

1.7 Conclusion

With significant advances in technology, scientific research, and infrastructure recommendation systems have evolved rapidly, moving from simple methods based on similarity measurements to the use of machine learning techniques and then to more advanced models such as deep learning. From a business perspective, both users and organizations are seeking personalized and immediate recommendations, requiring intelligent and scalable systems that can respond quickly and effectively to product diversity and user density.

Despite this advancement, modern recommendation systems still face real challenges that impact the quality of the user experience. One of the most prominent of these challenges is the "filter bubble" phenomenon, where algorithms tend to display the same type of content a user has previously interacted with. This reduces diversity, limits discovery opportunities, and negatively impacts long-term user satisfaction. In the next chapter, we will discuss this problem in detail, examining its causes, effects, and the most important methods used to mitigate its impact.

Chapter 2

Filter Bubble Effect in Recommender Systems

2.1 Introduction

Recommender systems face a number of technical and cognitive challenges that directly impact the quality of recommendations and user satisfaction. These challenges include data sparsity, where the system doesn't have enough information to learn from; cold start problems, especially with new users or products; scalability, where the database grows and the system is required to respond quickly; and the difficulty of balancing diversity and relevance in recommendations. However, among all these challenges, filter bubbles stand out as one of the most serious problems users faces. This phenomenon occurs when a system consistently recommends the same type of content based on a user's past behavior, thus isolating itself within a "bubble" of repetitive recommendations, unable to discover new or differentiated content. A simple example of this is in the field of e-commerce: if a user is interested in fitness products, the system will only recommend products of the same type, marginalizing other categories such as books or clothing, even though the user may be interested in them. This limits the user experience and even impacts platform sales. Therefore, the design of recommendation algorithms, especially in real-time systems, can either help reduce these bubbles or deepen them, depending on how they balance personalization with openness to diversity. In this work, we developed a hybrid recommendation engine that operates in real time.

2.2 Definition of filter Bubble Effect

A filter bubble is a state of informational isolation created by algorithms that personalize content based on a user's past preferences and behavior. The concept was first introduced by Eli Pariser in 2011, who described it as "a user's personal world," where information that doesn't align with their pre-existing interests is blocked out, reinforcing cognitive bias and reducing openness to different perspectives. In the context of recommendation systems, a filter bubble occurs when algorithms rely excessively on a user's past interaction data—such as clicks, views, or purchases—without considering content diversification or incorporating new elements. As a result, the same type of content is repeatedly presented, limiting users' exploration of new topics or potentially valuable alternatives. This algorithmic isolation can weaken the overall user experience, reduce diversity, and, in some cases, reduce long-term user satisfaction or engagement [11][12][21].

2.3 Consequences of the Bubble Effect

- **Reduction of the horizon (Reduction of the horizon)**
  - Restrict the user's ability to discover new topics or content outside of their current areas of interest.
  - The end result is that the user is left with limited options.
  - Leaves users' easy prey to fake news.
- **Limited Diversity (Limited Diversity)**
  - Based on submitted preferences, the systems primarily recommend similar content.
  - Outcome: Less intellectual and cultural diversity, which may lead to boredom or an increased risk of illness [13].

2.3.1 Can we increase the effect of filter bubbles?

If the system is served by an algorithm that depends only on the user's current behavior (videos seen, clicks, etc.) without diversifying it? The channels used are:

- ✓ Collaborative Filtering
- ✓ Content-Based Filtering

The result is that the user always sees the same type of content. The system is wrapped around the user's preferences. There is no exploration.

- For example: a person who loves cartoons, the system sends him videos of the same type directly after each viewing, so he enters a bubble [8].

2.3.2 How to reduce the effects of filter bubble?

The system is served by an algorithm that combines exploitation and exploration even in real time. The techniques used to reduce the bubble are:

- Multi-Armed Bendit balances between providing known content that the user likes and new content for him.
  - Example: Provide 80% of what the user likes and 20% new, unexpected content.
- Diversification Techniques: Forcing the system to choose diverse recommendations in topics or sources

  - for example: videos from different sources, even if they are far from the user's previous preferences.
- Context-Aware Recommendations rely on context (location, time, device) to deliver different content, even if the user's general user behavior is inclined toward a certain type.
  - For example: In the morning, they recommend news, in the evening, they switch to entertainment, even if you prefer entertainment.
- Temporal Decay gives less importance to old activities and more importance to new ones.
  - Example: A recommendation for the last thing you saw, not your entire history [9].

From what we've seen, it's clear that the system can increase or decrease the bubble effect depending on how it serves its users. By relying solely on what the user likes, it will repeat the same content and trap them in a vicious cycle. However, if there's a balance between the content they like and new content, it can change the atmosphere and make them discover things they never expected. In the next section, we'll explore how we can build a system with this balance, one that always produces the same type of recommendations, and thereby mitigate the bubble effect in a simple and effective way. In next section, we detail approaches which can mitigate filter bubble effect.

2.4 Approaches to mitigate Filter Bubble

In Recent research in the field of recommender systems has proposed a number of approaches that aim to mitigate the filter bubble effect and achieve a balance between personalization and diversity. Among these approaches are [13]:

2.4.1 Hybrid Recommendation

The database stores the user, the product and the user's interactions, and when the user browses or reacts, the system records the reaction (clicking on the product). This data is later used to build recommendations based on:

- Collaborative Filtering (based on similar user interactions).
- Content-Based Filtering (based on product features).

2.4.2 Real-Time Recommendation Based on Dynamic User Preference Updates

- **Monitor user interaction**

  Continuously track user actions such as clicks, views, and purchases during a session.
- **Session Data Logging**

  Instantly record each mew interaction and associate it with the current session.
- **Feature Extraction**

  Analyze interacting products to extract features such as category, brand, and keywords.
- **Update dynamic preferences**

  Instantly edit and reorder recommendations based on updated preferences.
- **Recommendation Refresh**

  "Recommendation Refresh" pertains to the activity of instantly updating and regenerating recommendations based on live changes in user behaviors or preferences occurring in a single session of interaction with the recommender system. Recommendation refresh is an important aspect of real-time recommender systems as it enables a real-time interaction aspect so that the functioning of the recommender system remains adaptable to a user's current orientation. (in this project we use this type of recommendation)

  How does it work?

  As soon as a user performs any new action (e.g., a click, view, or purchase), the system:

  - **Analyses the interaction** and updates the session data in real time.
  - **Recalculates the preference weights** using the most recent information.

- **Reorders the list of recommendations** based on these updated preferences.
- **Displays a refreshed set of recommendations** or reorganizes the current list to reflect the user's new interest.

- **Hybrid Model Reweighting**
  Rebalance collaborative and content-driven output to reflect current user interest.
- **Real-Time Feedback loop**
  Repeat this process with each new action to keep the recommendations consistently relevant.

In a real-time recommendation system, recommendations are constantly changing based on current user behavior. This can help reduce the filter bubble effect if the system is designed to account for diversity and exploration. However, if the system remains solely based on current and past user interactions without diversification, it can further reinforce the filter bubble.

We find also an algorithm used for diversification to mitigate filter bubble.

2.4.3 Algorithm Epsilon-Greedy

The Epsilon-Greedy algorithm can help to break a filter bubble because at each time we explore, we'll encounter content that's different from what we used to. This allows to break out of the bubble and explore new topics. A small amount of exploration (say, 10%) is enough to

Steps for using the Epsilon-Greedy algorithm in recommender systems

- Analyzing user interactions: The system records the content that the user interacted with (e.g., movies they watched, articles they read...).
- Calculating the expected value of each item (Estimated Reward) Items (movies, products, videos...) are evaluated based on how much previous users liked them.
- Generate a random number between 0 and 1. This number determines whether the system will recommend new or expected content.
- Decision (content selection): If the number $< \varepsilon$ (e.g. 0.1): Exploration → The system suggests new or unusual content, outside the user's preference range.
- If $\geq \varepsilon$: Exploitation → Suggests content based on past interaction history.

- Record new interactions. The system records what the user chooses to update recommendations in the future [14].

Gradually create diversity in the experience without negatively impacting recommendations [14].

The decision-making process of the Epsilon-Greedy algorithm can be summarized in the following flowchart.

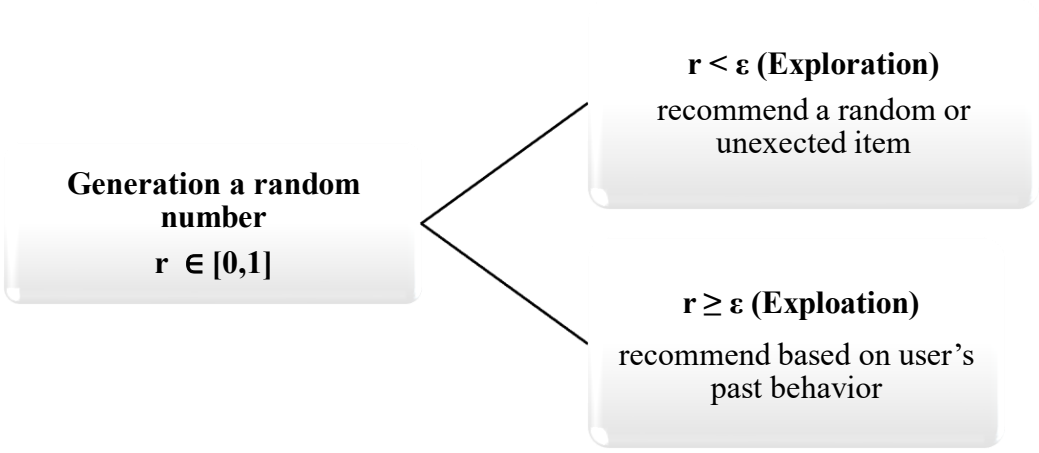


Figure 6 Algorithms Epsilon-Greedy Flowchart

Using Epsilon-Greedy, the system can balance providing accurate and relevant recommendations with discovering new products that increase the diversity of the user experience, gradually reducing the filter bubble effect.

Discussion

As we saw in the previous section, the negative impact of recommendation algorithms on the filter bubble phenomenon depends largely on the algorithm's internal design and the extent to which it relies on the principles of diversity and exploration. Therefore, it becomes necessary to propose algorithms that combine multiple techniques and balance existing user preferences with new diverse content. In this context, we developed a hybrid engine that aims to break the filter bubble by combining traditional recommendation techniques with real-time exploration and diversity methods.

2.5 Conclusion

In this chapter, we highlighted the filter bubble phenomenon as one of the most critical challenges facing modern recommender systems, particularly those relying solely on users' past preferences. We examined its underlying causes, its negative impact on content diversity and user experience, as well as several mitigation strategies proposed in the literature. This analysis sets the foundation for the next chapter, where we introduce our own approach to addressing this issue through a real-time hybrid recommender system.

Chapter 3

Our Approach

3.1 Introduction

In this chapter, we present our proposed approach for building a personalized recommendation system for a multi-product e-commerce platform. This system aims to improve the user experience by promptly and accurately suggesting relevant products, while minimizing the filter bubble phenomenon caused by duplicate and inconsistent recommendations.

We chose to adopt a Real-Time Hybrid Recommender System, which combines collaborative and content filtering technologies and is based on tracking user interactions in real time to update recommendations based on changing user behavior.

This system is designed to interact with users in real time, providing personalized, diverse, and contextually relevant recommendations, helping to enhance user satisfaction and increase conversion rates within the platform.

3.2 Idea

Our idea is to develop a real-time recommendation system for a multi-product e-commerce platform, with the goal of improving user experience and increasing engagement and sales. The system relies on a sequential hybrid approach that begins with collaborative filtering and then moves on to content-based filtering to ensure personalized and effective recommendations.

To mitigate the filter bubble effect two complementary techniques were adopted: diversification in the offline phase:

- To provide diverse and non-repeating recommendations;
- Real-time recommendation mechanism that responds to changes in user behavior immediately, allowing for continuous updating of recommendations.

The approach is divided into two main phases:

Phase1

The recommendation engine is constructed offline using a cascade hybrid architecture. This phase includes the training of the selected algorithms, the evaluation of their performance using standard metrics, and the preparation of the model for deployment within the system.

Phase 2

The trained model is then deployed online within the e-commerce platform, where it begins to generate real-time recommendations based on continuous user interactions during their session.

3.3 Data collection

To train and test the model, we relied on a popular dataset known as Amazon Product Data, one of the most widely used databases in the field of recommendation systems. This data is available via the link: https://www.kaggle.com/datasets/karkavelrajaj/amazon-sales-dataset

We chose this database for several reasons, most notably:

- Its large size and diversity, which enable modelling realistic scenarios in the e-commerce environment;
- Its availability of real user interactions with products (reviews, ratings, time of purchase, etc.);
- Its inclusion of detailed product characteristics that can be exploited in content-based recommendation algorithms.

Data used includes:

- User data: such as user ID, preferred category, past activity, etc.
- Product data: such as product ID, category, brand, price, description, color, size, etc.
- Interaction data: includes user reviews and ratings, as well as the date and time of interaction (timestamps), which are used to track real-time changes in preferences.

This data is initially processed through stages of cleaning, formatting, and transformation into a format usable by recommendation algorithms. This aligns with the second stage of the recommender system construction cycle, as described in (figure 1, chapiter1) [23].

3.4 Algorithms

The suggested recommendation system relies upon a cascade hybrid system which intends to blend the accuracy of collaborative filtering and the granular personalization of content-based filtering. It would accomplish this in two fundamental phases:

- **Collaborative Filtering**: In this step, we constructed a User–Item Matrix, where each cell represents a numerical rating indicating the user's level of interaction with a particular product. To process this matrix, we used the Alternating Least Squares (ALS) algorithm, a popular matrix factorization recommendation algorithm known for its effectiveness in handling sparse data.

  The ALS model learns latent factor representations for both users and products, allowing it to predict any unknown ratings and to suggest personalized products. Once the model is trained, we have an initial recommendation list of the K products for each user based on each user's historical interaction with the products.
- **Content-Based Filtering:** After obtaining the initial recommendations from the ALS algorithm, we applied a second filtering stage aimed at re-ranking the results based on their similarity to the user's content preferences.

  In this stage, the product description (about product) was analysed and its textual features were extracted using the Term Frequency–Inverse Document Frequency (TF-IDF) technique, which provides a numerical representation of text that highlights the most distinctive words. We then calculated the similarity score between the product representation and the user profile using Cosine Similarity, where the user profile was constructed from the average vector of products they had previously interacted with.

This ranking ensures that the displayed products are not only predictable based on the behavior of others, but also aligned with the user's specific content interests, improving the recommendation experience and reducing the filter bubble effect.

To improve recommendation, we adopted an architecture called cascade hybrid engine for recommender system. Figure 5 shows our cascade hybrid engine for recommender system.

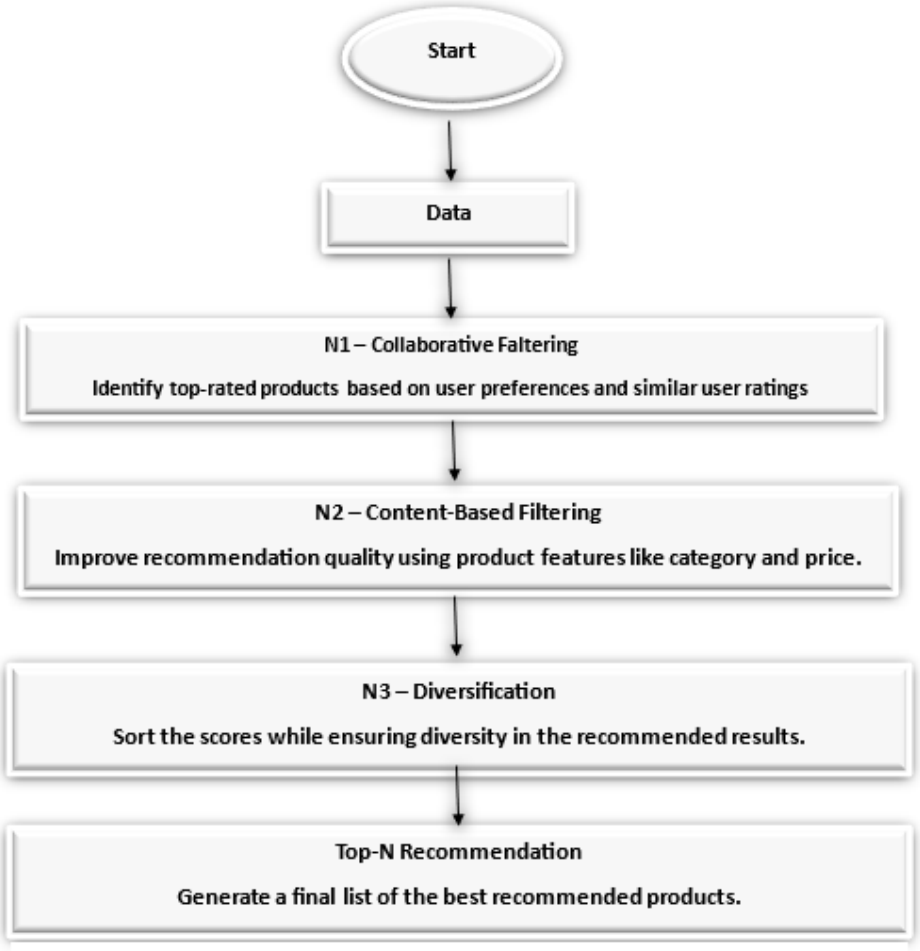


Figure 7 Cascade Architecture of hybrid engine recommender system

3.5 Model

The proposed model was built in two integrated stages:

Offline Mode

At this stage, we tested and refined the recommendation algorithms using previously stored data. The models were trained and their results analyzed against performance metrics, with the goal of developing a robust prototype that could later be used in a live environment.

Online Mode

After the model was evaluated and adjusted, it was implemented to operate in a real-time manner in the online platform. The system has real-time interaction with the user; it monitors user behavior as they go through the platform and updates recommendations automatically. The user interface and the recommendation engine are linked with application programming interfaces (APIs), which allows to providing personalized recommendations as the user Clicks through the platform [23].

3.6 Offline testing

After implementing the proposed algorithms on stored data, the performance of the recommendation system was evaluated using a set of common metrics: accuracy, recall.

3.7 Online Testing

Online Testing After training, we deployed and tested the model on real users within the e-commerce platform.

We tracked a set of metrics:

- click-through rate (CTR).
- number of interactions with recommendations.
- time spent on the page, and conversion rate.

The results showed that the hybrid system significantly improved click-through rates compared to the single-method systems (collaborative or content-only).

3.8 Conclusion

By the end of the recommendation system development cycle, we were able to build a hybrid system that combines collaborative filtering and content-based filtering, supported by a

real-time recommendation mechanism. This system was specifically designed to meet the needs of a multi-product e-commerce platform.

By combining the two methods and Analyzing user interactions in real time, we were able to:

- ✓ reduce the filter bubble effect by diversifying the content presented to users;
- ✓ improve the quality of recommendations and increase user satisfaction;
- ✓ provide more accurate, relevant, and faster suggestions.

This strategy arguably represents a helpful stride toward the intelligent, personalized, and routinely updated recommendations that will improve the user experience, and potentially raise engagement and sales revenue on the platform.

Chapter 4

Implementation of our approach

4.1 Introduction

Having established the theoretical foundations of recommender systems (Chapters1 and 2) and reviewed our proposed approach (Chapter 3), we now turn our attention in this chapter, to the practical side of the project. This chapter describes the implementation of the system - how the proposed hybrid recommender system was developed, tested, and operated in the testing environment of an e-commerce system.

The implementation is divided into two phases:

Phase 1 (Offline): This stage encompasses building the recommendation model and evaluation of the model, using collaborative filtering along with content-based filtering. Here the focus is on both accuracy and diversity in the recommendations, in order to mitigate the filter bubble impact.

Phase 2 (Online): In this phase, we integrate the trained model to an interactive e-commerce site. We can track these user actions in real-time and update the recommendation engine dynamically based on the evolving needs of the user.

This chapter presents a summary of the tools and technologies implemented in both phases as well as an analysis of the results obtained. Next, we detail how recommended interventions are displayed and updated through the interface of the platform to demonstrate the end-user's perspective on the recommendation in real-time.

4.2 Tools and libraries Used to implement our approach

In this section we identify the main tools, libraries and technologies used to develop our recommendation system, including both the offline development and online deployment.

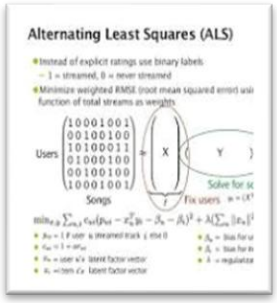
4.2.1 Tools libraries used for Phase 1

TF-IDF (Term Frequency–Inverse Document Frequency): Used to extract meaningful textual features from the product descriptions to assist in content-based filtering [27].

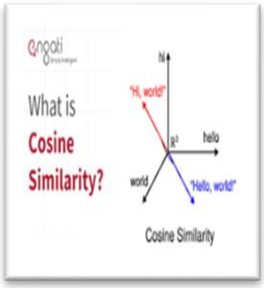
Python: The primary programming language employed based on its extensive collection of libraries and ease of use for data processing and machine learning tasks [22].

PySpark: Considered the primary framework for large-scale data processing as well as building recommendation algorithms. PySpark is especially useful for big data in a distributed environment [24,31].

NumPy and Pandas: Used for the first-stage of data preparation, manipulation, and transformation before the data goes into the recommendation pipeline 25].



ALS (Alternating Least Squares): Used PySpark's ml. recommendation module to perform collaborative filtering by factorizing the user-item interaction matrix [26].



Cosine Similarity: Used to measure the similarity between the user profile and product content vectors produced from TF-IDF for providing content-aware, personalized recommendations [28].

Visual Studio Code is a lightweight yet powerful source code editor that runs on your desktop and is available for Windows, macOS, and Linux [29].

4.2.2 Tools for libraries Phase 2

Django: Utilized as the primary web framework as the overarching framework for the e-commerce platform (also known as "online storefront") for managing the user interface and governing the user flow of interactions [30].

Interaction Tracking System: This system was built natively in Django for the purpose of logging user behaviors (clicks and views) as they occurred in the session, without the use of third-party tracking tools [32].

Bootstrap (optional): A CSS framework designed to ensure responsive and user-friendly interface design [33].

HTML, CSS, JavaScript: Used to design the frontend interface of the e-commerce platform where users interact with product recommendations [34].

Database (Sqlite3): Used to store product information, user data, and recommendation logs [35].

These items were selected for their efficiency, ease of integration, and the support provided by the data science and web development communities. Together they allowed us to create a modular, scalable, and responsive recommendation platform.

4.3 Hardware used to implement our approach

The development, testing, and deployment of the proposed hybrid recommender system were carried out on a single local machine. Although modest in terms of modern hardware capabilities, this setup proved to be sufficient for handling both the offline modelling phase and the online deployment phase of the system. The specifications of the machine used are as follows:

- **Processor: Intel® Core™ i5 (4th Generation):** Processor: Intel® Core™ i5 (4th Generation)

  The Intel Core i5 processor of the 4th generation provided a balanced performance for running both the Django web server and computationally demanding PySpark algorithms. While it does not match the speed of newer-generation CPUs, it was capable of supporting parallel data processing for small- to medium-scale experimental datasets.
- **Memory (RAM): 8.00 GB:** The 8 GB of RAM facilitated the normal execution of standard development activities, such as loading datasets, extracting features with TF-IDF, and calculating similarities. It also managed to perform basic Spark operations during the offline training phase with no apparent memory bottlenecks.
- **Storage: 320 GB Solid State Drive (SSD);** Adding an SSD resulted in faster data read/write speeds, which led to more efficient dataset preprocessing and application boot-up periods. This is especially relevant with local development servers, and loading recommendation models.
- **Operating System: Microsoft Windows 10 Education (64-bit);** The system was configured with a 64-bit version of Windows 10 Education, so it could run the required development tools for the project. This included Python, PySpark, and the web framework, Django, as well as the capacity to incorporate visualization tools and also test the web interface using the browser.
- **System Type: 64-bit Operating System:** A 64-bit system architecture is important for running modern data science libraries using large memory addressing and efficient matrix multiplication (some recommendation algorithms like ALS or TF-IDF similarity rely on scalable memory and computing).

This hardware environment was sufficient for developing the prototype system, testing its core functionalities, and ensuring its responsiveness during user interaction simulation. However, it is worth noting that for real-world deployment in a production environment especially with large-scale datasets and concurrent users a more powerful server with distributed processing capabilities would be recommended.

4.4 Results Presentation (Offline Phase)

In order to better understand how our hybrid recommender system behaves during a user session, we present a step-by-step breakdown of how it generates and updates recommendations. This analysis highlights when and how **diversification occurred**, and how the system managed to **overcome the filter bubble effect** by balancing personalization with content discovery in real time.

1. **Testing and interpretation results of model for all Users**

We test our model on 20% of data set. Figure 6, 7 show result of accuracy and recall respectively.

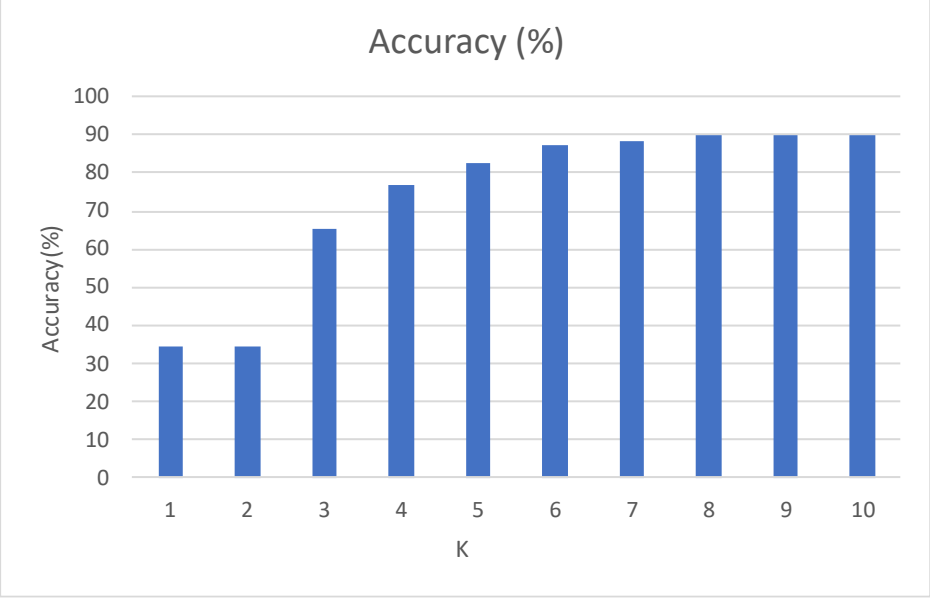


Figure 6 Accuracy (%) vs. Number of Neighbors (K)

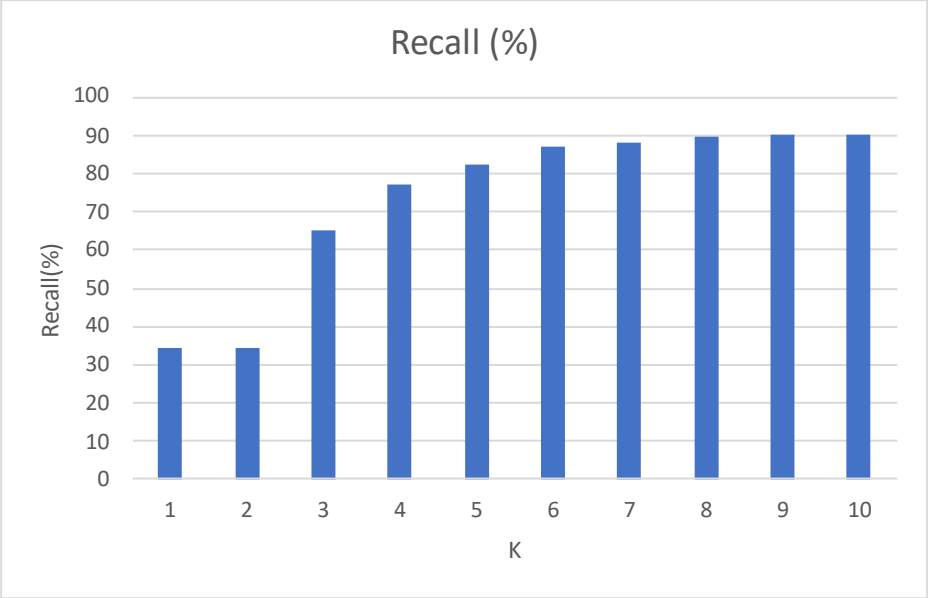


Figure 7 Recall (%) vs. Number of Neighbors (K)

By Analyzing the results presented, which highlights the system's performance metrics (Accuracy and Recall) across different values of K (number of recommendations), we can observe how the system's effectiveness evolved over time. This analysis reflects the hybrid model's ability to adapt, offering a balance between relevance and diversity in recommendations:

- **Accuracy remained consistently high**
  - Despite the increase in the number of suggested items, the accuracy did not drop.
  - This means that diversification was not random, but rather well-targeted and aligned with the user's real interests.
- **Consistent improvement in Recall**
  - Indicates the system's ability to retrieve an increasing number of relevant items as the number of recommendations (K) grows.
  - This suggests that the system does not limit itself to repetitive content, but effectively expands the scope of exploration.

The results for **Accuracy and Recall** clearly show that the system improved gradually and intelligently. With every increase in the number of recommendations, it managed to strike an ideal balance between quantity and quality, offering diverse products that are still relevant to the user's preferences. This reflects the strength of the adopted **hybrid model** and its ability to respond in real time to changes in user behavior without sacrificing accuracy or relevance.

Furthermore, the performance metrics confirm this evolution: Recall improved steadily, indicating broader and more inclusive recommendations, while Accuracy remained stable, proving that this diversification did not compromise the relevance or quality of the results.

2. **Testing and interpretation results of model on selected User's historical Behavior**

To evaluate our approach, we select a user from dataset with random manner. The user selected is UserID=AGD5KTBDTS26I2SB3B7LCYBR6U3A, the system begins by Analyzing past interactions:

- Previously viewed or purchased item
- In this case: multiple interactions with **Redmi Note 11T 5G smartphones**.

The system initially uses collaborative filtering (ALS) to generate recommendations:

- It suggests the same smartphone model in different colours (**Stardust White, Aquamarine Blue, Matte Black**).

✓ No diversification yet — the system is still relying purely on past behavior. This is a sign of a potential filter bubble (repetitive recommendations).

3. **Content-Based Re-ranking**

Next, the system applies content-based filtering using TF-IDF and cosine similarity:

- It analyses product descriptions and builds a user profile based on preferred keywords/features (e.g., "fast charging," "USB-C," "Android").

Now, the system realizes that the user may also be interested in **accessories**, not just smartphones.

✓ First level of diversification (Horizontal Diversification):

- The system begins recommending **Amazon Basics USB-C charging cables**.
- These are still tech-related but belong to a **different product category** (Accessories > Cables).

4. **Real-Time Interest Shift Detection**

As the session continues, the system dynamically tracks what the user is interacting with:

- The user spends more time on general electronics pages.

- Shows subtle interest in broader display/audio-related products.

Second level of diversification (Vertical Diversification):

- The system recommends **MI Smart TVs** (32", 40", 43").
- A completely new product category → **from phones to smart televisions**.

✓ This shift is possible because the system incorporates **real-time re-ranking**, updating recommendations on the fly as user preferences evolve.

5. **Balancing Exploration and Exploitation**

At the end of the session, the system returns to suggesting **a Redmi Note 11T 5G**, alongside TV recommendations. This proves the system is:

- Still **exploiting known preferences**.
- But **not stuck in them** — it's actively **exploring related and new categories**.

From this we conclude that, this shows the system successfully broke the filter bubble, offering diverse yet relevant recommendations without sacrificing personalization.

The combined analysis of the system's behavioral flow and its performance metrics (Accuracy and Recall) demonstrates the effectiveness of the proposed hybrid recommendation approach. Throughout the recommendation process, we observed a gradual and intelligent diversification of suggested products—from repeated smartphone models to accessories and finally to smart TVs. This smooth expansion reflects a successful escape from the filter bubble effect, where users would otherwise remain trapped in repetitive suggestions.

Altogether, these outcomes confirm that our system not only avoided the pitfalls of repetitive filtering but also delivered a dynamic, diverse, and personalized recommendation experience—validating the success and impact of our work.

4.5 Results Presentation (Online Phase)

4.5.1 E-commerce platform implementation

After validating the hybrid recommendation model in the offline stage, the model was implemented into an e-commerce platform that was fully operational and designed using the

Django web framework. This stage represents the live application of the system by providing personalized recommendations through user engagement and interaction in real-time.

The main components of the platform are:

- **Account Creation:** Upon their first visit to the platform, the user will need to register for a new account before they will have access to all the functionality in the system. The registration will involve entering key information like their name, email, and password. This step is essential to create a personalized experience for the user and to track each individual's use of the application.

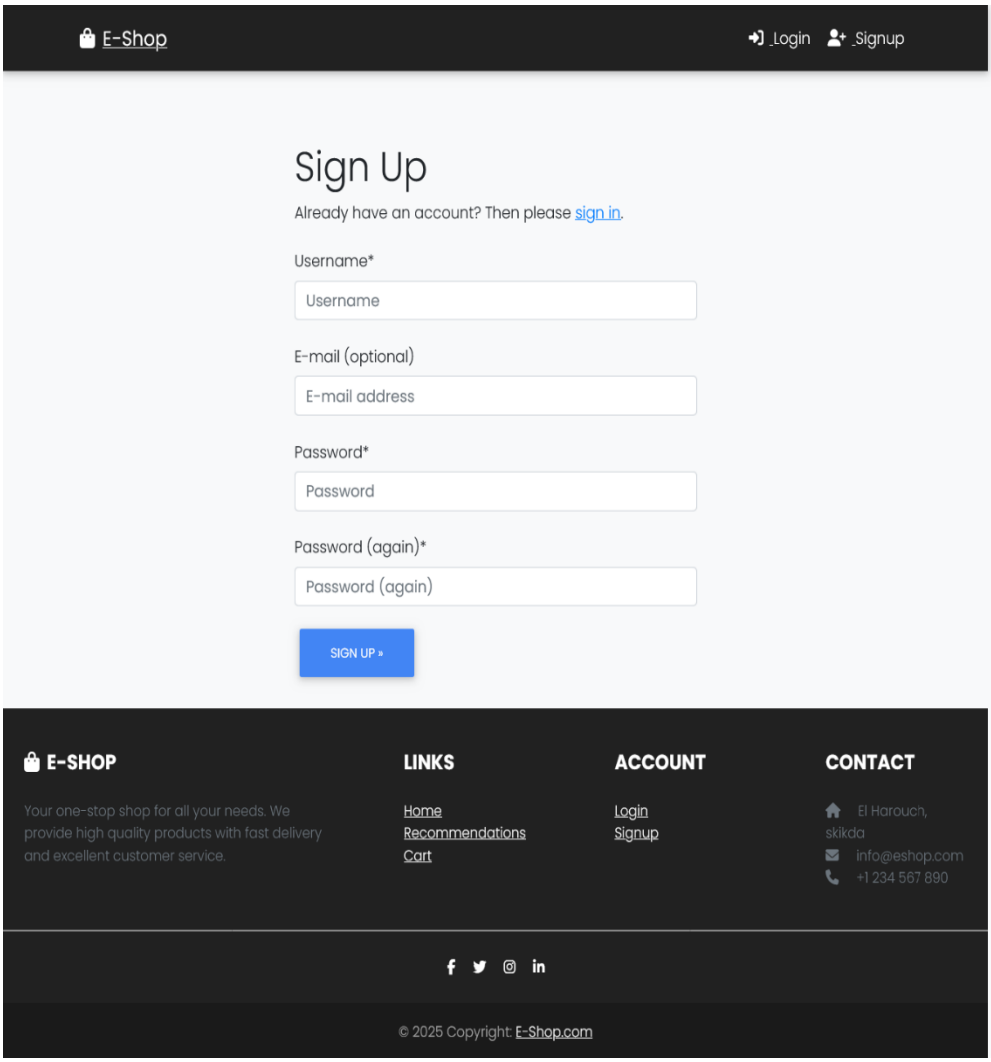


Figure 8 Create an account

- **Login;** After creating an account, the user must log in using their credentials to access their personal profile. Logging in enables the system to monitor individual user activity, such as browsing products, adding items to the cart, and completing purchases. This allows the recommendation engine to provide accurate and real-time personalized suggestions based on user behavior.

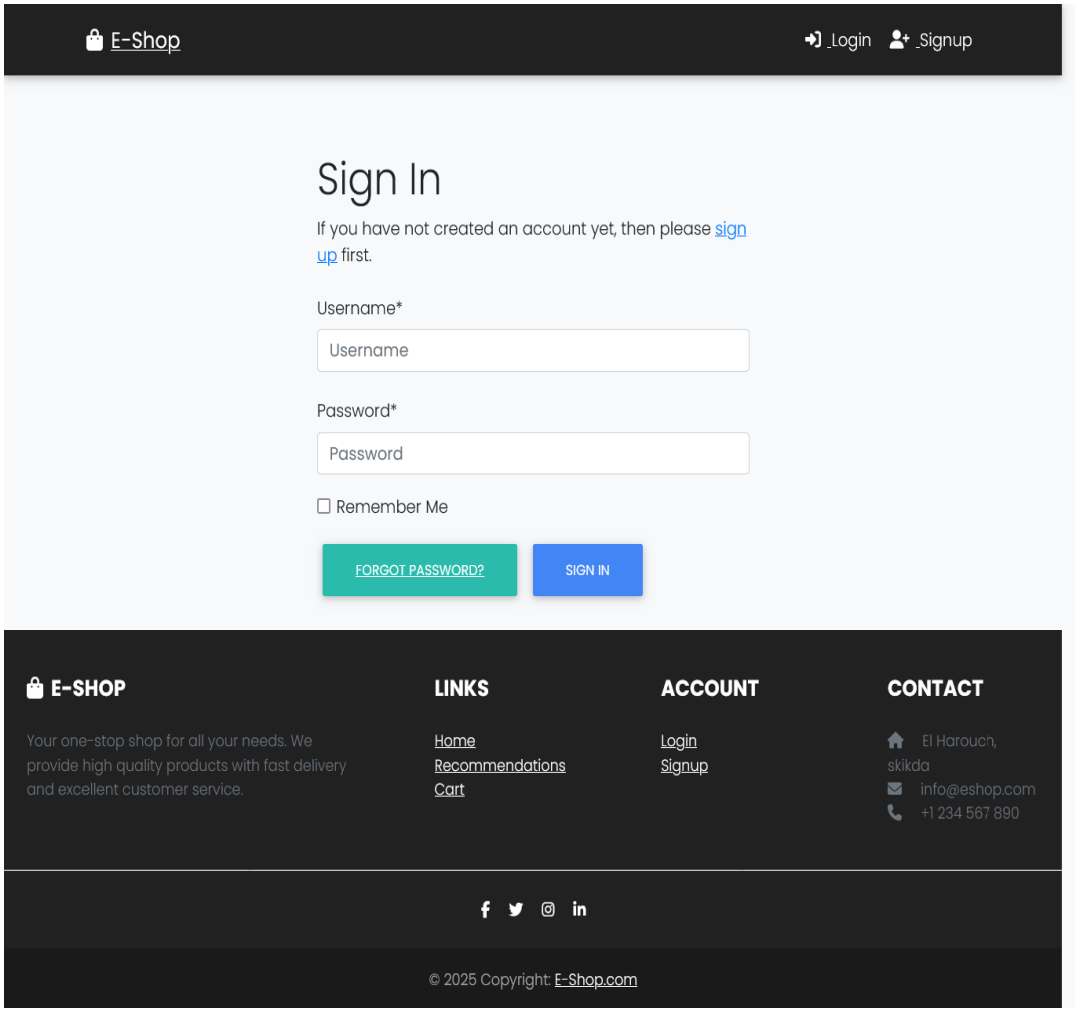


Figure 9 Login process

- **User Interface Design:** The site has two main views
  - **Product Listing View:** Shows all products filtered by type.

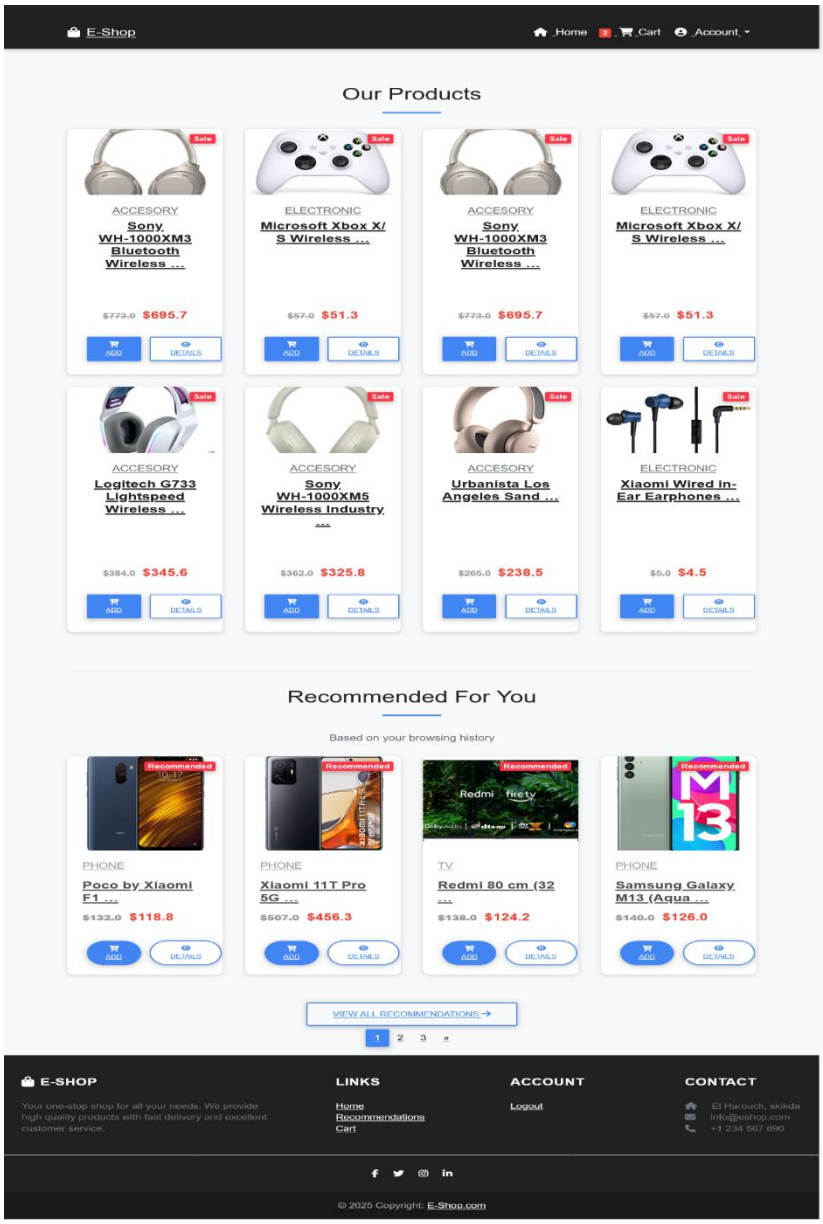


Figure 10 Product view

- **Personalized Recommendations View**: The displays real-time, dynamic user suggestions based on activity.

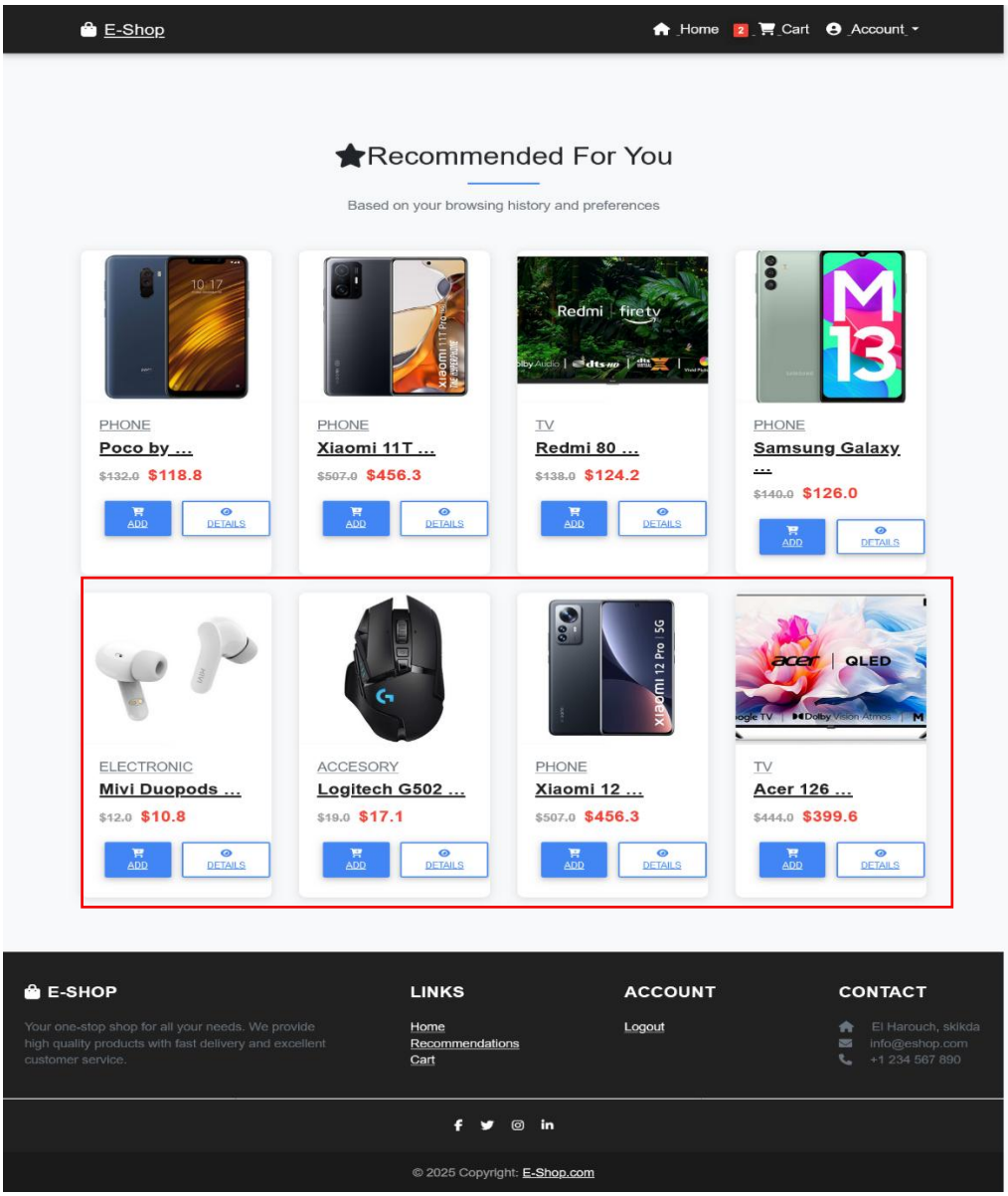


Figure 11 Personalized Recommendation

- **What users did:** After registration and login, the user can fully interact with the platform. A lightweight interaction tracking system was developed directly within Django to capture key user behaviors during each session, including:
    - **Viewing a product**: The user views a product or visits a product page.

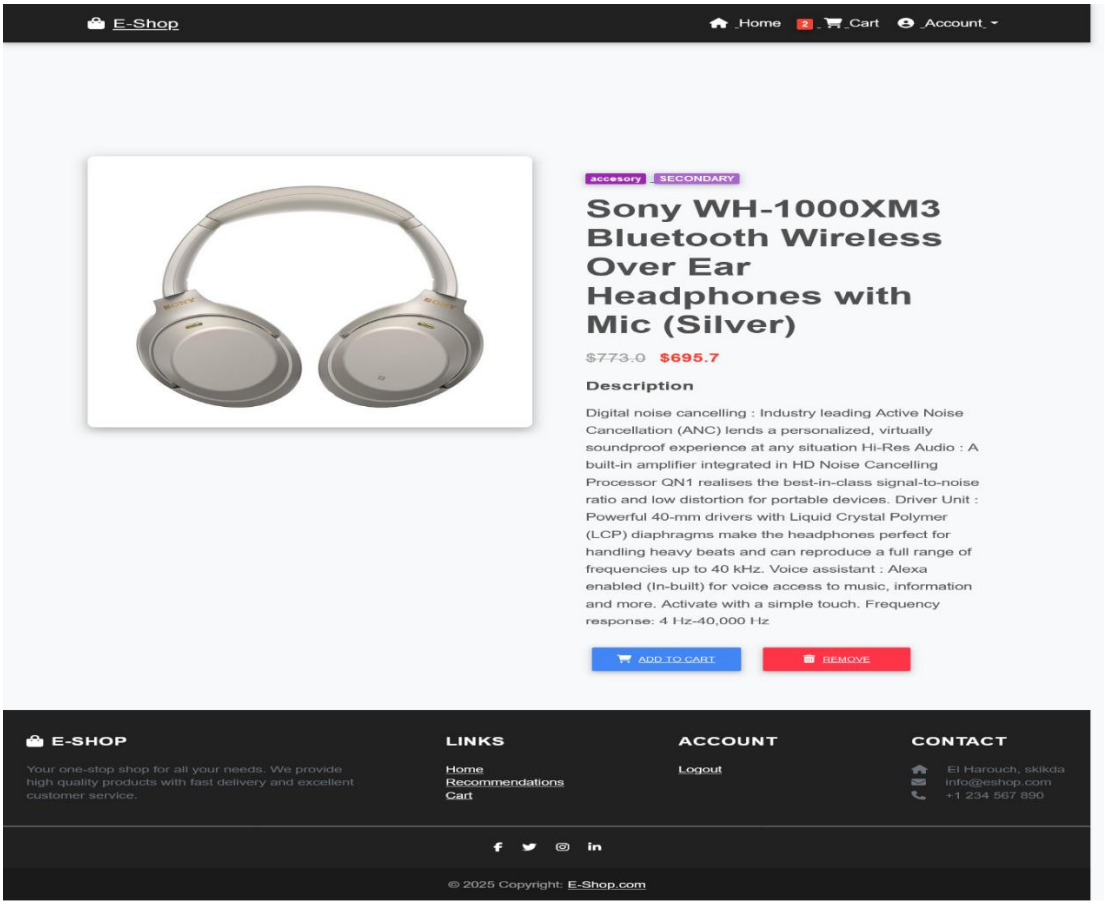


Figure 12 Viewing a product

- **Adding a product to the cart:** The user adds the product to their shopping cart or Wishlist.

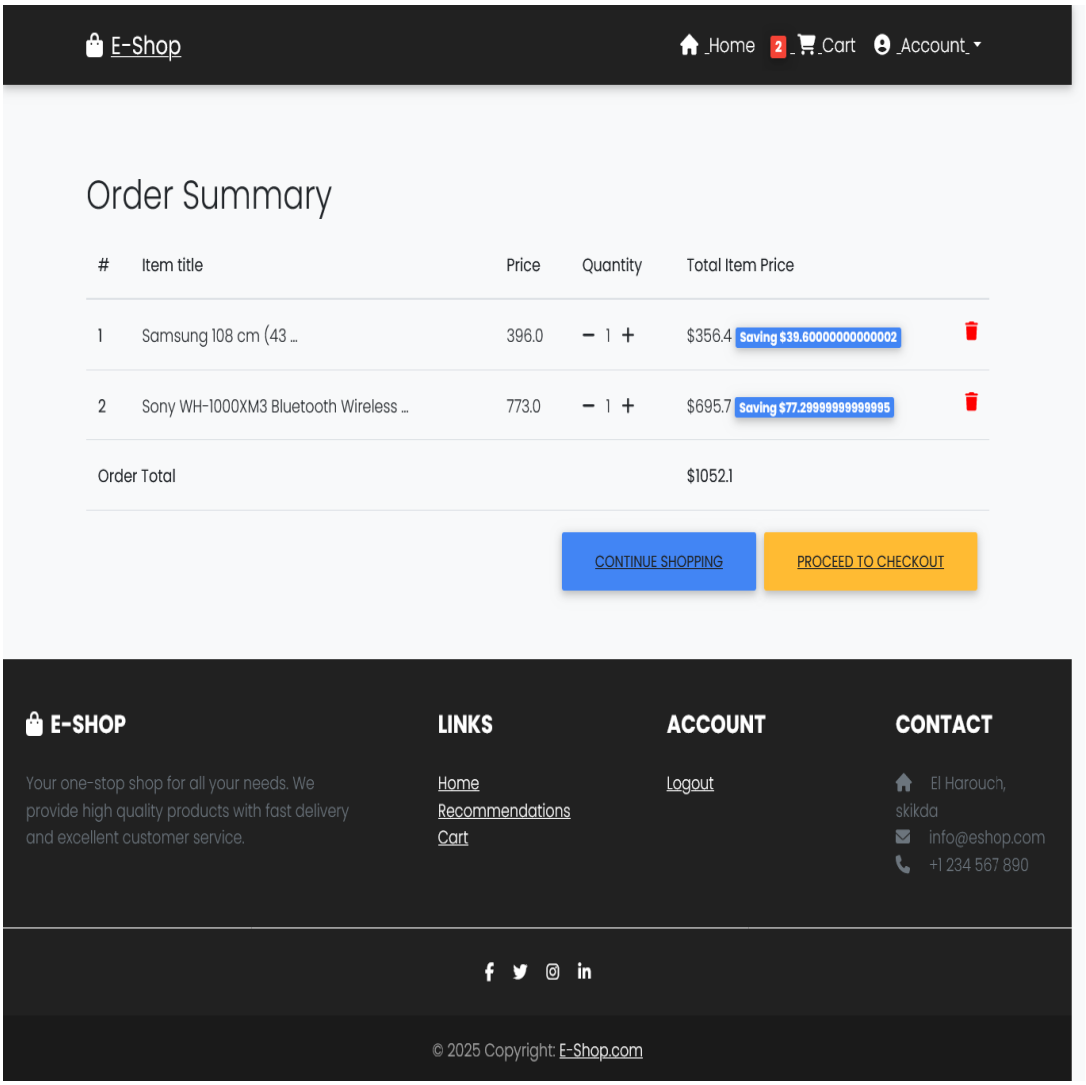


Figure 13 Add to cart

- **Completing a transaction:** The user completes the purchase and makes a payment.

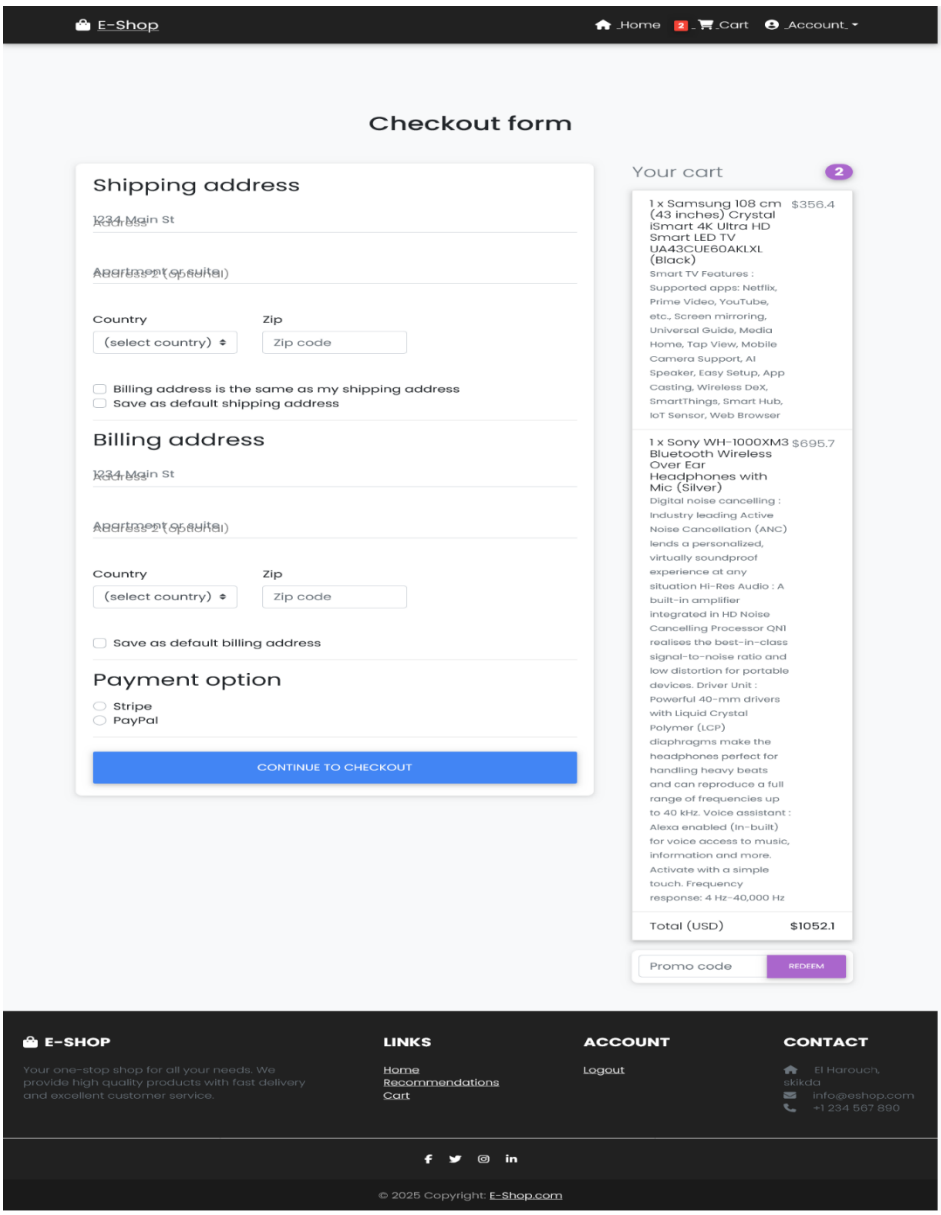


Figure 14 Payment and purchase process

- **Personalized Recommendations View**: Dynamic, real-time suggestions are shown to users based on their activity. These recommendations are continuously updated through the interaction tracking system built into Django, which captures key user behaviors during each session.
- **Model Embedded in the Platform:** The PySpark-based recommendation logic was incorporated directly into the Django server-side code. This allows recommendation

computations to run internally within the platform itself, simplifying deployment and minimizing communication overhead.

- **Real-Time Recommendations Updates**: As soon as the user takes any action on a product (e.g. add to cart), the platform should pick up on that behavior and make instantaneous recommendations updates without retraining the model. This also means that partially integrated the PySpark model logic within the Django backend.

4.6 Conclusion

In this chapter, we explored the practical implementation of the proposed hybrid recommender system, covering both the offline development phase and the deployment of the model into a live, real-time e-commerce platform.

During the offline phase, the model was trained and validated with a large-scale dataset collected from Amazon´s services. The results of the experiment demonstrated that cascade hybrid approach incorporating both collaborative filtering (exploiting ALS) and content-based filtering (exploiting TF-IDF and cosine similarity) were successful at producing personal recommendations with a decent level of diversity. The ability to bring diversity and balance between novelty and relevance is important to mitigate one of the critical areas in any recommendations, the filter bubble effect - when a user is continuously recommended the same pieces of content. Overall, the model showed the ability to break that pattern by recommending a mix of recommendations that included both known and exploratory products.

During the online deployment phase, the model was integrated into an e-commerce platform developed using the Django framework, with a user interaction tracking system built directly into the platform. This allowed recommendations to be dynamically updated as the user browsed, or when they performed an action such as clicking, adding to cart, or completing a purchase. This was accomplished without the need to retrain the model, making the system more responsive and flexible.

Real-time responsiveness has proven to be one of the system's most significant strengths. Unlike traditional models that rely solely on historical data, the system was able to understand the user's current intent and respond immediately by updating recommendations in line with their

current behavior within the session. This has led to a significant improvement in the accuracy of recommendations, increased user satisfaction, and an increased likelihood of engagement or purchase.

Most importantly, the combination of a cascade hybrid model and real-time behavioral updates allowed the system to effectively address the filter bubble problem, since users were no longer trapped in a narrow band of recommendations based on only their past behavior. Users were shown a wider and more diverse set of options—aligned with preferences, but still promoting content discovery. The intelligent design of the interface was able to successfully balance personalization and diversification—meaning that, the system addressed the limits imposed by algorithms, while also bringing richer and fairer user experience.

In conclusion, the results achieved in this application phase confirm the effectiveness of the proposed hybrid solution and demonstrate that recommendation systems, when built on flexible and intelligent foundations and leveraging real-time interaction, can overcome one of the most complex problems in modern recommendation—the filter bubble—and deliver a more equitable, diverse, and satisfying user experience.

General Conclusion and perspectives

In this thesis, we addressed the growing importance of recommender systems within the rapidly expanding domain of e-commerce, focusing specifically on the filter bubble phenomenon a challenge that reduces content diversity and, over time, may negatively impact user satisfaction and engagement.

We started by investigating the theoretical frameworks for recommender systems, including collaborative filtering, content-based filtering, and hybrid systems, examining their strengths and weaknesses. In doing so, we pointed out how recommender systems that rely solely on users' historical activities are at risk of perpetuating or exacerbating filter bubbles and digital isolation by reinforcing the exposure to repetitive content.

To deal with this issue, we proposed and implemented a real-time hybrid recommender system built on a cascade architecture. Our architecture contained: collaborative filtering based on ALS model, content-based filtering based on TF-IDF, and cosine similarity, diversification, and real time user interaction processing; therefore, the recommendations provide immediate recommendations from those influences in the same session, quickly providing a personalized recommendation experience.

The system was tested in two phases:

- In the **offline phase**, we trained and evaluated the model using a real-world Amazon dataset. The results showed notable improvements in key performance metrics such as precision, recall, and recommendation diversity.
- In the **online phase**, the trained model was deployed within a fully functional Django-based e-commerce platform. It demonstrated its ability to update recommendations in real time as the user interacted with products, showcasing strong responsiveness and personalization.

Most importantly, the system successfully mitigated the filter bubble effect. By delivering a balanced mix of familiar and exploratory content, it broke away from the pattern of narrow, repetitive suggestions and encouraged content discovery. This led to an enhanced user experience, increased engagement, and a more dynamic interaction with the platform.

Based on the outcomes of this project, several precise and forward-thinking directions can be proposed to enhance the capabilities, fairness, and adaptability of recommender systems. These include:

- ✓ **Application of Generative AI Models; u**sing large language models (LLMs) such as GPT or BERT can provide improved insights and/or comprehension of product content and user reviews. It can increase the quality of user profiles and even allow for context-aware personalized recommendations in natural language. These models can continue to improve the user experience and overall intelligence of the system.
- ✓ **Enhanced Contextualized Recommendations**; Adding the dependence of real-time context such as time of day, user's location, type of device being used, weather, or events happening at the same time would allow for context-relevant recommendations and improve the relevance and timing of recommendations.
- ✓ **User Personality-Adaptive Recommendation Models**; the systems of the future could use personality insights or profiles of interest and therefore include additional sources of information beyond behavioral interactions (to the extent the data is being collected). Users can have some dynamic control over the exploration-exploitation ability of the recommendation system that therefore can provide recommendations consistent with a user's user type (i.e., someone who likes novelty vs. someone who looks for regularity).
- ✓ **Fairness or Ethics-based Recommendation Designs; i**mposing quotas and/or fairness constraints that detect and diminish algorithmic bias (commercial, demographic, cultural, etc.) barriers can improve access to content in a way that does not continually reinforce the (narrow) feedback loop or reinforce discriminative systems.

References

[1] Recommendation System, *nvidia.com*, [on line] available at: https://www.nvidia.com/en-us/glossary/recommendation-system/. [Accessed on: 17/6/25]

[2] LI, Ruiheng, SHU, Yuhang, CAO, Yue, *et al*, "Federated cross-view e-commerce recommendation based on feature rescaling", *Scientific Reports*, 2024, vol. 14, no 1, p. 1-19.

[3] DONG, Zhenhua, WANG, Zhe, XU, Jun, *et al.*A brief history of recommender systems. *arXiv preprint arXiv:2209.01860*, 2022.

[4] KAR, Pushpendu, ROY, Monideepa, et DATTA, Sujoy, Steps in Building a Recommendation Engine. In: *Recommender Systems: Algorithms and their Applications*. Singapore: Springer Nature Singapore, 2024. p. 101-111.

[5] NASIR, Mahreen et EZEIFE, Christie, "A Survey and Taxonomy of Sequential Recommender Systems for E-commerce Product Recommendation". *SN Computer Science*, 2023, vol. 4, no 6, p. 708.

[6] Pariser. E, "The Filter Bubble: What the Internet is Hiding", *UCLA Journal of Education and Information Studies*, 2012. available at: DOI 10.5070/D482011835

[7] Dansu. E.J, The Filter Bubble: Isolation from Diverse Perspectives. [Online] available at: https://www.linkedin.com/pulse/filter-bubble-isolation-from-diverse-perspectives-online-dansu-a8tmc/. [Accessed on: 13/6/25]

[8] ZHANG, S., LIU, K., YU, Z., *et al.* Hybrid recommendation system combining collaborative filtering and content-based recommendation with keyword extraction. *Applied and Computational Engineering*, 2023, vol. 2, no 1, p. 927-939.

[9] Karlsson. J, What it takes to build a real-time recommendation system, *tinybird.co,* [on line] available at: https://www.tinybird.co/blog-posts/real-time-recommendation-system. [Accessed on: 18/6/25]

[10] Hazga. S, بناء نظام توصيات باستخدام التعلم الآلي, Unit.AI, [online] available at: https://www.unite.ai/ar/building-recommendation-system-using-machine-learning/.[Accessed on: 13/6/25]

[11] FALK, Kim. *Practical recommender systems*. Simon and Schuster, 2019.

Reference

[12] KIM, Yang Sok, HWANGBO, Hyunwoo, LEE, Hee Jun, *et al.* Sequence aware recommenders for fashion E-commerce. *Electronic Commerce Research*, 2024, vol. 24, no 4, p. 2733-2753.

[13] ESMELI, Ramazan, CAN, Ali Selcuk, AWAD, Aya, *et al.* Understanding customer loyalty-aware recommender systems in E-commerce: an analytical perspective. *Electronic Commerce Research*, 2025, p. 1-27.

[14] Pop the Filter Bubble, AVID Open access, [online] available at: https://avidopenaccess.org/resource/pop-the-filter-bubble/. [Accessed on: 13/6/25]

[15] KIM, Yang Sok, HWANGBO, Hyunwoo, LEE, Hee Jun, *et al.* Sequence aware recommenders for fashion E-commerce. *Electronic Commerce Research*, 2024, vol. 24, no 4, p. 2733-2753.

[16] KANDULA, Ramya. Designing diverse features to reduce the filter bubble effect on social media. 2023.

[17] Epsilon-Greedy Algorithm in Reinforcement Learning, *geeksforgeeks.org.* [online] available at:https://www.geeksforgeeks.org/epsilon-greedy-algorithm-in-reinforcement-learning/.
[Accessed on: 11/6/25]

[18] BODDULURI, Kailash Chowdary, PALMA, Francis, KURTI, Arianit, *et al.* Exploring the landscape of hybrid recommendation systems in e-commerce: A systematic literature review. *IEEE Access*, 2024, vol. 12, p. 28273-28296.

[19] CANIARD, Étienne. Les recommandations de bonnes pratiques: un outil de dialogue, de responsabilité et de diffusion de l'innovation. *Saint-Denis: Haute Autorité de Santé*, 2002.

[20] HUSSIEN, Farah Tawfiq Abdul, RAHMA, Abdul Monem S., et WAHAB, Hala Bahjat Abdul. Recommendation systems for e-commerce systems an overview. In : *Journal of Physics: Conference Series*. IOP Publishing, 2021. p. 012024.

[21] NASIR, Mahreen et EZEIFE, Christie I. A survey and taxonomy of sequential recommender systems for e-commerce product recommendation. *SN Computer Science*, 2023, vol. 4, no 6, p. 708.

[22] NGUYEN, Tien T., HUI, Pik-Mai, HARPER, F. Maxwell, *et al.* Exploring the filter bubble: the effect of using recommender systems on content diversity. In : *Proceedings of the 23rd international conference on World wide web*. 2014. p. 677-686.

Reference

[23] Python books, python.org, [online] available at: https://wiki.python.org/moin/PythonBooks. [Accessed on: 20/4/25]

[24] Pyspark, *databricks.com,* [online] available at: https://www.databricks.com/fr/glossary/pyspark. [Accessed on: 22/4/25]

[25] Introduction-to-Jumpy-and-pandas, *code_academy*. [online] available at: https://www.codecademy.com/article/introduction-to-numpy-and-pandas. [Accessed on: 7/4/25]

[26] FILLO, T. C. et PACCA, C. A. Alternating Least Squares in Multidimensional Data Processing and its Applications. 2023.

[27] Cosine Similarity, MarketMuse.com, [online] available at: https://blog.marketmuse.com/glossary/cosine-similarity-definition/. [Accessed on: 5/3/25]

[28] Definition visual-studio, *Bility.fr*, [online] available at: https://bility.fr/definition-visual-studio-code/. [Accessed on: 5/4/25]

[29] Django Project, *Djangoproject.com,* [online] available at: https://www.djangoproject.com/. [Accessed on: 5/4/25]

[30] Python, [online] available at https://spark.apache.org/docs/latest/api/python/index.html. [Accessed on: 5/3/25]

[31] SQLite 3, S*ervbay.com*, [online] available at: [https://support.servbay.com/ar/database-management/getting-started/sqlite3-management-and-usage. [Accessed on: 5/4/25]

[32] Accuracy and precision, [online] available at: https://en.wikipedia.org/wiki/Accuracy_and_precision?utm_source. [Accessed on: 11/6/25]

[33] Definition amazon, [online] available at: https://www.techtarget.com/whatis/definition/Amazon. [Accessed on: 9/4/25]

[34] What Is Alibaba, [online] available at [https://www.speedcommerce.com/what-is/alibaba/ . [Accessed on: 1/4/25]

[35] Netflix, [online] available at: https://www.netflix.com/dz-en/title/81128579. [Accessed on: 8/6/25]

[36] Noon E-Commerce Company Economics Final Project, [online] available at: https://www.scribd.com/presentation/661946184/Noon-E-commerce-Company-Economics-Final-Poject. [Accessed on: 5/4/25]