INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA DE INTELIGÊNCIA ARTIFICIAL




An intelligent hybrid recommender system improved with Association Rules

JOÃO FILIPE COELHO MOREIRA
outubro de 2024

P.PORTO

An intelligent hybrid recommender system improved with Association Rules

João Filipe Coelho Moreira

Aluno nº: 1110297

Dissertação para obtenção do Grau de Mestre em Engenharia de Inteligência Artificial

Orientador: Joaquim Filipe Peixoto dos Santos, Professor Adjunto do Instituto Superior de Engenharia do Instituto Politécnico do Porto

Co-orientador: Doutor António Constantino Lopes Martins, Professor Adjunto do Instituto Superior de Engenharia do Instituto Politécnico do Porto

Júri:

Presidente: Ana Maria Neves Almeida Baptista Figueiredo, Professora Coordenadora do Instituto Superior de Engenharia do Instituto Politécnico do Porto

Arguente: Doutor Jorge Alexandre Novo Meira, Professor Adjunto Convidado do Instituto Superior de Engenharia do Instituto Politécnico do Porto

Orientador: Joaquim Filipe Peixoto dos Santos, Professor Adjunto do Instituto Superior de Engenharia do Porto do Instituto Politécnico do Porto

Porto, setembro 2024

Resumo

Com a popularização da Internet e o amadurecimento das tecnologias associadas, o ambiente digital tornou-se um mercado global que facilita a troca de bens e serviços, conhecido como e-commerce. Este mercado cresceu substancialmente com a expansão dos catálogos de produtos, assim como a necessidade de sistemas de recomendação eficazes de forma a melhorar a experiência do utilizador e aumentar a competitividade das empresas.

Esta dissertação investiga o estado atual dos sistemas de recomendação de comércio eletrónico, analisando as técnicas utilizadas atualmente, as suas limitações e métodos de avaliação. Propõe ainda uma abordagem híbrida que combina técnicas de recomendação com regras de associação derivadas de dados históricos de compras, com a atribuição de pesos para controlar a influência de cada técnica. O objetivo é oferecer aos utilizadores recomendações personalizadas e eficazes, tirando partido da combinação técnicas de recomendação estabelecidas com regras de associação, de forma a procurar mitigar limitações existentes.

A eficácia dos componentes desta abordagem híbrida é avaliada com métricas padrão e complementada pelo feedback de utilizadores de teste, contribuindo para o ajuste dos pesos e na análise da relevância das recomendações. Os resultados desta abordagem contribuem para o aumento da satisfação dos utilizadores de plataformas de comércio eletrónico, embora exista a necessidade de uma grande quantidade de dados para a criação de regras de associação.

Palavras-chave: Inteligência Artificial, E-commerce, sistemas de recomendação, regras de associação, filtragem colaborativa, filtragem baseada em conteúdo

Abstract

With the popularization of the Internet and the maturation of associated technologies, the digital environment has evolved into a global marketplace facilitating the exchange of goods and services, commonly referred to as e-commerce. This market has experienced substantial growth due to the expansion of product catalogues and the rising demand for effective recommender systems that enhance user experience and boost the competitiveness of companies.

This dissertation examines the current landscape of e-commerce recommender systems, analysing the techniques currently in use, their limitations, and evaluation methods. It also proposes a hybrid approach that integrates recommendation techniques with association rules derived from historical purchase data, assigning weights to balance the influence of each technique. The primary goal is to provide users with personalised and effective recommendations, leveraging the combination of established recommendation methods with association rules, to mitigate existing limitations.

The effectiveness of the components in this hybrid approach is evaluated using standard metrics, supplemented by feedback from test users, which aids in adjusting the weights and analysing the relevance of the recommendations. The findings of this approach contribute to increased user satisfaction on e-commerce platforms, although the creation of meaningful association rules requires substantial amounts of data.

Keywords: Artificial Intelligence, E-commerce, recommender systems, association rules, collaborative filtering, content-based filtering.

Acknowledgements

A big thanks to my family, girlfriend, friends, and my teammates from Basecone.

A special thanks to Professor Joaquim Filipe Peixoto dos Santos and Professor Constantino Martins for the guidance provided for this dissertation and valid inputs.

Index

Figures list

Tables list

Equations list

Code snippets

List of Acronyms

ARM	Association Rule Mining
AUC	Area Under Curve
BSON	Binary JavaScript Object Notation
CB	Content-based filtering
CF	Collaborative Filtering
CSV	Comma-separated values
DSR	Design Search Research
DSRM	Design Search Research Methodology
EDA	Exploratory Data Analysis
FP	Frequent Pattern
GB	Gigabyte
GDPR	General Data Protection Regulation
IDF	Inverse document frequency
JSON	JavaScript Object Notation
MAE	Mean Absolute Error
MB	Megabyte
MSE	Mean Squared Error
NMF	Non-negative matrix factorization
NLTK	Natural Language Toolkit
PRISMA	Preferred Reporting Items for Systematic Reviews and Meta-Analyses
RMSE	Root Mean Squared Error
ROC	Receiver operating characteristic
RS	Recommender system
SVD	Singular value decomposition
TF	Term frequency
US	United States

1 Introduction

This chapter introduces the dissertation, beginning with the context, problem definition, and primary objectives. It also outlines the key contributions of the work and presents the research methodology employed. Finally, the structure of the document is described.

1.1 Context

As the Internet became available to the average user and its associated technologies became sufficiently mature, this medium became a digital space where it is possible to carry out transactions of goods or services between various types of users including sellers, buyers, and others, and types of commerce such as Business-to-Business, Business-to-Consumer, and so on (Ferrera & Kessedjian, 2019). Several companies, such as Amazon and eBay, have been pioneers and contributed to scaling these transactions to a global level. A global market has thus emerged via the Internet, with various designations such as e-commerce or electronic commerce (Ferrera & Kessedjian, 2019).

To illustrate the evolutionary trajectory of e-commerce, Figure 1 depicts the notable milestones between 1994 and 2012.

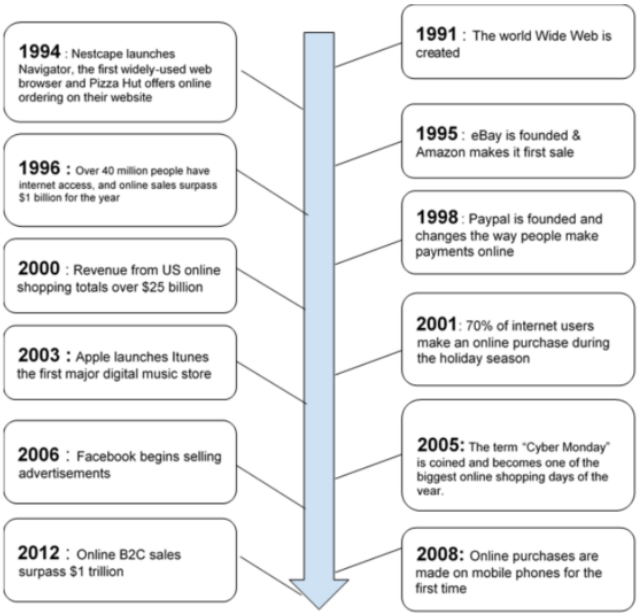

Figure 1 - E-commerce evolution (1994 – 2012) (Ferrera & Kessedjian, 2019)

As companies have embraced e-commerce, the catalogue of products available for sale has grown and recommender systems applied to e-commerce have become popular and equally important (Sharma & Singh, 2016).

In 1998, Amazon.com implemented a Recommender system which made recommendations on the website available, reaching millions of their users, using a catalogue featuring millions of items that vary from time to time. Since then, the original algorithm has been refined and improved and remains one of the most popular today, it was also used by YouTube for video recommendations (Smith & Linden, 2017).

As in physical spaces, such as electronic or retail stores, sales techniques such as cross-selling have been incorporated into online shops to make sales more profitable. Recommender systems are adaptable to user attributes and preferences, such as search history and visualizations, so that recommendations are accepted by the user (Ghoshal et al., 2021).

1.2 Problem definition

The requirements of the evolution of e-commerce have allowed recommender systems to be exposed to various scenarios and some weaknesses of the pioneer models have been identified. One identified scenario happens when a new item is added to the catalogue, it doesn't have any ratings and was never bought before, so it's hard for the recommender system to recommend it without any information. This also may affect a new user without purchase history or preferences, this limitation is called cold start (Bobadilla et al., 2013).

Hybrid recommender systems have been designed that combine models such as Content-Based Filtering or Collaborative Filtering, to take advantage of the advantages of some and cancel out the weaknesses of others (Aggarwal, 2016).

Association Rules, including popular algorithms such as Apriori, Eclat and FP-growth, make it possible to uncover patterns between items given their frequency in sets of items. If a pattern exhibits enough confidence, it could potentially serve as a basis for making recommendations. (Shao, 2022) Figure 2 provides an example of the multiple steps of the FP-growth algorithm.

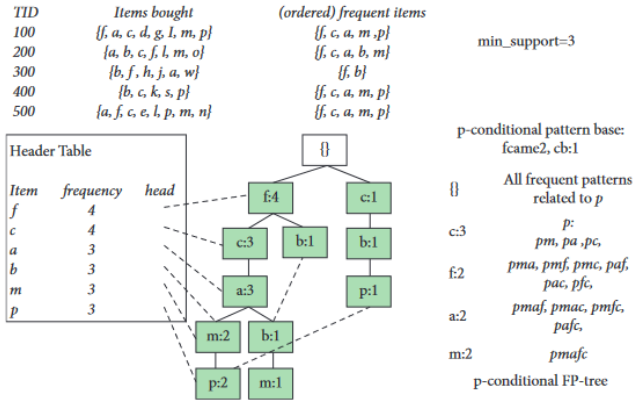

Figure 2 - Example flowchart of the FP-growth algorithm (Shao, 2022)

This dissertation aims to explore the current state of hybrid recommender systems, their application to e-commerce and the incorporation of Association Rules into a system of this type, to adjust or refine suggestions according to the user profile and item characteristics.

1.3 Objectives

Presents the dissertation's main research question and objectives:

This dissertation's main research question explores the intersection of e-commerce and recommender systems: **How can association rules be effectively integrated into an e-commerce hybrid recommender system to improve known limitations?** This question leads to the main tasks definition to accomplish this main research question:

Analyse the current landscape of e-commerce hybrid recommender systems: Conduct a systematic review of existing literature to gain insights into the challenges faced by e-commerce hybrid recommender systems and identify key trends in their application.

Examine evaluation strategies for recommender systems: Investigate the current landscape of evaluation metrics used for recommender systems through an extensive literature review, focusing on understanding the criteria commonly employed in assessing the effectiveness of hybrid recommender systems.

Study the impact of integrating Association Rules in e-commerce hybrid recommender systems: Evaluate and compare the effectiveness of integrating association rules with other techniques within the context of hybrid recommender systems in e-commerce. Analyse the outcomes to understand the potential enhancements in recommender accuracy and user satisfaction.

The primary goal of this dissertation is to explore the use of association rules within a hybrid recommender system for e-commerce. To achieve this, a prototype recommender system will be developed, which aims to provide product recommendations and address the user cold-start problem.

1.4 Contributions

This dissertation makes significant contributions to the field of recommender systems by addressing both theoretical gaps and practical challenges, particularly in the e-commerce context.

A key contribution is the comprehensive survey of the current state of recommender systems, conducted through a systematic review using the PRISMA methodology. This review critically examines the limitations of classical techniques, explores various evaluation methods, and explores the role of association rules in enhancing recommendation systems, with a particular focus on e-commerce.

In addition to the systematic review, this work presents the design and development of a hybrid recommender system prototype. The system integrates item-based collaborative filtering, content-based filtering, and association rule mining, creating a cooperative framework that leverages the strengths of each approach. The development of this prototype involved building each component from scratch, followed by their integration into a unified system.

A major practical contribution is the evaluation of this hybrid system's ability to mitigate the new user cold start problem. By applying a weighted combination of collaborative filtering, content-based methods, and association rules, the system improves the quality of recommendations for new users with no interaction history. The system's weights were adjusted to the feedback provided by test users.

1.5 Research Methodology

One possible research methodology for this dissertation would be one adaptation of **Design Search Research** (DSR). DSR aims to improve human knowledge with the creation of innovative artefacts and the generation of design knowledge via innovative solutions to real-world problems. (vom Brocke et al., 2020)

Hevner et al., (2004) proposed seven guidelines: **1) Design as an artefact; 2) Problem relevance; 3) Design evaluation; 4) Research contributions; 5) Research rigour; 6) Design as a search process and 7) Communication of research.**

Building upon the foundation laid by Hevner et al. (2004), (Peffers et al., 2007) updated the DSR framework by incorporating a structured six-step process and four possible entry points: (vom Brocke et al., 2020)

1. **Problem Identification and Motivation:** The initial phase involves identifying and articulating the problem, emphasizing its significance and motivation for investigation.
2. **Definition of Objectives for a Solution:** Clearly define the objectives that the designed solution aims to achieve, setting a focused direction for the research.
3. **Design and Development:** Executing the design and development of the solution, leveraging innovative and effective methods to address the identified problem.
4. **Demonstration:** Provide a tangible demonstration of the developed solution to showcase its functionality and practicality.
5. **Evaluation:** Rigorous evaluation of the designed artefact to assess its performance, usability, and overall effectiveness in solving the identified problem.
6. **Communication:** Effectively communicating the research outcomes, ensuring that the insights gained are disseminated to relevant audiences, contributing to the broader academic discourse.

And four possible entry points: (vom Brocke et al., 2020)

1. **Problem-Centered Initiation:** Beginning the research process by identifying and exploring a specific problem that warrants investigation.
2. **Objective-Centered Solution:** Initiating the research based on clearly defined objectives and crafting a solution to address the identified problem.
3. **Design and Development-Centered Initiation:** Starting the research with a focus on the design and development aspects, aiming to create a tangible solution.
4. **Client/Context Initiation:** Commencing the research by considering the client or contextual requirements, ensuring the designed solution aligns with the practical needs of stakeholders.

Figure 3 represents the DSRM process proposed by (Peffers et al., 2007).

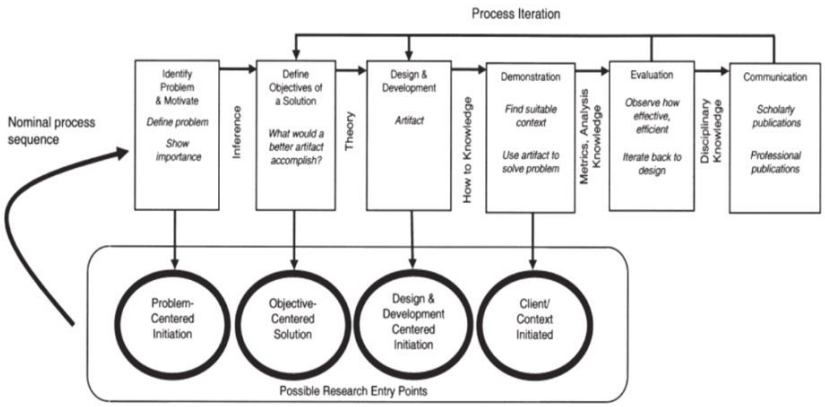

Figure 3 - DSR Methodology Process Model (vom Brocke et al., 2020)

This dissertation is designed to follow the DSRM process through the following steps:

1. **Problem Identification and Motivation:** Investigate the possibility of using Association Rules to improve e-commerce recommender systems.
2. **Definition of Objectives for a Solution:** Definition of clear goals for improving the recommender system, providing a focused direction for the research.
3. **Design and Development:** Design and development of an hybrid recommender system improved with association rules.
4. **Demonstration:** Demonstration of the developed prototype to the dissertation supervisors, showcasing its functionalities.
5. **Evaluation:** Using standard recommender systems evaluation methods to evaluate the prototype components.
6. **Communication:** Concluding the research process with a dissertation presentation to the academic audience (Juris).

1.6 Document structure

This subsection outlines the structure of the dissertation.

The first chapter, Introduction, not only includes this subsection but also provides essential elements for understanding the dissertation. It includes the context, problem definition, objectives, and expected contributions to the fields of Recommender Systems and Artificial Intelligence and details the research methodology employed in creating this document.

The second chapter is the State of the Art of Recommender Systems. It begins with an introduction to Recommender Systems concepts, covering definitions, techniques, known limitations, and evaluation methods. Following this, the chapter includes a comprehensive description of the search methodology, detailing the data extraction process, the results and their contribution for practical approach.

The third chapter, Experimentation, includes a detailed description of the technologies employed to implement the prototype of a hybrid recommender system. It follows with an overview of the Amazon Review Dataset, including its description, evolution over time, and the steps taken for exploratory data analysis and pre-processing. Additionally, the chapter presents the evaluation methods for the approach and concludes with a discussion on data protection, security analysis, and the ethical considerations involved.

Chapter four, Implementation, describes a proposed implementation for a weighted hybrid recommender system prototype, using content-based filtering and item-based collaborative filtering that considers association rules when recommending items. This implementation was presented to test users to study the relevance of the recommendations and perform some weight adjustments.

Finally, chapter five, Conclusions, reviews the objectives achieved, discusses the limitations of the research as well as potential directions for future work and concludes with final notes.

2 State of the Art

This chapter is divided into four main sections. It begins by introducing key concepts related to recommender systems, including their definition, various techniques, common limitations, and methods of evaluation. The second section provides a brief overview of Association Rules and the Apriori algorithm. Next, a PRISMA systematic review is presented to synthesize the existing knowledge and advancements in the field of recommender systems.

2.1 Recommender systems

Recommender systems allow users to navigate through the immensity of information available on the most diverse platforms. Their primary objective is to provide personalised recommendations that are useful to the user.

These systems employ sophisticated algorithms, mainly categorized between content-based recommendation, knowledge-based, and collaborative filtering-based recommendation, but they can also be used together, creating a hybrid recommender system.

2.1.1 Recommender system techniques

This section explores the recommender system techniques: content-based filtering (CBF), collaborative filtering (CF), knowledge-based recommenders (KBF), and hybrid filtering (HF), each contributing to a comprehensive understanding of how recommendation systems work.

2.1.1.1 Content-based filtering (CBF)

The Content-based filtering approach was designed to offer users personalized suggestions by examining intrinsic attributes considered similar to those that the user had interacted with.

CBF primarily involves performing a comprehensive analysis of the content related to each item. This content includes keywords, metadata, item genre and description. In e-commerce, examples of attributes may include the item category, price, size, and weight. (Ko et al., 2022)

In this process, a similarity function is employed to define the likeness between items. This function considers the previously mentioned characteristics and the user's known preferences. (Murty et al., 2022) It's important to note that, while relying on item feature similarity is essential, it may lead to novelty problems. This arises because the system might not explore new items that could interest the user but lack similar features. (Salter & Antonopoulos, 2006)

Figure 4 provides a visual summary of the content-based filtering flow, where the users are recommended items similar to the user preferences.

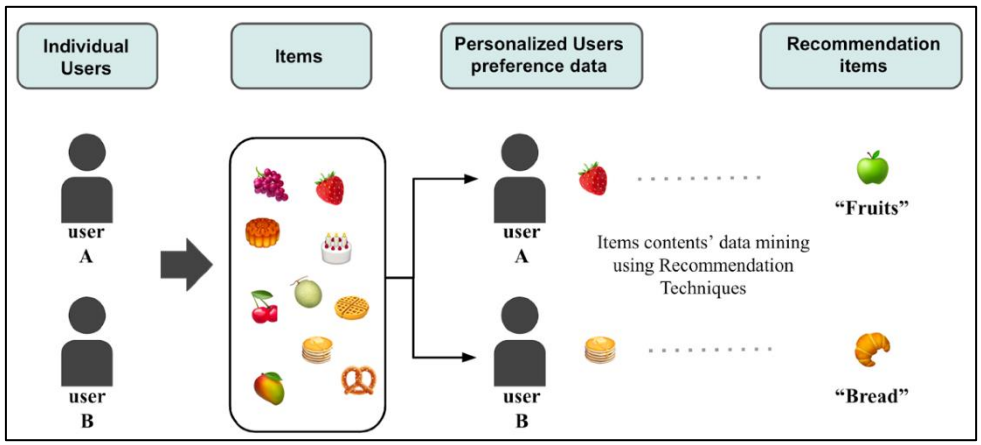

Figure 4 - Content-based filtering flow (Ko et al., 2022)

Nearest Neighbour classification

The nearest neighbour classifier is one of the most intuitive and straightforward methods used in content-based recommender systems. This method relies on the principle of similarity, where the class label of a new instance is determined based on the labels of the closest instances in the training data. The effectiveness of this approach relies on the choice of a similarity or distance function, as it determines how "near" data points are to each other (Aggarwal, 2016).

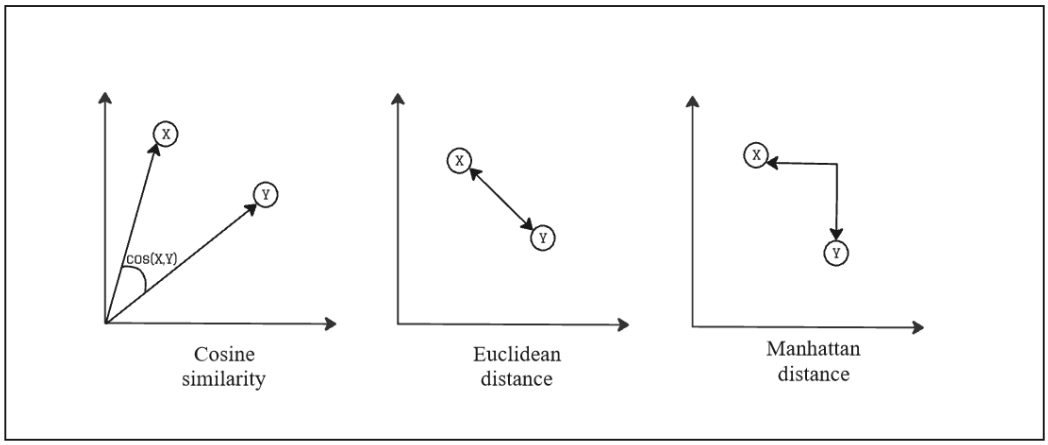

Figure 5 – 2-dimension distance measures representation adapted from (Maarten Grootendorst, 2021)

In the text domain, cosine similarity (1) is frequently employed due to its ability to account for varying document lengths. This makes it particularly well-suited for comparing text-based data, where the magnitude of the vectors can differ significantly. For other types of structured and multidimensional data, alternative similarity or distance measures, such as Euclidean distance (2) or Manhattan distance (3), are often used to capture the most relevant aspects of similarity between instances (Aggarwal, 2016).

$$Cosine(X, Y) = \frac{X \cdot Y}{|X||Y|} \quad (1)$$

Where:

- X and Y are vectors.

$$d_{\text{Euclidean}}(X, Y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2} \quad (2)$$

$$d_{\text{Manhattan}}(X, Y) = \sum_{i=1}^{d} |x_i - y_i| \quad (3)$$

Where for both equations (2) and (3):

- X and Y are vectors
- d is the number of dimensions in the data.
- x_i is the i-th coordinate of vector X.
- y_i is the i-th coordinate of vector Y.

2.1.1.2 Collaborative filtering (CF)

The Collaborative Filtering recommender system technique stands out as a widely adopted and popular approach. This technique relies on the opinions and preferences of other users to provide recommendations to the current user, irrespective of the specific attributes of the items in question. Its flexibility lies in the ability to focus either on users like the current user or on products akin to those the current user has expressed interest in. The determination of similarity is accomplished by a function tailored for this purpose (Ni et al., 2021;Alam et al., 2021).

When the system prioritizes users with similar preferences, it falls under the subtype known as User-based Collaborative Filtering. In this case, the similarity function can consider various user attributes such as age, gender, profession, hobbies, and more. This approach aims to identify users with comparable profiles and preferences, enhancing the relevance of recommendations (Bobadilla et al., 2012).

Alternatively, if the system directs its focus toward products, it aligns with the subtype called Item-based Collaborative Filtering. Here, the similarity function considers products that are

likely to be deemed similar if other users who have common purchases have bought them together. This strategy relies on the idea that items frequently bought together are likely to share similarities (Pirasteh et al., 2014).

The graphical representation in Figure 6 serves as a visual guide to understanding the different types of Collaborative Filtering. It illustrates how user-based and item-based collaborative filtering operates, showcasing the interconnected relationships between users and items within the recommender system.

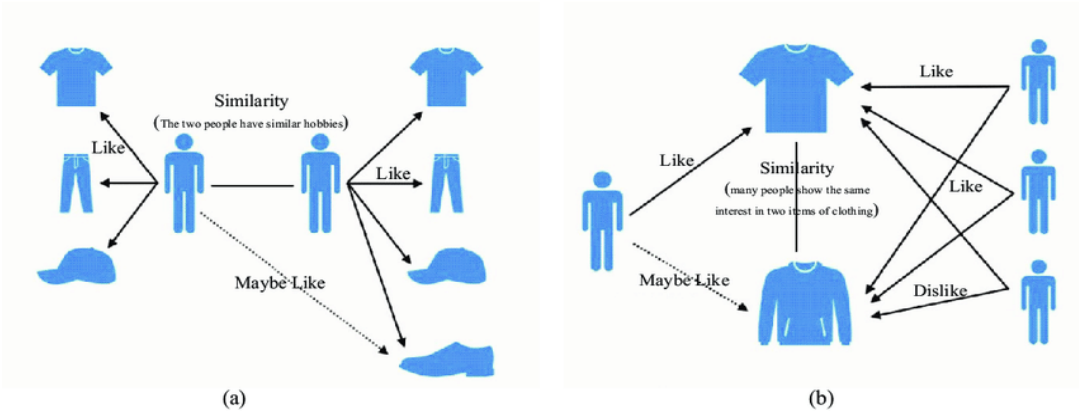

Figure 6 - User-based (a) and Item-based (b) collaborative filtering (Ni et al., 2021)

Both subtypes of Collaborative Filtering require the existence of a historical record, both for the user and the items. This historical data includes the user's interests in products, such as previous purchases, as well as evaluations and reviews provided for items. Therefore, whether it's the user's history of product interests or the items' history of purchases and reviews, both are critical for the effective functioning of Collaborative Filtering.

It's worth noting that both users and items can face challenges related to the cold-start problem when they are newly introduced to the system (Kalidindi et al., 2019). If the users have a history but don't review the items, they may also run into Sparsity problems (Alhijawi & Kilani, 2020).

Matrix dimensionality reduction methods

Dimensionality reduction is a technique used to create more manageable and interpretable data representations, especially when dealing with incomplete or high-dimensional datasets. Methods such as Singular Value Decomposition and Non-Negative Matrix Factorization, are advanced methods that not only reduce dimensionality but also directly estimate the underlying data matrix (Aggarwal, 2016).

Singular value decomposition (SVD) is a form of matrix factorization in which the columns of U and V are constrained to be mutually orthogonal. The SVD theorem (Golub & Kahan, 1965), states that any given matrix *M* can be decomposed into a product of three matrices (4) as represented in Figure 7 (Jannach et al., 2010).

$$\mathrm{M} = \mathrm{U}\,\Sigma\,V^{\mathrm{T}} \quad (4)$$

Where:

- U represents the left singular vectors of M.
- Σ represents the singular values of M.
- V^{T}represents the right singular vectors of M.

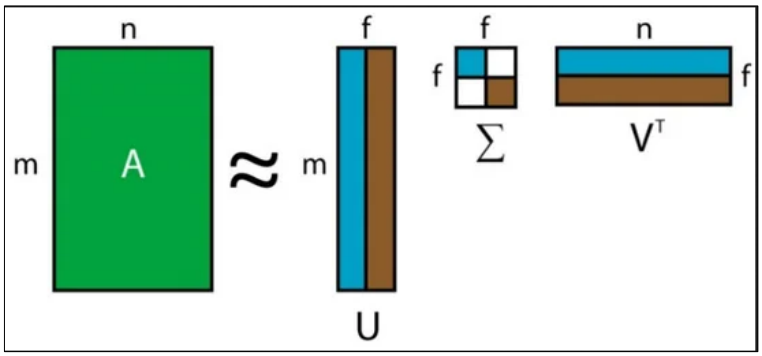

Figure 7 - Matrix decomposition using SVD (Widiyaningtyas et al., 2022)

Non-Negative Matrix Factorization (NMF) is another dimensionality reduction technique, but it imposes the constraint that all elements in the factorized matrices must be non-negative (Jannach et al., 2010).

2.1.1.3 Knowledge-based Recommenders (KBF)

Knowledge-based recommenders recommend items by leveraging domain-specific knowledge, aligning certain item features with user needs and preferences, and gauging the overall usefulness of items (Katarya & Verma, 2017).

According to Ricci et al., (2011), two notable types of knowledge-based systems are identified:

Case-based recommender systems: these systems employ a similarity function to assess the match between user needs (problem descriptions) and recommended solutions. The resulting similarity score directly signifies the utility of the recommendation for the user.

Constraint-based recommenders: primarily relying on predefined knowledge bases, these systems incorporate explicit rules on how to correlate customer requirements with item features.

Ricci also suggests that while knowledge-based systems tend to exhibit strong performance initially, particularly in the early stages of deployment, their effectiveness may diminish without the integration of learning components.

2.1.1.4 Hybrid filtering (HF)

Hybrid filtering represents a combination of diverse features from different recommender systems. By combining various recommender systems, the goal is to cancel out the weaknesses of one system while capitalizing on the strengths of another, ultimately creating a more resilient and versatile recommender system.

According to Burke (2002), hybrid recommender models can be categorized into seven distinct types, each employing a unique method for combining filtering techniques:

Cascade Hybridization: In this model, the output of one recommender is refined in the next one, creating a layered or hierarchical recommendation process.

Feature augmentation: This involves enhancing the feature set of one recommender system with information from another. This approach aims to supplement and enrich the existing feature space to improve recommendation accuracy.

Feature combination: Feature combination focuses on merging the features or characteristics extracted from various recommender systems aiming to create a unified feature set that captures the strengths of each system.

Meta-level: The full output model from one recommender is fed to the next one.

Mixed Hybridization: In the mixed hybridization model, recommendations from different systems are combined without assigning explicit weights.

Switching Hybridization: This approach involves selecting one recommender system over another based on specific conditions or contexts. The system adapts between different recommendation methods depending on the current scenario.

Weighted Hybridization: In this model, different recommendation systems contribute to the final suggestion with varying weights.

Figure 8 represents a Hybrid recommender model, merging the recommendations from content-based filtering and collaborative filtering algorithms. The figure illustrates the integration of these approaches to provide a more comprehensive and personalized recommender system.

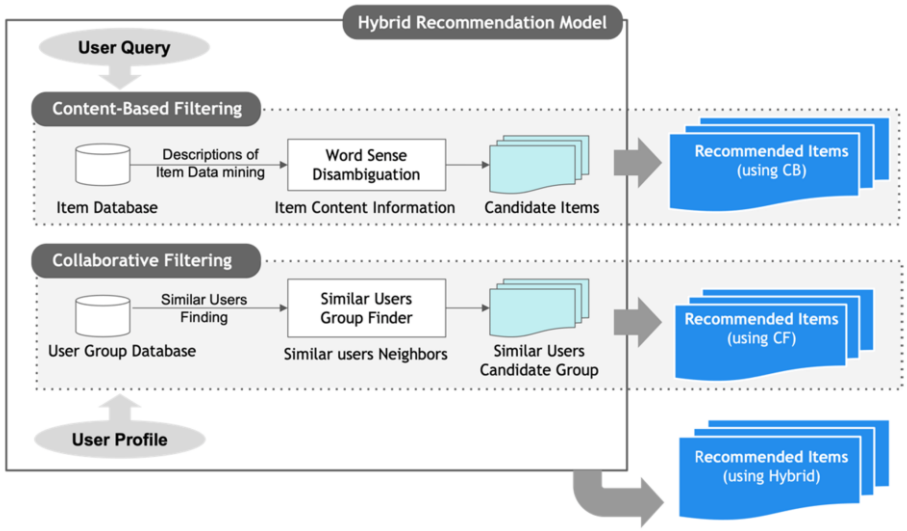

Figure 8 - Hybrid recommendation model (Ko et al., 2022)

2.1.2 Known Recommender systems challenges

Recommender systems face several key challenges that impact their accuracy and performance. Among the most significant are the grey-sheep problem, the cold start issue, and data sparsity, all of which limit the effectiveness of recommendations. Below is a concise overview of these challenges.

Grey-sheep

The grey sheep problem represents a unique challenge within collaborative filtering methods. It revolves around users whose opinions exhibit a certain level of neutrality or inconsistency, making it difficult to categorize them into a specific group with shared preferences. Unlike users who consistently agree or disagree with a particular group, these “grey sheep” users fall into a middle ground where their preferences do not align strongly with any established pattern (Lucas et al., 2013).

Cold start

In the domain of recommender systems, cold-start problems pose a significant challenge, affecting the generation of reliable recommendations due to a lack of initial ratings. According to Bobadilla (Bobadilla et al., 2012) three distinct types of cold-start problems emerge: new community, new item, and new user.

The new community problem refers to the challenge of acquiring enough data (ratings) to make reliable recommendations when starting a recommender system. Insufficient users and ratings make it difficult to retain new users who encounter a system with content but no precise recommendations.

The new item problem arises because newly added items in the recommender system often lack initial ratings, making them unlikely to be recommended. This sets off a potentially vicious cycle where unnoticed items receive fewer ratings and recommendations (Taneja & Arora, 2018).

The new user stands out as a major challenge in operational recommender systems. When users join a platform, they lack information usually as they haven't cast any votes, preventing personalized Collaborative Filtering recommendations. As users enter their first ratings, the number may not be sufficient for reliable recommendations, potentially causing users to feel dissatisfied with the system (Taneja & Arora, 2018).

Sparsity

This challenge is primarily experienced due to sparse information, where the available ratings set by users are limited, impacting the system's ability to provide accurate recommendations (Koohi & Kiani, 2021).

This issue arises from the non-availability of a comprehensive set of ratings, leading to a sparse rating matrix with many missing values. Specifically, this challenge is observed when a user has provided ratings for only a few items (Rahim et al., 2022).

Because of sparsity, the task of defining similarity between users becomes highly challenging. This difficulty, impacts the identification of suitable neighbours, leading to inaccurate recommendations (Shambour et al., 2021).

2.1.3 Recommender systems results evaluation

Evaluating recommender systems is used to ensure their effectiveness in delivering accurate and relevant suggestions to users. Aggarwal (2016), in *Recommender systems: The textbook*, and Henriques & Pinto (2023) outline two primary evaluation methods: online and offline, defining three main types of evaluation: user studies, online evaluations and offline evaluations with historical data sets. The first two methods use user feedback.

User studies involve recruiting participants to interact with a recommender system and perform specific tasks. Feedback is gathered before and after the interaction, while the system collects data on user behaviour. This data is then used to evaluate the system's effectiveness. A key advantage of user studies is the ability to test different scenarios, such as changing algorithms or interfaces. However, these studies are costly, often involve biased user samples, and the participants' awareness of being tested can skew their responses. As a result, the findings may not always be fully reliable or representative of the general population (Aggarwal, 2016).

Online evaluations, involve real users in a live system, reducing recruitment bias. These evaluations test algorithm performance in a more natural setting, often through randomized sampling of users (Aggarwal, 2016).

Offline evaluation, uses historical data, is the most common used approach since it doesn't require real-time user feedback, and some frameworks and evaluation measures have been developed for this purpose, already exploring the domain of Machine Learning.

Within the offline method, accuracy metrics can be considered important, but some secondary objectives such as Novelty, Trust, Coverage and Serendipity also play significant roles in enhancing the user experience (Aggarwal, 2016; Jannach & Jugovac, 2019).

Accuracy and accuracy metrics for offline evaluations

Accuracy is a critical metric for evaluating how well a recommender system predicts user preferences. It is assessed in two ways: prediction accuracy (estimating ratings) and classification accuracy (ranking recommendations).

Rating prediction typically employs regression-based metrics such as Mean Squared Error (MSE) (5), Root Mean Squared Error (RMSE) (6) and Mean Absolute Error (MAE) (7).

MAE, for example, measures the average difference between predicted and actual values. Lower MAE values indicate better accuracy, making it a preferred metric due to its simplicity (Aggarwal, 2016).

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2 \qquad (5)$$

$$\mathrm{RMSE} = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2} \qquad (6)$$

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|x_i - y_i| \qquad (7)$$

Where for all equations:

- n is the number of data points.
- y_i is the predicted value for each sample.
- x_i is the observed value for each sample.

For top-k ranking accuracy, Precision and Recall are key metrics (Schröder et al., 2011; Gupta et al., 2018). Precision (8) measures the accuracy of positive classifications within the recommended items, while Recall (9) assesses how many relevant items the system identified from all possible relevant items.

$$\text{Precision} = \frac{\text{Number of True Positives}}{\text{Number of True Positives} + \text{Number of False Positives}} \quad (8)$$

$$\text{Recall} = \frac{\text{Number of True Positives}}{\text{Number of True Positives} + \text{Number of False Negatives}} \quad (9)$$

Where for both equations:

- True Positives are the count of items that are both relevant and retrieved.
- False Negatives are the count of items that are relevant but not retrieved.

Novelty measures the system's ability to suggest new, less familiar items to users. In e-commerce, novelty might involve recommending innovative products like virtual reality gadgets to users typically purchasing smartphones or headphones, enhancing the user experience by offering less popular options (Aggarwal, 2016) (Karthik & Ganapathy, 2021).

Coverage refers to the system's ability to provide recommendations to a wide range of users and items, despite limitations like sparse data. A higher coverage indicates the system can recommend items across a broader user base. (Aggarwal, 2016)

Trust evaluates user confidence in the recommendations, with explainability being a key factor. Systems that explain their suggestions effectively can increase user trust (Jha et al., 2023), Aggarwal (2016).

Serendipity measures the element of surprise in recommendations, offering unexpected but relevant items. For instance, suggesting a portable solar-powered charger to a customer who frequently buys camping gear introduces a new product that aligns with their interests, enhancing their experience (Aggarwal, 2016), (Wang et al., 2019).

2.2 Association Rules

Association rules or Association rule mining (ARM) is considered a common technique used to identify patterns in large-scale sales transactions, such as pairs or groups of products often purchased together. This knowledge can be used for promotional and cross-selling purposes or for designing store layouts (Jannach et al., 2010).

Association rule mining involves discovering relationships of the type X implies Y, where X is the antecedent and Y is the consequent. It is a powerful method for market basket analysis, helping companies increase sales by identifying buying patterns and recommending complementary products (Shao, 2022).

Agrawal's (Agrawal & Srikant, 1994) the example illustrates that, for instance, 98% of customers buying tyres and auto accessories also opt for automotive services.

The Apriori algorithm, represented in Figure 9, a commonly used rule-mining algorithm, focuses on discovering frequent item sets, regardless of the purchase order, by iteratively scanning the database. This algorithm is employed, with three key aspects, Confidence (10), Support (11) and Lift (12) (Lourenco & Varde, 2020).

Confidence: the probability that Y occurs given X occurs.

$$\text{Confidence}(\{X\} \rightarrow \{Y\}) = \frac{\text{T}ransactions\ containing\ both\ X\ and\ Y}{Transactions\ containing\ X} \quad (10)$$

Support: the probability of both X and Y occurring together.

$$\text{Support}(\{X\} \rightarrow \{Y\}) = \frac{\text{T}ransactions\ containing\ both\ X\ and\ Y}{Total\ number\ of\ transactions} \quad (11)$$

Lift the probability of X and Y occurring together divided by the multiplication of their probabilities.

$$\text{Lift}(\{X\} \rightarrow \{Y\}) = \frac{(Transactions\ of\ both\ X\ and\ Y) \Big/ (Transactions\ containing\ X)}{Transactions\ containing\ Y} \quad (12)$$

Where for all equations (10), (11) and (12):

- X is the antecedent item.
- Y is the consequent item.

However, Apriori algorithm efficiency is challenged with massive data due to the need for multiple scans, resulting in low performance (Shao, 2022).

1) $L_1 = \{\text{large 1-itemsets}\}$;
2) **for** ($k = 2$; $L_{k-1} \neq \emptyset$; k++) **do begin**
3) $C_k = \text{apriori-gen}(L_{k-1})$; // New candidates
4) **forall** transactions $t \in \mathcal{D}$ **do begin**
5) $C_t = \text{subset}(C_k, t)$; // Candidates contained in t
6) **forall** candidates $c \in C_t$ **do**
7) c.count++;
8) **end**
9) $L_k = \{c \in C_k \mid c.\text{count} \geq \text{minsup}\}$
10) **end**
11) Answer = $\bigcup_k L_k$;

Figure 9 - Apriori algorithm (Agrawal & Srikant, 1994)

Another commonly used algorithm is the FP-Growth (Frequent Pattern Growth) algorithm, which is an efficient alternative to the Apriori algorithm for mining association rules. Unlike Apriori, FP-Growth reduces the number of database scans to just two, improving execution efficiency. The algorithm operates by first scanning the transaction database to identify frequent items and constructing a header table. In the second scan, it builds an FP tree by organizing transactions based on these frequent items. Association rules are then mined by recursively extracting conditional pattern bases from the FP tree. The key advantage of FP-Growth is that it avoids generating candidate sets, instead using a divide-and-conquer strategy, which significantly enhances performance compared to Apriori (Shao, 2022).

2.3 Search methodology

PRISMA stands for “Preferred Reporting Items for Systematic Reviews and Meta-Analyses” which was initially published in 2009 (Page et al., 2021), it was developed to assist systematic reviewers in transparently documenting the rationale behind their reviews, the methodologies employed, and their findings.

As systematic review methodology and terminology have evolved in the past decade, an update to the guidelines became necessary, the PRISMA 2020 replaced the original version (Page et al., 2021).

This search follows an adapted version of the PRISMA guidelines, required by time constraints. The approach involves considering the top results provided by the data sources’ search engines.

All the data was extracted from the data sources that will be presented next in this section and the search process was started in November 2023.

2.3.1 Data sources

In this subsection, it's presented the data sources explored to construct a comprehensive foundation for this search, outlined in Table 1. The selection comprises four distinct and reputable data sources: B-ON (Biblioteca do Conhecimento Online, 2024), IEEE Xplore (Institute of Electrical and Electronics Engineers, 2024), ACM Digital Library (Association for Computing Machinery, 2024) and Google Scholar(Google, 2024).

These data source platforms enable the user to perform advanced Boolean searches and filter which document metadata is relevant for the search.

Table 1 - Data sources used in the search.

Identification	Description
DS1	B-ON
DS2	IEEE Xplore
DS3	ACM Digital Library
DS4	Google Scholar

2.3.2 Search questions

The investigation presented in this dissertation is driven by three critical search questions, outlined in Table 2, each aimed at finding distinct aspects of e-commerce recommender systems. These questions lay the base for a comprehensive exploration of the challenges, evaluation metrics, and association rules integration strategies pertinent to the current landscape of intelligent recommender systems in the e-commerce domain.

Table 2 - Search questions

Identification	Description
SQ1	What are the key challenges and limitations of existing e-commerce recommender systems?
SQ2	Which evaluation metrics are suitable for assessing the performance of an e-commerce hybrid recommender system?
SQ3	How can association rules be effectively integrated into an e-commerce hybrid recommender system?

The initial search question, SQ1, aims to identify and understand the primary challenges and limitations associated with current e-commerce recommender systems. The focus of the second search question, SQ2, is on the identification of appropriate evaluation metrics to

measure the effectiveness and performance of intelligent e-commerce hybrid recommender systems. The third and final search question, SQ3, addresses the integration of association rules into e-commerce hybrid recommender systems exploring strategies and approaches for effectively incorporating association rules to enhance the recommendation capabilities of the system.

2.3.3 Search terms

Search terms play a crucial role in the systematic identification of relevant studies. This subsection delineates the specific terminology employed and the respective domain to navigate the data sources systematically, ensuring that the search is comprehensive and reflective of the study's objectives. Five domains were identified, and the research terms and their associated domains are detailed in Table 3.

Table 3 - Search terms and domains

Domain	**Term**
Recommender systems	"hybrid recommendation systems" OR "recommendation system" OR "recommender system"
General Challenges or Limitations	"limitations" OR "problems" OR "challenges"
E-commerce	"e-commerce" OR "electronic commerce" OR "online shopping"
Evaluation Metrics or Performance Metrics	"evaluation" OR "performance" OR "metrics"
Data Mining	"Association rules mining" OR "association rules"

2.3.4 Inclusion and exclusion criteria

The inclusion criteria define the characteristics that studies must possess to be considered for analysis and are outlined in Table 4.

Table 4 - Inclusion criteria

Inclusion criteria	**Description**
IC1	Publications in English language.
IC2	The data source provides access to the integral publication.
IC3	Published since 2017.
IC4	Focus on e-commerce recommender systems
IC5	Journal articles or conference papers are peer-reviewed.

The initial search was confined to scholarly publications written in English over the last seven years, with a specific focus on e-commerce recommender systems. Only peer-reviewed journal articles or conference papers were considered, and preference was given to data sources providing access to the integral publication.

The exclusion criteria outline the parameters that render studies ineligible, represented in Table 5. The exclusion criteria were applied during the initial screening process to refine the search. Duplicated publications found in multiple databases were excluded to avoid redundancy. Publications categorized as surveys or systematic reviews were omitted to prioritize original research contributions and innovative approaches. Additionally, publications not introducing or applying an approach for recommender systems were excluded, ensuring relevance to the search focus.

Table 5 - Exclusion criteria

Exclusion criteria	**Description**
EC1	Duplicated publications
EC2	Not relevant to the search question and outcomes
EC3	Published before 2017

2.3.5 Data extraction

For the defined 3 search questions, 3 initial potential query strings were created and for readability are shown in Table 6. However, the final query string represents the combination of all 3 using adequate operators represented in Table 7.

Table 6 - Initial Query strings

Identification	**Query string**
QS1	(hybrid recommendation systems OR recommendation system OR recommender system) AND (e-commerce or electronic commerce or online shopping) AND (limitations OR problems OR challenges)
QS2	(hybrid recommendation systems OR recommendation system OR recommender system) AND (e-commerce OR electronic commerce OR online shopping) AND (evaluation OR performance OR metrics)
QS3	(hybrid recommendation systems OR recommendation system OR recommender system) AND (e-commerce OR electronic commerce OR online shopping) AND (association rules mining OR association rules)

Table 7 - Final query string

	Query string
Final Query String	(hybrid recommendation systems OR recommendation system OR recommender system) AND (e-commerce OR electronic commerce OR online shopping) AND ((limitations OR problems OR challenges) OR (evaluation OR performance OR metrics) OR (association rules mining OR association rules))

The first step of this process was executing the final query string in all specified data sources. The B-ON library search yielded 108 publications, IEEE Xplore returned 57 publications, the ACM Digital Library provided 98 results, and Google Scholar retrieved 1210 entries, resulting in a total of 1473 publications.

Given the large number of total results from all the sources, the review focused primarily on the B-ON search results. From the initial pool of 1473 publications, 7 duplicates were removed, leaving 101 records from the B-ON dataset for title and abstract screening. During this phase, 4 entries were excluded, 2 for being published before 2017 and 2 due to unavailability of the full documents.

The remaining 97 documents underwent a thorough eligibility review, resulting in the removal of 25 off-topic articles. This careful selection process yielded a final set of 72 relevant documents. These publications formed the foundation of the State-of-the-Art subsection, contributing to answering the search questions and the explanation of fundamental recommender system concepts discussed earlier in this chapter.

In the following subsections, a comprehensive analysis of the findings for each search question will be conducted, examining their impact on the dissertation's objectives. Figure 11 details this process and Figure 10 presents the distribution of the results per category.

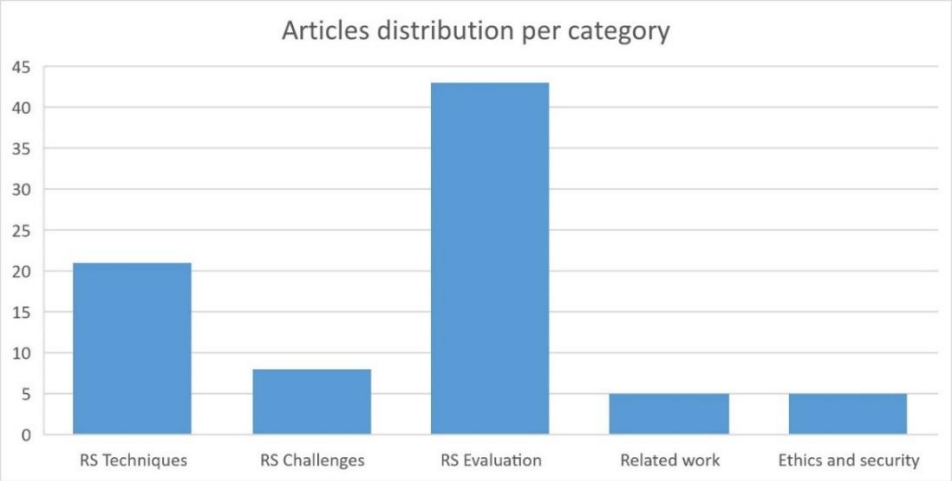

Figure 10 - Articles distribution per category

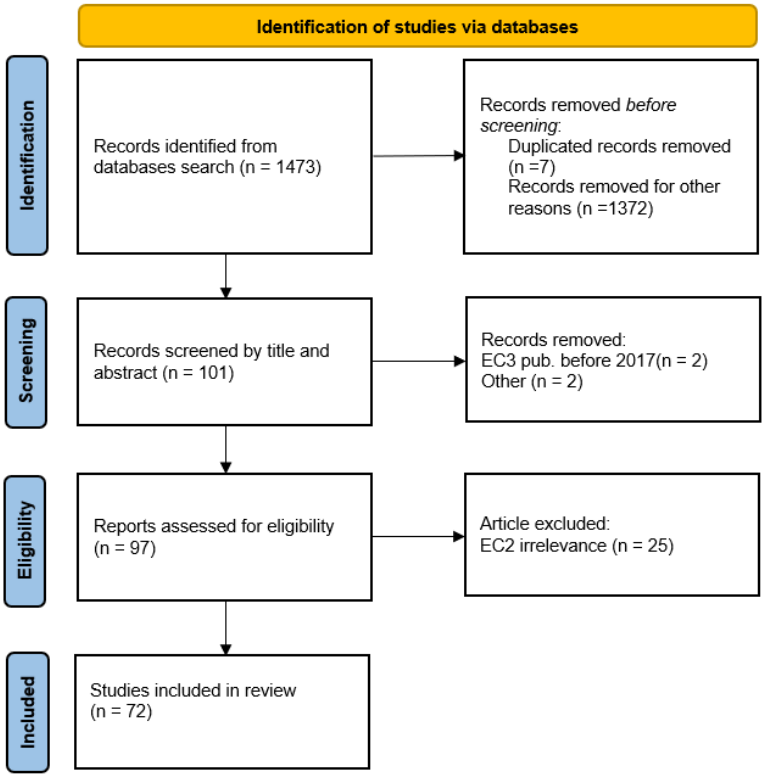

Figure 11 - PRISMA flow diagram

2.3.6 Search questions results

The findings are presented for each search question defined previously followed by an overall summary.

SQ1 - What are the key challenges and limitations of existing e-commerce recommender systems?

The results for the search question highlights key challenges in e-commerce recommender systems, specifically the cold-start problem, long-tail item recommendation, and data sparsity, as well as the proposals to solve this kind of problems. These issues stem from the difficulty of recommending new or niche items, particularly in systems that rely on collaborative filtering.

Several studies returned in the query results, propose enhancements to CF-based approaches. For instance, the use of side-information such as user attributes, social data, or item content has been suggested to address cold-start issues, as described in “On Both Cold-Start and Long-Tail Recommendation with Social Data” (Li et al., 2021). This document emphasizes that simply recommending popular items to new users does not offer much benefit, and side-information can enable more personalized recommendations.

Similarly, “An Effective E-Commerce Recommender System Based on Trust and Semantic Information” (Shambour et al., 2021) discusses how sparsity in user-item interactions can affect the similarity measurements necessary for CF, leading to less accurate recommendations.

To combat long-tail issues, “Serendipitous Recommendation in E-Commerce Using Innovator-Based Collaborative Filtering” (Wang et al., 2019) introduces algorithms aimed at overcoming the Matthew Effect, which biases recommendations towards popular items, thus limiting the discovery of new or niche products. This paper underscores the importance of recommending such items to enhance user experience and product visibility.

Moreover, “Discrete Deep Learning Based Collaborative Filtering Approach for Cold Start Problem” (Kalidindi et al., 2019) explores model-based collaborative filtering as an alternative to traditional methods, addressing the cold-start issue by leveraging statistical models for rating predictions. Trust-based CF methods are also discussed in “An Efficient Recommender System Algorithm Using Trust Data” (Rahim et al., 2022), which highlights improvements in recommendation accuracy for users with sparse data.

However, while these solutions focus on specific components of CF-based systems, such as trust or deep learning, many fall short of implementing a full hybrid approach, which could combine other recommendation techniques to address these challenges.

SQ2 - Which evaluation metrics are suitable for assessing the performance of an e-commerce hybrid recommender system?

The findings for this question identified several key evaluation metrics used to assess the performance of hybrid recommender systems in e-commerce, including RMSE (Root Mean Square Error), MAE (Mean Absolute Error), Precision, and Recall.

RMSE is commonly used to evaluate the accuracy of rating predictions by calculating the difference between the predicted and actual ratings, as demonstrated in “Enhance Rating Prediction for E-commerce Recommender System Using Hybridization of SDAE, Attention Mechanism and Probabilistic Matrix Factorization” (Hanafi, 2022).

MAE, which measures the average absolute difference between predicted and actual ratings, is another widely used metric in hybrid recommender systems. It provides effective measure of error, as shown in “Effectual Recommendations Using Artificial Algae Algorithm and Fuzzy C-Mean” (Katarya & Verma, 2017).

Precision is used to measure the proportion of recommended items that are relevant, while recall measures the proportion of relevant items that were successfully recommended. Both metrics were used to assess the quality of recommendations in “Hybrid Collaborative Movie Recommender System Using Clustering and Bat Optimization” (Vellaichamy & Kalimuthu, 2017) and “Effectual Recommendations Using Artificial Algae Algorithm and Fuzzy C-Mean”.

In summary, the combination of RMSE, MAE, Precision, and Recall provides a robust framework for evaluating the performance of e-commerce hybrid recommender systems.

SQ3 - How can association rules be effectively integrated into an e-commerce hybrid recommender system?

Recent literature highlights various methods for integrating association rules into hybrid recommender systems to enhance their performance. Association rules are leveraged to uncover patterns of item co-occurrence, enabling systems to recommend products based on previous customer behaviour. Several approaches combine association rules with other recommendation techniques, such as collaborative filtering, content-based filtering, and even advanced machine learning models, to create more robust hybrid systems or improve baseline recommenders.

An approach is illustrated in **"**Item-Based Collaborative Filtering and Association Rules for a Baseline Recommender in E-Commerce**"** (Lourenco & Varde, 2020), where association rules were integrated with item-based collaborative filtering.

This baseline system uses association rule mining (ARM) to detect patterns in purchase history and applies the Apriori algorithm to generate frequent item sets. The identified patterns are then used to recommend items that users might buy next or as complementary products. The system also employs item-based collaborative filtering using Singular Value Decomposition (SVD) to reduce matrix dimensionality and calculate similarity between products.

This hybrid system effectively combines two perspectives, association rules for product co-occurrence and collaborative filtering for user-product relationships, to enhance recommendation quality in online shopping environments. The proposed approach uses two modules that are presented Figure 12 as a visual explanation of the architecture for this approach.

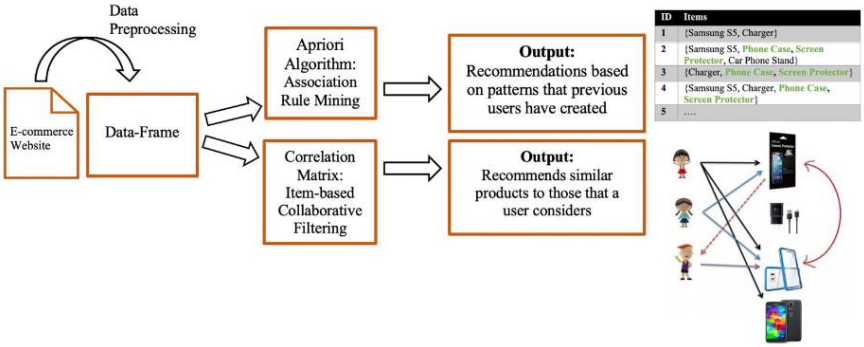

Figure 12 - Recommender system architecture and outputs(Lourenco & Varde, 2020)

Another approach integration of association rules is seen in "Collaborative Recommendation System Using Dynamic Content-based Filtering, Association Rule Mining and Opinion Mining" (Tewari & Barman, 2017), who proposed a dynamic hybrid system that combines association rules with content-based filtering and opinion mining.

In this system, the Association Rule Generator module analyses user purchasing histories to create association rules from items in the top rating quartile. These rules are then applied to recommend products based on similar users' purchasing patterns. This approach integrates association rules within a broader system that includes collaborative filtering and sentiment analysis (opinion mining) to provide more personalized and precise recommendations. By using association rules to identify frequently bought items and content-based profiling to understand user preferences, the system achieves a balance between personalization and pattern-based recommendations. Figure 13 exemplifies the communication between the modules.

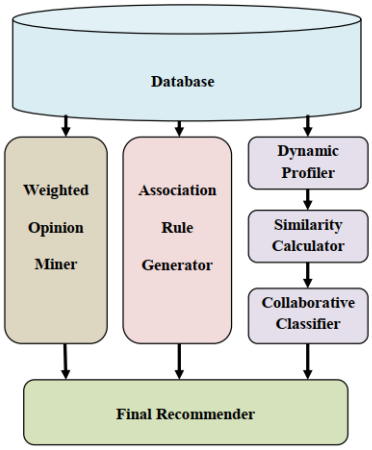

Figure 13 - Recommender system architecture (Tewari & Barman, 2017)

2.3.7 Search conclusion

In this section, the PRISMA methodology was applied to systematically identify and analyse the key challenges and limitations in existing e-commerce recommender systems. The findings provide valuable insights that contribute to the overall objective of this dissertation:

SQ1: What are the key challenges and limitations of existing e-commerce recommender systems?

This search question results revealed several common challenges in existing e-commerce recommender systems, including the cold-start problem for new users and items, data sparsity, and the tendency to recommend only popular items (long-tail issue). These challenges limit the accuracy and diversity of recommendations. These limitations restrict the performance of CF-based models, implying the need for hybrid methods that combine multiple techniques to mitigate these challenges such as content-based filtering or external user data.

For this dissertation intended objective it is important to replicate the user cold-start problem to be able to mitigate it, according to the findings it can be replicated using a Collaborative filtering component. Its implementation in this dissertation is discussed in "Item-based collaborative filtering ".

SQ2 - Which evaluation metrics are suitable for assessing the performance of an e-commerce hybrid recommender system?

The review identified several key evaluation metrics commonly used to assess the performance of hybrid recommender systems. Root Mean Square Error (RMSE), Accuracy, Precision, Recall, and Mean Absolute Error (MAE) are frequently employed to measure the quality and effectiveness of these systems. RMSE is often used to evaluate prediction error, while Precision and Recall assess the relevance and accuracy of recommendations.

For this dissertation, these metrics will be used to evaluate the performance of a hybrid recommender system components, providing insights into both prediction accuracy and recommendation quality. The mentioned metrics can be found the item-based collaborative filtering "Trained models analysis" and Content-based recommender "Recommender evaluation with mocked user interactions" subsections.

SQ3 - How can association rules be effectively integrated into an e-commerce hybrid recommender system?

The integration of association rules into hybrid recommender systems has been shown to enhance recommendation quality. Association rule mining uncovers patterns in user purchasing behaviour, which can be combined with collaborative filtering or content-based approaches to offer more personalized and relevant recommendations.

These findings are useful in shaping the hybrid recommender approach, offering an alternative method to mitigate the user cold-start problem by leveraging purchasing patterns for more effective recommendations. The association rules implementation is discussed in "Association Rule recommender component".

2.4 Chapter remarks

This chapter provides key definitions related to recommender systems, including techniques, challenges, and evaluation methods, as well as an introduction to association rules.

Additionally, the results of a systematic review conducted using the PRISMA methodology were presented. The review highlighted the cold-start problem's impact on collaborative filtering and creating the possibility for this issue to be replicated in the proposed approach in Implementation. Notably, the results of search question 3 confirmed the feasibility of integrating association rules with recommendation techniques.

3 Experimentation

This chapter is organized into four sections. It begins by describing the technologies used in the exploration of the datasets and the development of the hybrid recommender system prototype. The second section provides an overview of the Amazon Review Dataset, including its description and evolution, as well as exploratory data analysis and pre-processing steps. The third section focuses on the evaluation of the approach. Finally, the chapter concludes with an analysis of data protection, security considerations, and the ethical aspects considered.

3.1 Technologies

When considering technologies for implementing the recommender system prototype, the open-source programming language **Python** emerges as a suitable choice owing to its versatility in all implementation stages and the author's familiarity and eagerness to expand the skills with it. To integrate code and documentation. Jupyter Notebook (jupyter.org, 2024) will be used.

The following subsections will present Python libraries used for the data set exploratory analysis and the development of each hybrid recommender component, starting with the common libraries used between the stages.

3.1.1 Common libraries for multiple steps

Some of the common steps, such as manipulating datasets, creating graphs, or connecting to databases, are performed using the same libraries:

Pandas: Dataset manipulation is accomplished using Pandas (McKinney, 2010). Pandas provides data structures like Data Frames, which are essential for handling and processing datasets.

Matplotlib and Seaborn: For visualizing the results, Matplotlib (Hunter, 2007) and Seaborn (Waskom et al., 2017) are used. These libraries offer tools for creating a wide range of static, animated, and interactive visualizations. Seaborn can be used together with Matplotlib and provide an interface to draw graphics.

MongoDB and Pymongo: As a NoSQL database, MongoDB is used for storing and retrieving large volumes of unstructured data. The Pymongo library is employed to facilitate interactions between Python and MongoDB, allowing for efficient data management and querying within the research framework (mongodb.com, 2024).

NumPy: Provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays (Harris et al., 2020).

3.1.2 Exploratory Data Analysis (EDA) and Data pre-processing:

Using public-use datasets as a base implies exploring them to understand the characteristics besides what's available online and based on that improve them by handling missing values, duplicates, or outliers. Besides the standard Python package, other libraries can be used for these types of tasks which were already mentioned in the common libraries.

3.1.3 Association rules component

In addition to the already mentioned common tools and libraries, it was also used for the creation and analysis of Association Rules in this project are as follows:

MLxtend: For creating and experimenting with association rules (Raschka, 2018). This tool offers dedicated packages for implementing the Apriori and FP-Growth algorithms, as well as functions for generating association rules from the frequent itemsets produced by these algorithms. Additionally, the library includes a "TransactionEncoder", which facilitates the encoding of transactions into a format suitable for applying these algorithms.

3.1.4 Content-based recommender component

The content-based recommender component leverages several common libraries for tasks such as text processing, dataset manipulation, feature extraction, similarity computation, visualisation, and database interaction. Adding to the already mentioned common libraries the following libraries are utilised to facilitate these tasks:

Natural Language Toolkit (NLTK): used for various text-processing tasks (Bird et al., 2009).

Scikit-learn: this tool helps to convert text data into a numerical format and measure the similarity between text documents. Additionally, it is used for splitting the dataset into training and testing sets (Pedregosa et al., 2011).

3.1.5 Item-based collaborative-filtering recommender component

The item-based collaborative filtering recommender component also uses several libraries for tasks such as dataset manipulation and database interaction. However, it also needs tools for model training and its evaluation. The following imports are utilized to facilitate these tasks:

Surprise: used for building and evaluating collaborative filtering models (Hug, 2020).

Joblib: used for storing and loading machine learning models (*Joblib: Running Python Functions as Pipeline Jobs*, 2024).

3.2 Dataset

Training and evaluating a hybrid recommender system model requires access to substantial datasets comprising item catalogues, user-item ratings, and relevant properties defining relationships between items. However, in domains like e-commerce, the scarcity of publicly available data, due to the sensitive nature of content and user privacy concerns, as well as loss of business intelligence poses a significant challenge. Despite this, the existence of annual competitions like the ACM RecSys Challenge (RecSys Community, 2024) and Kaggle Competitions (Kaggle Inc, 2024) encourages the evolution of recommender systems, leading to the availability of suitable datasets for research and development purposes. The dataset used in this dissertation was the 2018 version of Amazon Review Data (Julian McAuley, 2018).

3.2.1 Amazon Review Data

The “Amazon Review Data” dataset is a comprehensive collection of data from the Amazon e-commerce platform, released for participation in RecSys’13 competition in 2013 (McAuley & Leskovec, 2013), and later improved versions were released in 2014 (He & McAuley, 2016), 2018 (Ni et al., 2019), and 2023 (Hou et al., 2024). Table 8 represents the dataset evolution.

This dataset is organized into product metadata and user-item reviews and can be accessed either as a complete set or by specific categories such as Automotive, Electronics, or Books.

Both the 2018 and 2023 editions were analysed for this study. However, the 2023 edition lacks data on the “also_buy” property, which is crucial for creating Association Rules as discussed in the Implementation chapter. This issue has been reported, by other users, on the dataset’s GitHub repository but has not been resolved at the time of writing (McAuley-Lab, 2024). Consequently, the 2018 edition was used in this dissertation due to its completeness and reliability.

The full product metadata of the 2018 edition is available in a JSON format file, requiring 12GB of memory. The complete set of reviews is also available in a JSON file, which uses 34GB of memory and contains 233.1 million reviews. Relevant extracted properties regarding product reviews are represented in Table 9

Table 8 - Amazon Review dataset evolution

Version	Data span	Raw review data (size)	Raw review data (quantity)	Products metadata
2013	05/1996 - 03/2013	Not available	34.6 million	2.44 million
2014	05/1996 - 07/2014	20Gb	142.8 million	9.86 million
2018	05/1996 - 10/2018	34Gb	233.10 million	15.17 million
2023	05/1996 - 09/2023	Not available	571.54 million	48.19 million

Table 9 - Relevant properties from user-item reviews

Property	Description
reviewerID	Identification of the reviewer
asin	Identification of the reviewed product
vote	Defines how helpful the review was for other reviewers
reviewText	Comment provided by the reviewer
overall	Rating provided by the reviewer
summary	Summary of the review
reviewTime	Timestamp of when the review was created

Relevant properties of each product representation are available in Table 10 and a sample is available in Figure 14.

Table 10 - Relevant properties from products metadata

Property	Description
asin	Identification of the product
title	Name of the product
feature	Features from the product
description	Product description
price	Price in US dollars
also_buy	List of Ids of products usually bought together
salesRank	Position on the rank of sales
brand	Brand name
categories	Associated categories list

Due to the huge amount of data when using the full version, a specific category subset will be used, Cell Phones and Accessories. This category metadata contains 590071 product entries and 10063255 reviews. The dataset authors also made available smaller user-item ratings subsets for experimentation, only including “asin”, “reviewerID”, “rating” and “timestamp” properties, which were used in this dissertation.


```
{
  "asin": "B00R3HMR8G",
  "title": "iPhone(R) 5/5s Urbanite(TM) Case (Metallic Gold/Black) By: BALLISTIC",
  "feature": [
    "Reinforced Ballistic Corners that provide protection against cracked screens ..."
  ],
  "category": [
    "Cell Phones & Accessories",
    "Cases, Holsters & Sleeves",
    "Basic Cases"
  ],
  "description": [
    "Reinforced Ballistic Corners that provide protection against cracked screens ..."
  ],
  "also_buy": ["B01H2JJ58Y", "B00KZNG3D0"],
  "brand": "BALLISTIC",
  "rank": [
    ">#5,019,564 in Cell Phones & Accessories (See Top 100 in Cell Phones & Accessories)",
    ">#2,142,470 in Cell Phones & Accessories > Cases, Holsters & Clips > Basic Cases"
  ],
  "also_view": ["B007Y08RZE", "B06XJCGHYD", "B00410VD2Y"],
  "main_cat": "Cell Phones & Accessories",
  "date": "February 11, 2011",
  "price": "$6.59"
}
```


Figure 14 – Product metadata entry sample

3.2.2 Amazon Review Data EDA

Analysing the available input data helps to understand its structure and patterns, as well as identify possible noise that could affect the training and evaluation of systems using this data. As mentioned, the dataset was made available in 2 files, one with product metadata and the other with user-item reviews.

Product metadata analysis

The first step involved importing the file in JSON format and analysing the product metadata. This analysis confirmed the structure and the number of items available, totalling 590071, as provided in Code snippet 1.

```
[8]
# Dataset structure
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 590071 entries, 0 to 590070
Data columns (total 19 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   category         590071 non-null  object
 1   tech1            590071 non-null  object
 2   description      590071 non-null  object
 3   fit              590071 non-null  object
 4   title            590071 non-null  object
 5   also_buy         590071 non-null  object
 6   tech2            590071 non-null  object
 7   brand            590071 non-null  object
 8   feature          590071 non-null  object
 9   rank             590071 non-null  object
 10  also_view        590071 non-null  object
 11  details          590071 non-null  object
 12  main_cat         590071 non-null  object
 13  similar_item     590071 non-null  object
 14  date             590071 non-null  object
 15  price            590071 non-null  object
 16  asin             590071 non-null  object
 17  imageURL         590071 non-null  object
 18  imageURLHighRes  590071 non-null  object
dtypes: object(19)
```

Code snippet 1 - Product metadata dataset structure

By examining the unique values of the 'main_cat' property, which represents the main category of each item, it was identified items from other categories, such as books and computers, represented in Code snippet 2. Although these items are not numerous, as shown in Figure 15, they can be removed during the pre-processing phase.

```
[10]
# Unique categories
main_cat = df['main_cat'].unique()
print("Unique categories", main_cat)
```

```
Unique categories ['Movies & TV' 'Books' 'Cell Phones & Accessories'
 'Tools & Home Improvement' 'Portable Audio & Accessories'
 'All Electronics' 'Amazon Home' 'Home Audio & Theater'
 'Health & Personal Care' 'Sports & Outdoors' 'Amazon Devices' 'Computers'
 'Car Electronics' 'GPS & Navigation' 'Automotive' 'Toys & Games' 'Baby'
 '<img src="https://images-na.ssl-images-amazon.com/images/G/01/nav2/images/gui/amazon-fashion-store-new._CB520838675_.png" class="nav-categ-image" alt="AMAZON FASHION"/>'
 'Camera & Photo' 'Office Products' 'All Beauty' 'Pet Supplies'
 'Musical Instruments' 'Video Games' 'Software' 'Industrial & Scientific'
 'Grocery' 'Arts, Crafts & Sewing' '' 'Fire Phone' 'Appliances'
 '<img src="https://m.media-amazon.com/images/G/01/nav2/images/gui/amazon-fashion-store-new._CB520838675_.png" class="nav-categ-image" alt="AMAZON FASHION"/>'
 'Apple Products' '3D Printing'
 '<img src="https://images-na.ssl-images-amazon.com/images/G/01/handmade/brand/logos/2018/subnav_logo._CB502360610_.png" class="nav-categ-image" alt="Handmade"/>'
 'Amazon Fire TV']
```

Code snippet 2 - List of unique product categories

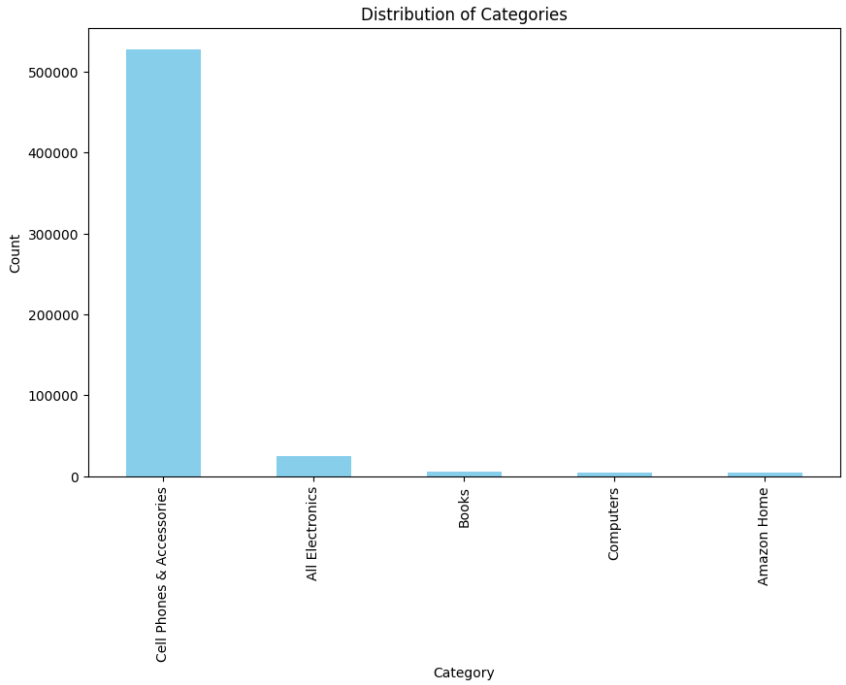

Figure 15 - Distribution of product categories on the metadata file

Another noteworthy analysis was the distribution of product brands. As depicted in Figure 16, the brand with the highest item count is “Generic.”

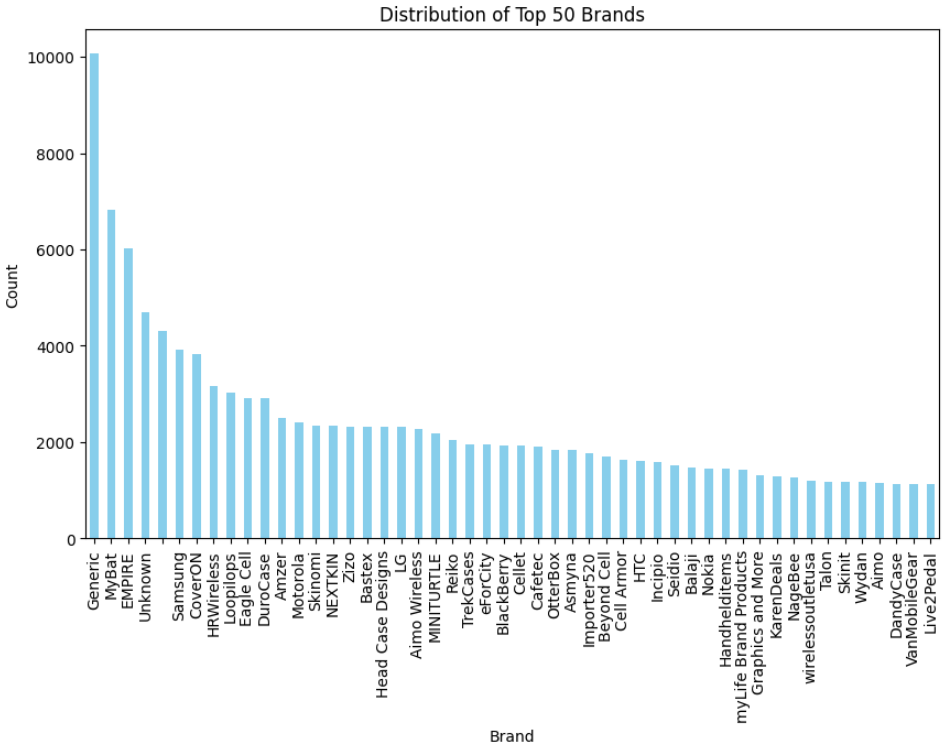

Figure 16 - Top 50 brand distribution

User-item reviews analysis

The first step involved importing the respective file in JSON format and analysing the user reviews. Like the previous dataset file, it was confirmed the structure, however, the properties didn't have attributed names, so it was done manually. It was also possible to confirm the number of items available, 10063253, as illustrated in Code snippet 3.

```
df_ratings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10063254 entries, 0 to 10063253
Data columns (total 4 columns):
 #   Column     Dtype
---  ------     -----
 0   id         object
 1   user_id    object
 2   rating     float64
 3   timestamp  int64
dtypes: float64(1), int64(1), object(2)
memory usage: 307.1+ MB
```

Code snippet 3 - User-item reviews dataset structure

Since the less frequently reviewed items will be removed during the pre-processing phase, the products that had the highest number of reviews were identified in Code snippet 4.

```
item_counts = Counter(df_ratings['id'])

sorted_counts = dict(sorted(item_counts.items(), key=lambda x: x[1], reverse=True))
sorted_counts
```

```
{'B00MQSMDYU': 13543,
 'B005NF5NTK': 13236,
 'B00X5RV14Y': 13031,
 'B00QN1T6NM': 11227,
 'B014EB532U': 10728,
 'B00AANQLRI': 10132,
 'B00MXWFUQC': 10100,
 'B00G7UY3EG': 9282,
 'B00VH88CJ0': 9038,
 'B018JW3EOY': 8890,
 'B00P7N0320': 8829,
 'B00BT8L2MW': 8760,
 'B0092KJ9BU': 8525,
 'B0194WDVHI': 8307,
```

Code snippet 4 – Ordered products based on reviews count

A significant imbalance was observed between the items with the most and least reviews as it's visible in Figure 17. Through trial and error, a minimum threshold of 500 reviews was established to define frequent items. This threshold will be crucial for creating association rules later.

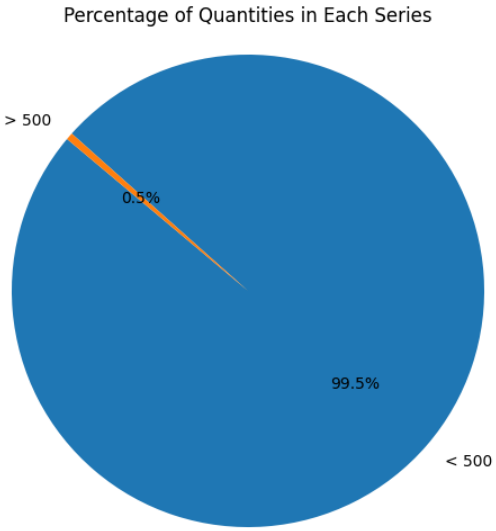

Figure 17 – Product reviews count over and under a defined threshold

Additionally, the distribution of ratings for each review was analysed, Figure 18, revealing that 20.2% of the ratings were negative, with the remainder being positive. This finding raises an important consideration for the recommendation system: whether it is appropriate to recommend products that a significant portion of users have rated poorly.

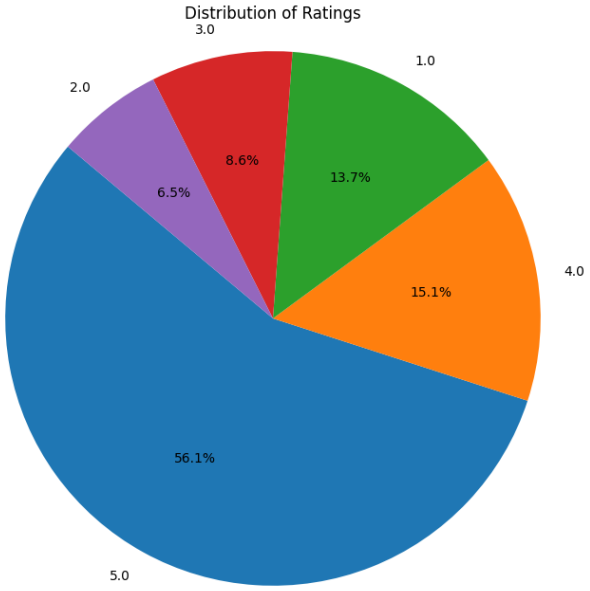

Figure 18 – User rating distribution

3.2.3 Amazon Review Data pre-processing

Using the findings during the EDA phase, some pre-processing actions were taken on both files, to keep the essential information only.

Product metadata pre-processing

After completing the EDA phase and continuing to work with the loaded JSON file, it was identified that several categories other than 'Cell Phones & Accessories' were present. These categories were filtered out to focus exclusively on the relevant data.

The property identifying each entry, 'asin', was renamed to 'id', and all entries considered duplicates based on this identifier were removed.

To optimize memory usage for subsequent processes, unused properties were filtered out, retaining only 'title', 'brand', 'asin', and 'also_buy'. This reduction decreased the file size from 85.5 MB to 16.1 MB. Finally, the processed file was exported to CSV format and stored for future use.

User-item reviews pre-processing

It was identified that many products had few reviews, introducing noise into subsequent processes. Through trial and error, a minimum threshold of 500 reviews was established to define frequent items, and entries considered infrequent were removed.

Using a composite key with "user_id" and "id" properties, it was possible to identify almost 30 thousand duplicated ratings.

To optimize memory usage for subsequent processes, unused properties were filtered out, retaining only 'id', 'user_id', and 'rating'. This reduction decreased the file size from 313.8 MB to 84.1 MB. Finally, the processed file was exported to CSV format and stored for future use.

3.3 Evaluation methods

Recommender system evaluation methods were part of the State of the Art chapter, since their definitions to the metrics used in recent literature.

The proposed approach described in Implementation contains different recommender components that can be measured using offline testing methods such as Accuracy and Recall for the Content-based filtering component and RMSE and MAE for the item-based Collaborative filtering component. The association rules created and used by the respective component used Confidence and Support as quality metrics.

It was also possible to perform two sets of interviews with 10 test users to study the relevance of the weights attributed to each recommender system component and obtain the opinion of the recommendation's relevance.

3.4 Data protection, security analysis and ethical aspects

In the development of recommender systems, it is essential to address key aspects related to data protection, system security, and ethical considerations. This section explores the necessary adherence to data privacy regulations, analyses potential security vulnerabilities in recommender systems, and highlights the moral challenges, including user privacy, transparency, fairness, and the societal impacts of recommendation algorithms.

3.4.1 Data protection

This dissertation approach uses public datasets and it adheres to the General Data Protection Regulation (GDPR) of the European Union (European Union, 2016) by using the collected data only necessary for research purposes.

3.4.2 Recommender systems security analysis

E-commerce platforms heavily rely on recommender systems to provide users with personalized recommendations and enhance their overall experience. However, the increasing exposure of these systems to the public domain makes them susceptible to malicious attacks, particularly those leveraging user feedback mechanisms like Collaborative Filtering recommendations.

Recent studies (Balasubramaniam & Chidambaram, 2018;Du et al., 2019;Mingxun et al., 2023), and (Yang, 2019) mention the vulnerability of recommender systems to manipulation by malicious users, who craft deceptive data intending to degrade performance while remaining undetected. These attacks pose a threat to the integrity of the recommendation process and erode user trust in these platforms.

3.4.3 Recommender systems ethical aspects

While addressing these security concerns is crucial, ethical challenges in recommender systems go beyond protecting against malicious intent. Karakolis et al. (2022), in the paper “Identifying and Addressing Ethical Challenges in Recommender Systems” explores the ethical dimensions of overall recommender systems. The authors delve into universally accepted ethical implications by focusing on Recommender systems behaviour and its outcomes, particularly those that harm stakeholders’ utility or potentially violate rights. The following ethical challenges are identified:

Privacy: The lack of consent by the user for the collection of their information endangers the user’s privacy. Recommender systems can collect user information through third-party tracking cookies, often accepted through vague “terms and conditions.” Additionally, these systems possess the capability to deduct data based on the information they already have. This raises

concerns about users being unaware of the data being collected and the potential use of their information without explicit consent.

Inappropriate Content: Recommender systems need to be prepared to filter out content that might be hurtful to the user, such as avoiding recommending content to underage users or hateful material.

Personal Identity: Recommender systems create dynamic user profiles by combining user information with existing data. However, this approach introduces a risk where the user profile might differ significantly from the user's reality and preferences, leading to potentially incorrect recommendations and violating user autonomy in the platform.

Transparency: The collection, processing, and use of user data by the recommender system are kept secret to maintain competitive advantages, making it unclear for the user what is happening behind the scenes. This lack of transparency makes it impossible for users to give informed consent.

Fairness: When the system collects data or training data, reflects existing social patterns, or has population imbalances, it can also reflect the existing society bias when providing recommendations targeting a specific group of users. (Liu et al., 2022)

Social Implications: The social implications of recommender systems become evident when users are confined to a limited set of opinions that align with their existing preferences. This confinement may result in the creation of "filter bubbles" or "echo chambers," isolating users from diverse perspectives. The malicious exploitation of such isolation can significantly impact users, influencing their viewpoints and potentially reinforcing pre-existing beliefs, thereby adversely affecting their overall experience on the platform.

3.5 Chapter remarks

This chapter outlines the technologies used in the experimental phase of this dissertation, focusing on both exploratory data analysis and the development of the recommender system.

The Amazon Reviews dataset was introduced, along with the EDA and pre-processing performed on the 2018 version. Although a 2023 version of the dataset exists, it was not employed due to its lack of the necessary properties for generating association rules.

4 Implementation

This chapter presents a comprehensive overview of the weighted hybrid recommender system architecture, followed by a detailed description of the implementation of each module.

It begins by explaining the creation of association rules and then introduces the collaborative filtering and content-based filtering modules.

Additionally, the integration of these modules into the overall system is discussed. The chapter concludes with an analysis of the relevance and effectiveness of the system's recommendations by test users.

4.1 Weighted hybrid recommender system overview

The proposed approach combines content-based filtering, item-based collaborative filtering, and association rule-based recommenders. Each component using the same user profile, generates recommendations, which are then combined using defined weights and integrated into a final set of personalized recommendations, the architecture is represented in Figure 19.

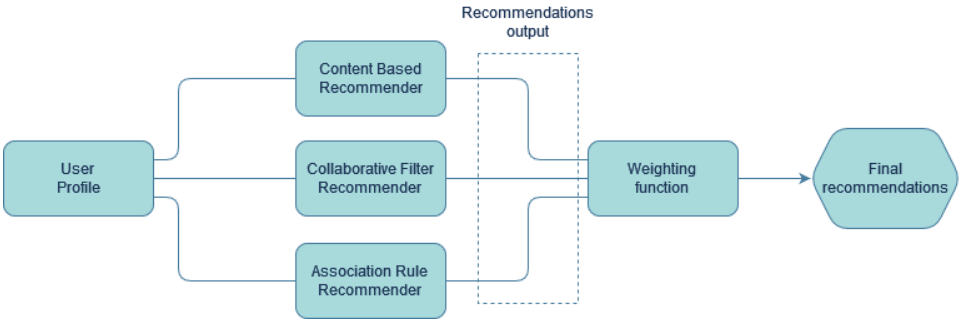

Figure 19 - Weighted Hybrid recommender basic architecture

This weighted approach offers several advantages. It simplifies integration by allowing different recommendation methods to contribute proportionally, ensuring a balanced influence based on their strengths. Additionally, adjusting the weights is straightforward, enabling fine-tuning to match specific application needs or user feedback. This flexibility, combined with the ease of integrating diverse recommendation techniques, makes the weighted approach both powerful and adaptable for delivering personalized recommendations.

4.2 Association Rule recommender component

This section outlines the development of an Association Rule recommender component, which is also represented in Figure 20. The process starts with data preprocessing, followed by parameter selection for the FP-Growth and Apriori algorithms. After creating and storing the created association rules in MongoDB, the results are interpreted to build the final recommender system.

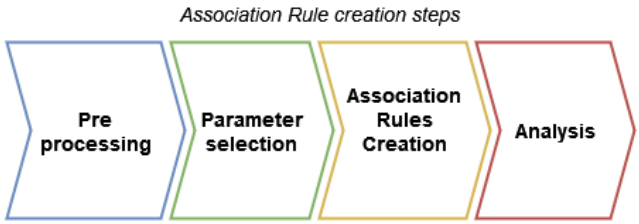

Figure 20 - Association Rule recommender creation steps

4.2.1 Preparation for Association Rules creation

This process began with importing the pre-processed datasets. Due to hardware constraints, a sample of 100,000 review entries was selected.

Next, this sample was merged with the product metadata dataset using the “asin” property as the primary key, ensuring a relationship between the reviews and their corresponding metadata.

Following the approach described by Lourenco & Varde (2020) in the paper “Item-Based Collaborative Filtering and Association Rules for a Baseline Recommender in E-Commerce”, the “also_buy” property was employed to generate association rules. This property indicates items frequently bought together, which helps discover consumer purchasing patterns.

The initial step involved filtering out entries lacking elements in the “also_buy” list, ensuring that only relevant transactions were kept. A new property, “bought_items”, was then created. This property concatenated the ID of each record with the IDs of the items listed in the “also_buy” property, compiling all associated purchases into a single field. Code snippet 5 contains an example of this property from the top 5 elements of the Data Frame.

```
merged_data_also_buy["bought_items"].head()
```

```
0    [B0194WDVHI, B01LWQAHUX, B01NCL75QK, B07D8NRL7...
1    [B00P6PWNW2, B01F3JC0NM, B00OZGN0XY, B0772MFV4...
2    [B00EF1OGOG, B00HA5RXYY, B01KC823F4, B018AJLKB...
6    [B016MNSRP8, B07JQQXJFN, B079M28DZ1, B07FPFZFP...
7    [B01FOAP73A, B01KLNJF7O, B07BZ31LWS, B07FSBYHR...
Name: bought_items, dtype: object
```

Code snippet 5 - “Bought_items” property sample

To facilitate the next stages, a new Pandas Data Frame was constructed from these concatenated transactions. This Data Frame was then used to apply the TransactionEncoder from the mlxtend library, performing one-hot encoding on the transactions. One-hot encoding transforms the transaction data into a binary matrix format, which is required for the application of Apriori and FP-Growth mining algorithms, detailed in the subsequent sections.

With the data encoded, the Apriori and FP-Growth algorithms were applied to identify frequent item sets and generate association rules. Code snippet 6 shows the result of this encoding operation, providing a visual representation of the transformed dataset.

```
df_encoded = pd.DataFrame(te_ary, columns=te.columns_)
df_encoded
```

	0062378074	0062422502	0062871315	0133940713	0195612701	024134848X	030788631X	0316204269	0316449571	0393123960	...	B07MP9ZL7P
0	False	False	False	False	False	False	False	False	False	False	...	False
1	False	False	False	False	False	False	False	False	False	False	...	False
2	False	False	False	False	False	False	False	False	False	False	...	False
3	False	False	False	False	False	False	False	False	False	False	...	False
4	False	False	False	False	False	False	False	False	False	False	...	False
...	...	...	...	...	...	...	...	...	...	...	...	...
72746	False	False	False	False	False	False	False	False	False	False	...	False
72747	False	False	False	False	False	False	False	False	False	False	...	False
72748	False	False	False	False	False	False	False	False	False	False	...	False
72749	False	False	False	False	False	False	False	False	False	False	...	False
72750	False	False	False	False	False	False	False	False	False	False	...	False

72751 rows × 26896 columns

Code snippet 6 - One-hot encoded transactions data frame

4.2.2 Parameter selection tests for FP-Growth algorithm

The “fpgrowth” package from the mlxtend library requires several parameters to generate frequent itemsets. These parameters include the input data frame, the support parameter, and the maximum length of itemsets. Additionally, the “association_rules” package from mlxtend requires the frequent itemsets generated by “fpgrowth”, an evaluation metric, and a minimum threshold for that metric as parameters.

A range of values was evaluated to decide the optimal support and confidence values. Specifically, support values between 1% and 5%, and confidence values between 10% and 90% were evaluated. The specific arrays used for these tests are as follows in Code snippet 7.

```
supports_array = [0.05, 0.03, 0.01]
confidence_array = [0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1]
```

Code snippet 7 - support and confidence arrays for FP-Growth

The support values were iterated over in a parent loop, and for each iteration, the confidence values were iterated over in a nested loop as the minimum confidence parameter. Due to hardware limitations, other potential support and confidence values were excluded from the analysis.

Figure 21 graphically illustrates the effect of these parameters on the performance and quality of the association rules generated. This visualization helps in understanding how different support and confidence thresholds impact the resulting rules, allowing for a more informed selection of parameters.

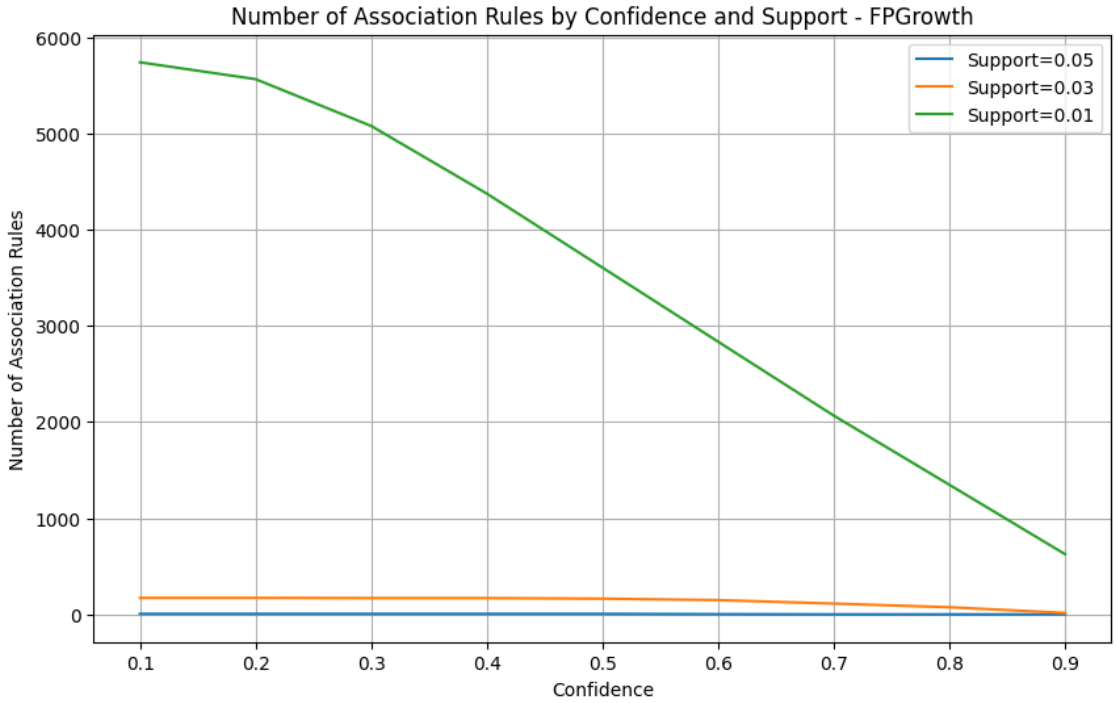

Figure 21 - Number of created Association Rules using the FP-Growth algorithm

4.2.3 Parameters selection tests for the Apriori algorithm

The “apriori” package from the mlxtend library was also used to generate frequent itemsets. Similar to the “fpgrowth” package, the “apriori” package requires several parameters, including the input DataFrame and the support parameter. The “association_rules” package from mlxtend is then used with the frequent itemsets generated by “apriori, along with an evaluation metric and a minimum threshold for that metric.

To determine the optimal support and confidence values for the Apriori algorithm, a range of values was tested. Specifically, support values between 2% and 5%, and confidence values between 10% and 90% were evaluated.

The specific arrays used for these tests are as follows in Code snippet 8.

```
supports_array = [0.05, 0.03, 0.02]
confidence_array = [0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1]
```

Code snippet 8 - support and confidence arrays for Apriori

The support values were iterated over in a parent loop, and for each iteration, the confidence values were iterated over in a nested loop as the minimum confidence parameter. Due to hardware limitations, other potential support and confidence values were excluded from the initial analysis.

Figure 22 graphically illustrates the effect of these parameters on the performance and quality of the association rules generated by the Apriori algorithm. This visualization helps in understanding how different support and confidence thresholds impact the resulting rules, allowing for a more informed selection of parameters.

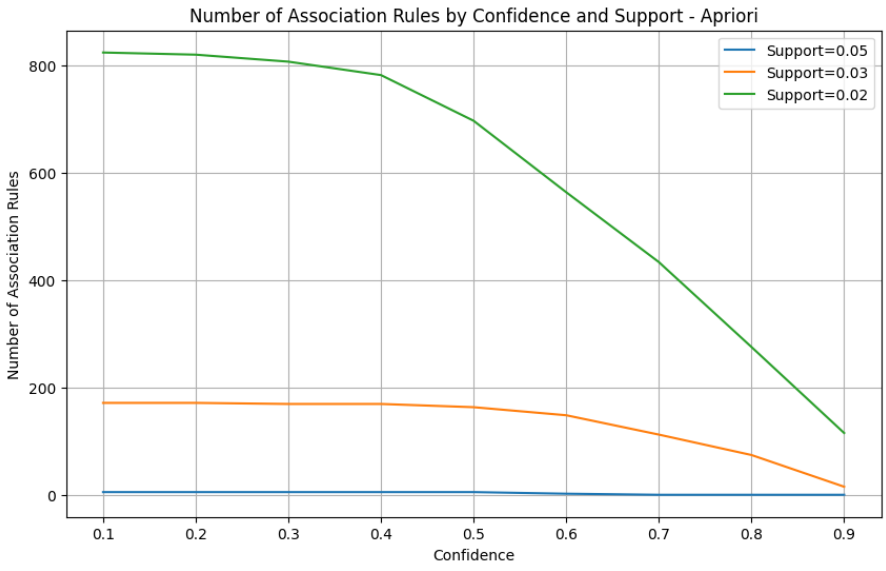

Figure 22 - Number of created Association Rules using the Apriori algorithm

4.2.4 Association rules creation

Following the parameter analysis, the final parameters were applied to frequent item set generation and the creation of association rules with both FP-Growth and Apriori algorithms.

FP-Growth:

It was selected 1% for support and 70% confidence which resulted in 2105 rules created as represented in Code snippet 9.

```
support = 0.01
confidence_threshold = 0.7
frequent_itemsets_fpgrowth = fpgrowth(
    df_encoded,
    min_support=support,
    use_colnames=True,
    max_len=max_length_generated_itemsets,
)
fpgrowth_rules = association_rules(
    frequent_itemsets_fpgrowth,
    metric=current_metric,
    min_threshold=confidence_threshold,
)
fpgrowth_rules
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
0	(B013JMBAMC)	(B071YMZ4LD)	0.054405	0.075023	0.041951	0.771097	10.278132	0.037870	4.040904	0.954643
1	(B072HHW3GK)	(B071YMZ4LD)	0.041841	0.075023	0.034556	0.825887	11.008447	0.031417	5.312509	0.948862
2	(B01CU1EC6Y)	(B01NCL75QK)	0.038295	0.042020	0.028467	0.743360	17.690598	0.026858	3.732772	0.981042
3	(B01MV2YGRR)	(B071YMZ4LD)	0.037979	0.075023	0.034281	0.902642	12.031534	0.031432	9.500786	0.953082
4	(B01MV2YGRR)	(B072HHW3GK)	0.037979	0.041841	0.028302	0.745204	17.810240	0.026713	3.760500	0.981114
...	...	...	...	...	...	...	...	...	...	...
2100	(B01HVG45FQ)	(B01E9ILNUE)	0.010570	0.023876	0.010570	1.000000	41.883132	0.010318	inf	0.986552
2101	(B00ONYDCQI)	(B01E9ILNUE)	0.011423	0.023876	0.011134	0.974729	40.824713	0.010861	38.626623	0.986777
2102	(B00ONYDCQI)	(B072KJRJ45)	0.011423	0.021855	0.010941	0.957882	43.828225	0.010692	23.223948	0.988475
2103	(B00ONYDCQI)	(B01KA3A1RI)	0.011423	0.022570	0.010941	0.957882	42.440243	0.010684	23.206978	0.987720
2104	(B00ONYDCQI)	(B00NF4SOFQ)	0.011423	0.015381	0.010667	0.933815	60.711306	0.010491	14.876694	0.994893

2105 rows × 10 columns

Code snippet 9 - Association rules creation code and results using FP-Growth algorithm

Apriori:

For Apriori, it was selected 2% for support and 10% confidence which resulted in 902 rules created which are represented in Code snippet 10.

```
support = 0.02
confidence_threshold = 0.1

frequent_itemsets_apriori = apriori(
    df_encoded,
    min_support=support,
    use_colnames=True,
    max_len=max_length_generated_itemsets,
)
apriori_rules = association_rules(
    frequent_itemsets_apriori, metric=current_metric, min_threshold=confidence_threshold
)
apriori_rules
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
0	(B009YCP1LS)	(B00BT8L2MW)	0.052453	0.041608	0.030996	0.590933	14.202498	0.028814	2.342873	0.981049
1	(B00BT8L2MW)	(B009YCP1LS)	0.041608	0.052453	0.030996	0.744962	14.202498	0.028814	3.715318	0.969947
2	(B009YCP1LS)	(B00DD62R40)	0.052453	0.030831	0.020838	0.397275	12.885478	0.019221	1.607977	0.973454
3	(B00DD62R40)	(B009YCP1LS)	0.030831	0.052453	0.020838	0.675881	12.885478	0.019221	2.923450	0.951736
4	(B009YCP1LS)	(B00HMCHUCU)	0.052453	0.042240	0.029842	0.568920	13.468768	0.027626	2.221770	0.977001
...	...	...	...	...	...	...	...	...	...	...
897	(B07F1DYK72)	(B076MP5QX8)	0.038900	0.032302	0.025677	0.660071	20.434384	0.024420	2.846762	0.989556
898	(B077263XQS)	(B07F1DYK72)	0.025278	0.038900	0.022625	0.895052	23.009153	0.021642	9.157841	0.981345
899	(B07F1DYK72)	(B077263XQS)	0.038900	0.025278	0.022625	0.581625	23.009153	0.021642	2.329783	0.995254
900	(B07DFWKBF7)	(B07DHLZ7Z2)	0.033030	0.034597	0.020646	0.625052	18.066412	0.019503	2.574764	0.976917
901	(B07DHLZ7Z2)	(B07DFWKBF7)	0.034597	0.033030	0.020646	0.596742	18.066412	0.019503	2.397894	0.978502

902 rows × 10 columns

Code snippet 10 - Association rules creation code and results using Apriori algorithm

4.2.5 Storing the association rules on MongoDB

To facilitate future use, the created association rules were stored in a MongoDB database collection. A new MongoDB client was instantiated to establish a connection to the database instance, allowing access to the collections created for the association rules created with both algorithms: “association_rules_apriori” and “association_rules_fpgrowth”. Upon successful connection, any existing association rules in the collections were removed to ensure that only the most recent rules were stored.

The Pandas DataFrame containing the association rules generated for both algorithms were converted to a dictionary format compatible with MongoDB’s BSON (Binary JSON) storage format. These converted dictionaries were subsequently inserted into the respective collections in the MongoDB database.

Figure 23 illustrates the structure of the stored association rules from the FP-Growth algorithm within the MongoDB collection.

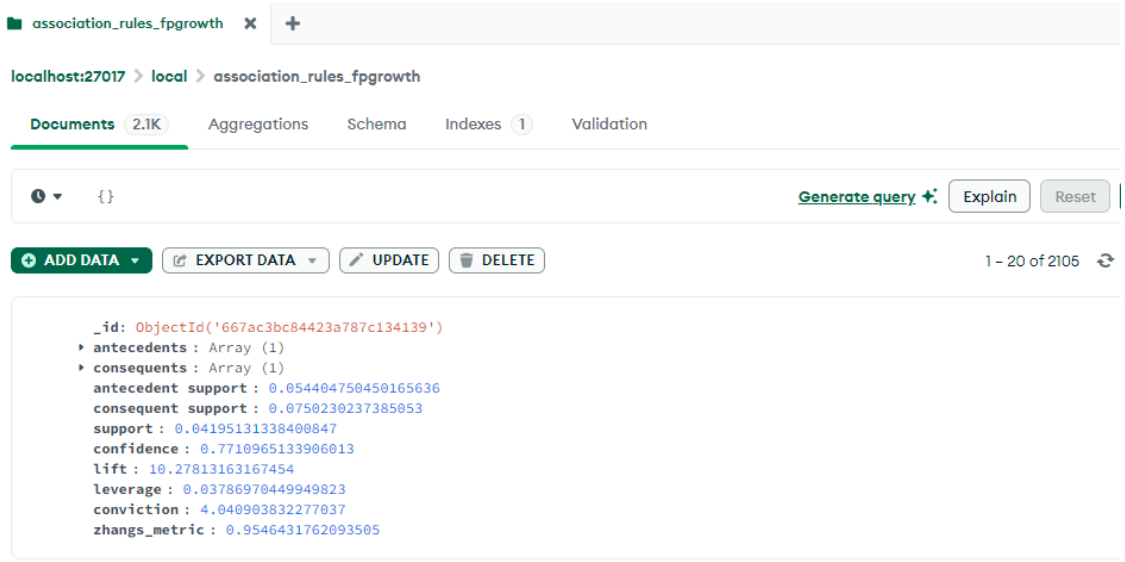

Figure 23 - MongoDB “association_rules_fpgrowth” collection sample

4.2.6 Association Rules interpretation

This sub-section provides the interpretation of the generated association rules. By analyzing various visualizations and metrics, such as correlograms and heatmaps, it’s possible to understand the relationships between item sets, specifically their antecedents and consequents. Additionally, the trade-off between support and confidence will be examined to evaluate the strength and reliability of these rules.

4.2.6.1 Created association rules properties correlograms

The created association rules contain properties such as other metrics besides the ones used to generate them. Using Spearman correlation, it was interesting to see how these properties correlate between them and observe how these correlations change across different algorithms.

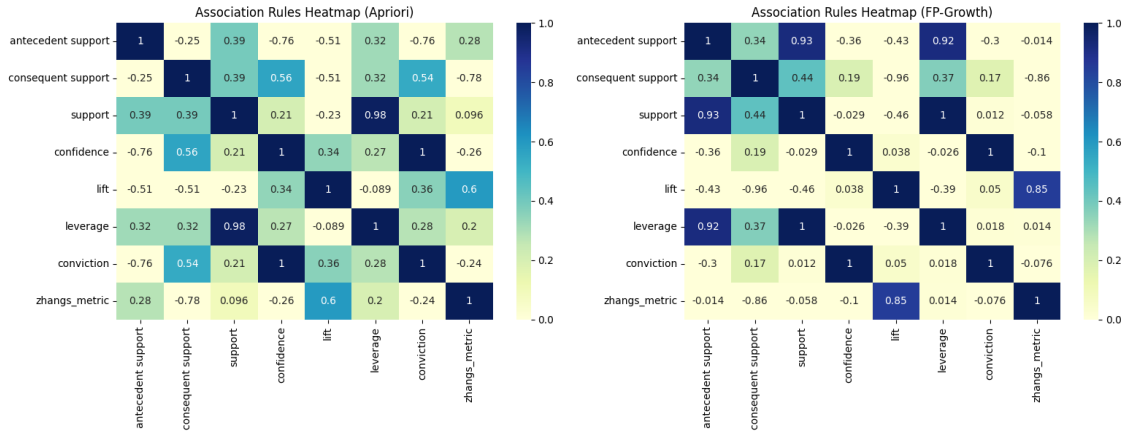

Figure 24 - Correlogram using created association rules properties

The correlograms in Figure 24 illustrate the Spearman correlations between the properties of association rules created by Apriori and FP-Growth algorithms.

Common observations:

- There is a strong positive correlation between “leverage” and “support”, suggesting that rules with high support tend to also have higher leverage.

- Conviction and confidence have the maximum value for a positive correlation on both algorithms, meaning they replicate each other.

- The strong positive correlation between “lift” and “zhangs_metric” indicates that these two metrics measure similar properties of the association rules.

FP-Growth Algorithm:

- There is a strong positive correlation between “support” and “antecedent_support”. This indicates that rules with higher overall support also tend to have higher support for their antecedents.

- There is also a strong positive correlation between “leverage” and “antecedent_support” which indicates that these properties are also closely related.

4.2.6.2 Antecedents and Consequents heatmaps

Another interesting relationship to analyse is between the antecedents and the consequents of an association rule using the confidence metric. For this analysis, a sample of 50 association rules was selected from those created by the Apriori and FP-Growth algorithms, as represented in Figure 25.

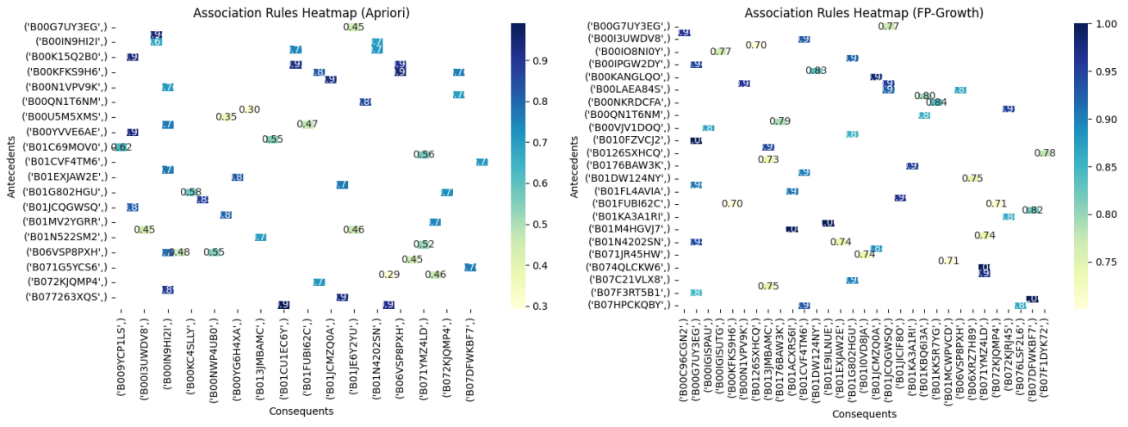

Figure 25 - Antecedents and Consequents heatmap

The heatmap for the Apriori algorithm shows confidence values starting as low as 30%. This indicates that the rules generated by Apriori have a wider range of confidence levels, from very low to high, suggesting more variability in the strength of the rules.

The FP-Growth heatmap, on the other hand, shows that the minimum confidence level starts at 70%. This suggests that the FP-Growth algorithm produces rules with consistently higher confidence levels.

4.2.6.3 Support vs Confidence

Finally, using a scatterplot, it's also possible to analyse the relationship between the support and confidence of the association rules generated by both the Apriori and FP-Growth algorithms. All the created association rules from both algorithms were included in this analysis, and the results are presented in Figure 26 where each point represents an association rule. The x-axis represents the support of the association rules and it's the confidence is represented on the y-axis.

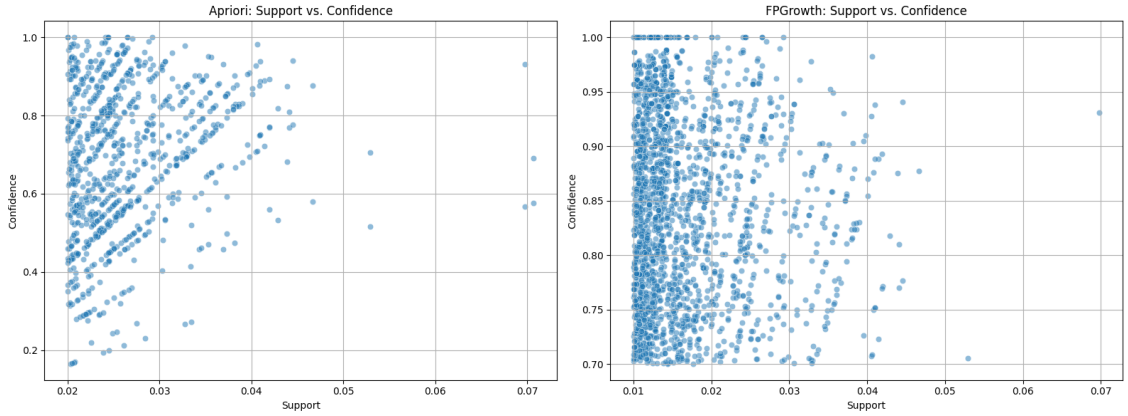

Figure 26 - Support vs Confidence scatterplot

Density - It's possible to observe that FP-Growth shows a higher density compared to Apriori since it has a higher number of association rules than Apriori.

Confidence - On the x-axis, the confidence level starts at a higher level compared to Apriori.

Both algorithms created association rules with 100% confidence.

Distribution - On Apriori, it's possible to observe that most generated association rules have a support lower than 3% and a confidence over 60%.

On FP-Growth most association rules have a support lower than 2%, which is expected since the confidence starts at 70%.

4.2.7 Association rule final recommender

The association rule recommender, during its initialization, connects to the MongoDB instance and loads the stored association rules generated with both algorithms.

It has a public "*recommend_items()*" function that receives the user profile basket and the algorithm name as a parameter. When this function is called, it checks if any association rule antecedents match the user profile basket, if so, the rules get sorted by the confidence metric and the respective association rule consequents are returned as a list.

An example in Code snippet 11, shows a user with one object on the basket and gets a recommendation from a matching the association rule generated using the FP-Growth algorithm.

```
user_basket = ['B013JMBAMC']

recommend_items(user_basket,'fpgrowth');
```

```
Top recommendations based on association rules: ('B071YMZ4LD',)
```

Code snippet 11 – Association rules recommender "recommend_items()" function results example

4.3 Item-based collaborative filtering component

This section details the development of an item-based collaborative filtering recommender, which predicts user preferences and recommends items based on past user ratings and the ratings of similar users. The models are trained using algorithms from the Surprise library, specifically SVD, SVD++, and NMF.

The performance of these trained models is evaluated through cross-validation, ROC curves, and Precision-Recall curves. This comprehensive analysis led to the final implementation of the item-based collaborative filtering recommender function. Figure 27 provides a summary of the steps involved in this process.

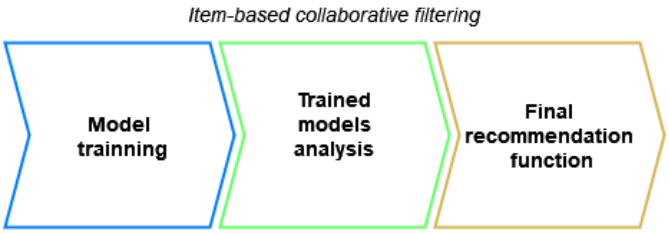

Figure 27 - Item-based collaborative filtering steps

4.3.1 Model training using Surprise library

The first step in building the item-based collaborative filtering recommender involved training a model using an algorithm suited for matrix factorization. The Surprise library offers various algorithms for this purpose, such as Singular Value Decomposition (SVD), Non-negative Matrix Factorization (NMF), and SVD++, an extension of SVD that incorporates implicit ratings. In this step, the results of these three algorithms are explored.

The process begins by loading the cleaned reviews dataset, focusing on the “user_id”, “id”, and “rating” properties. The data is then split into a training set and a test set, with 80% of the data used for training and 20% reserved for testing and evaluation.

Each algorithm is then trained using the same training set. After the training phase, the models are stored for the analysis step, where their performance will be evaluated and compared.

4.3.2 Store unique reviewed items to MongoDB

To enable real-time item recommendations, it is essential to retrieve all unique reviewed product IDs. This allows the model to predict how a user would rate each unique item during runtime. After identifying the unique IDs, they are stored in a MongoDB collection named “all_reviewed_items” for future use. This storage step ensures that the recommender system has quick access to the list of items for making predictions and recommendations.

4.3.3 Trained models analysis

To analyse the trained models, the process begins by loading the models for each of the three algorithms, SVD, NMF, and SVD++, into the system and running a cross-validation procedure for each one.

This procedure evaluates each model's accuracy by calculating metrics such as Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE), as well as measuring computation times. For this analysis, 5 splits were used as a parameter. The results of the cross-validation for each algorithm are summarized in

Table 11 providing the mean and standard deviation of each metric across the five folds.

Table 11 - Cross-validation result of the trained models

Metric	**SVD**	**NMF**	**SVD++**
Mean RMSE (testset)	0.6164 +/- 0.0007	0.6163 +/- 0.0008	0.6164 +/- 0.0007
Mean MAE (testset)	0.4852 +/- 0.0004	0.4850 +/- 0.0004	0.4852 +/- 0.0004
Mean Fit time (seconds)	36.92 +/- 3.74	36.15 +/- 1.70	36.92 +/- 3.74
Mean Test time (seconds)	2.98 +/- 1.22	3.24 +/- 1.68	2.98 +/- 1.22

Both mean RMSE and mean MAE metric values across the algorithms can be considered consistent, differing only in the fourth decimal place. Additionally, the SVD values match the SVD++ values, indicating similar performance between these two algorithms.

The mean time taken to train the model represented as "Mean Fit time," varies slightly across the algorithms, with SVD and SVD++ having identical values. This suggests that the training times for SVD and SVD++ are consistent and similar.

The mean time taken to evaluate the model on the test set, represented as "Mean Test time," also varies slightly across the algorithms, with NMF being slightly slower on average. The higher standard deviation for NMF indicates more variability in the time taken to evaluate the model on the test set.

Overall, while all three algorithms show similar performance in terms of RMSE and MAE, NMF exhibits a slight edge in predictive accuracy.

Precision-Recall curve

The Precision-Recall curve was created for the three algorithms, represented in Figure 28. To calculate the True Positives required for both metrics, a minimum threshold of 3.5 was set within a prediction range of 0 to 5. The algorithms exhibit a similar pattern where precision decreases as recall increases. Despite this trade-off, the precision remains near 90%, indicating that the models maintain a high level of accuracy.

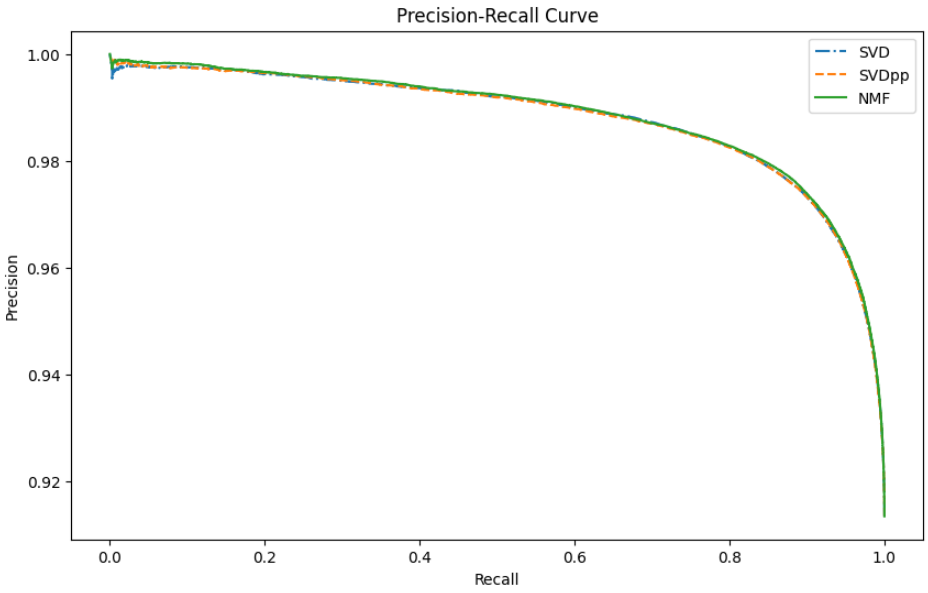

Figure 28 - Precision-Recall curve for each algorithm

Receiver Operating Characteristic curve

Figure 29 shows the Receiver Operating Characteristic (ROC) curves for the three algorithms. These curves depict the relationship between the True Positive Rate (sensitivity) and the False Positive Rate across various threshold settings. All three algorithms display similar patterns, with the curves closely approaching the top-left corner of the plot, indicating high accuracy in distinguishing between positive and negative predictions. The area under the ROC curves (AUC) is consistently high for each algorithm, also demonstrating their strong performance within the 0 to 5 range.

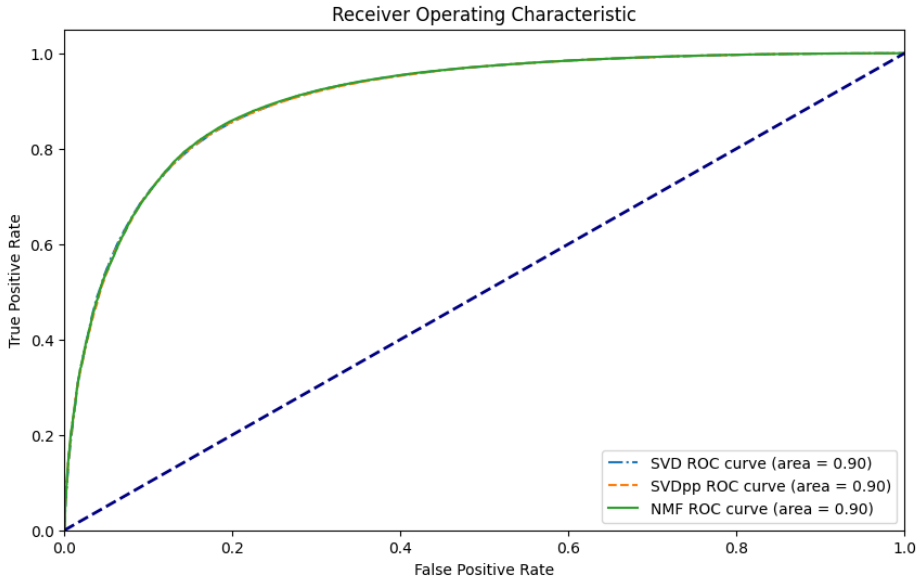

Figure 29 - ROC curves for the three algorithms

Conclusion

Through cross-validation and graphical analysis, it became evident that all three algorithms exhibit similar performance. The consistency of the SVD algorithm's results, validated by the SVD++ algorithm, reinforces its reliability. Given its proven effectiveness and historical significance, popularized by Simon Funk during the Netflix Prize contest in 2006 (Simon Funk, 2006), the SVD algorithm was selected for real-time use in the recommender system.

4.3.4 Item-based collaborative filtering final recommender

During its initialization, the item-based collaborative filtering recommender connects to the MongoDB instance to load the stored unique reviewed items and prepares them for use. It also loads the trained SVD model into the system.

The public function "recommend_items()", Code snippet 12, accepts a user profile ID as a parameter. When this function is called, the system predicts the user ratings for all the unique item IDs using the trained SVD model. The predicted ratings are then sorted in descending order, and by default, the function returns the top 10 items with the highest estimated ratings.

```
28      def recommend_items(self, user_id, n=10):
29          predictions = [self.algo.predict(user_id, item_id) for item_id in self.all_item_ids]
30
31          predictions.sort(key=lambda x: x.est, reverse=True)
32
33          return [(pred.iid, pred.est) for pred in predictions[:n]]
34
```

Code snippet 12 – Item-based collaborative filter recommend_items() function

4.4 Content-based recommender component

This section outlines the development of the content-based filter, which recommends items based on the similarity of their properties to those of items the user has interacted with. The recommender uses the cleaned metadata dataset, focusing on the “brand” and “title” properties. After pre-processing, the next steps involved choosing an appropriate similarity function and determining a minimum similarity threshold to balance Precision and Recall metrics. This process resulted in the final content-based recommender function. Figure 30 summarises the described steps.

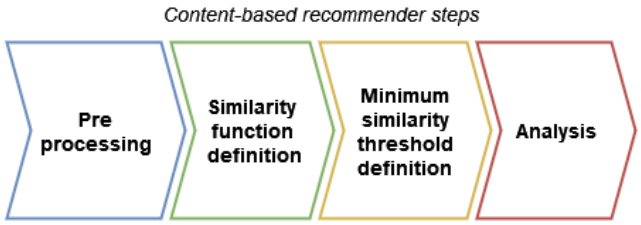

Figure 30 - Content-based recommender development steps

4.4.1 Text labels pre-processing

The process starts by importing the pre-processed metadata dataset and applying a text preprocessing function to the “brand” and “title” columns. This function performs the following steps:

- Removing special characters
- Word tokenization
- Removing English stop words
- Lemmatizing the tokens

This text preprocess function is part of a shared Python class, allowing it to be applied to the user profile items at runtime. The goal of this process is to have all the item’s titles and brands preprocessed, making them available for similarity comparison at runtime. The preprocessed items are stored in a MongoDB collection named “preprocessed_metadata_labels”.

4.4.2 Similarity function definition for recommendations

To determine which similarity function should be applied in the content-based recommender system, it was conducted a comparative analysis of three different functions: Cosine similarity, Euclidean distances and Manhattan distances. The goal was to evaluate whether these functions provide similar or distinct results in the context of item recommendations.

For this test, the database entries were loaded with the pre-processed brands and titles, and it was simulated a scenario where a user adds an item to their cart with the following details: brand - “Motorola” and title - “Motorola Droid RAZR M XT907 4G LTE (...)”.

Using this item as a reference, it was calculated the similarity scores between it and all other items in the database, employing each of the three similarity functions mentioned. Then identified and ranked the top 10 items with the highest similarity scores for the Cosine function and lower distance for Euclidean and Manhattan.

The results are provided in the following subsections along with the conclusion.

Cosine similarity results

The results of the cosine function, where a higher similarity value is better, are presented in Figure 31. This figure lists the top 10 items with the highest cosine similarity scores, along with their corresponding titles.

Additionally, these results are visualized in Figure 32 through a heatmap.

```
Top 10 Recommendations using Cosine similarity:
1. 0.52 - (B009V1D4T8) motorola droid razr xt907 hard design cover case green flower
2. 0.49 - (B005JGSUNY) motorola droid pro xt610 android smartphone verizon
3. 0.46 - (B00E4QJFKW) motorola vehicle navigation gps dock droid razr xt907 non retail packaging
4. 0.44 - (B009U1SXHW) motorola droid razr xt907 hybrid rubber hard case pink black shap stand
5. 0.42 - (B00IP7D75A) black vertical heavy duty rugged cover belt clip side case pouch motorola droid razr xt907
6. 0.41 - (B009NTSQPK) bw hard shield shell cover snap case verizon motorola droid razr xt907 tool black
7. 0.38 - (B00E66D3JO) cellphone trendz tm hybrid high impact bumper case yellow green aztec tribal teal silicone motorola droid razr xt907 4g lte verizon
8. 0.37 - (B009SU3DX4) sacred heart hard case snap rubberized cover motorola xt907 razr
9. 0.36 - (B00AP7DSPM) black rubberized snap protector case motorola droid razr hd xt926 verizon
10. 0.36 - (B00A7XFCYE) oem motorola droid razr hd shell combo w holster amp kickstand xt926 new verizon retail package
```

Figure 31 - Top 10 Cosine similarities and titles results

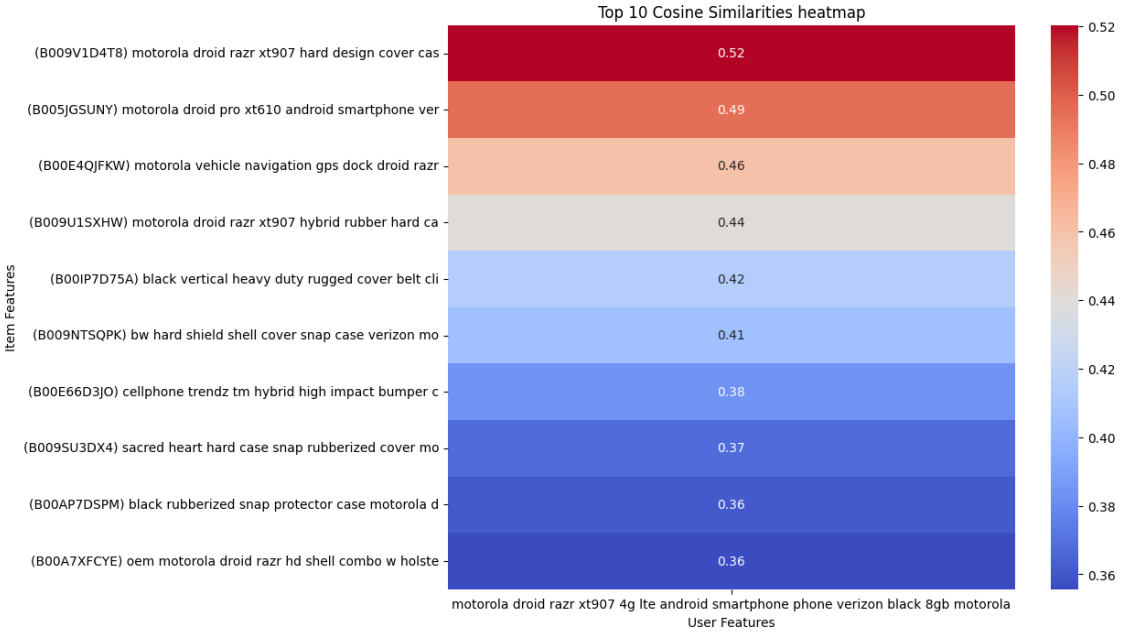

Figure 32 - Top 10 Cosine Similarities heatmap

Euclidean distance results

The results of the Euclidean distance function, where a lower distance value indicates greater similarity, are presented in Figure 33. This figure lists the top 10 items with the lowest Euclidean distance scores, along with their corresponding titles.

These results are further visualized in Figure 34 through a heatmap, which graphically represents the distance values.

```
Top 10 Recommendations using Euclidean distances:
1. 0.98 - (B009V1D4T8) motorola droid razr xt907 hard design cover case green flower
2. 1.01 - (B005JGSUNY) motorola droid pro xt610 android smartphone verizon
3. 1.04 - (B00E4QJFKW) motorola vehicle navigation gps dock droid razr xt907 non retail packaging
4. 1.06 - (B009U1SXHW) motorola droid razr xt907 hybrid rubber hard case pink black shap stand
5. 1.08 - (B00IP7D75A) black vertical heavy duty rugged cover belt clip side case pouch motorola droid razr xt907
6. 1.09 - (B009NTSQPK) bw hard shield shell cover snap case verizon motorola droid razr xt907 tool black
7. 1.11 - (B00E66D3JO) cellphone trendz tm hybrid high impact bumper case yellow green aztec tribal teal silicone motorola droid razr xt907 4g lte verizon
8. 1.12 - (B009SU3DX4) sacred heart hard case snap rubberized cover motorola xt907 razr
9. 1.13 - (B00AP7DSPM) black rubberized snap protector case motorola droid razr hd xt926 verizon
10. 1.14 - (B00A7XFCYE) oem motorola droid razr hd shell combo w holster amp kickstand xt926 new verizon retail package
```

Figure 33 - Top 10 Euclidean distances and titles

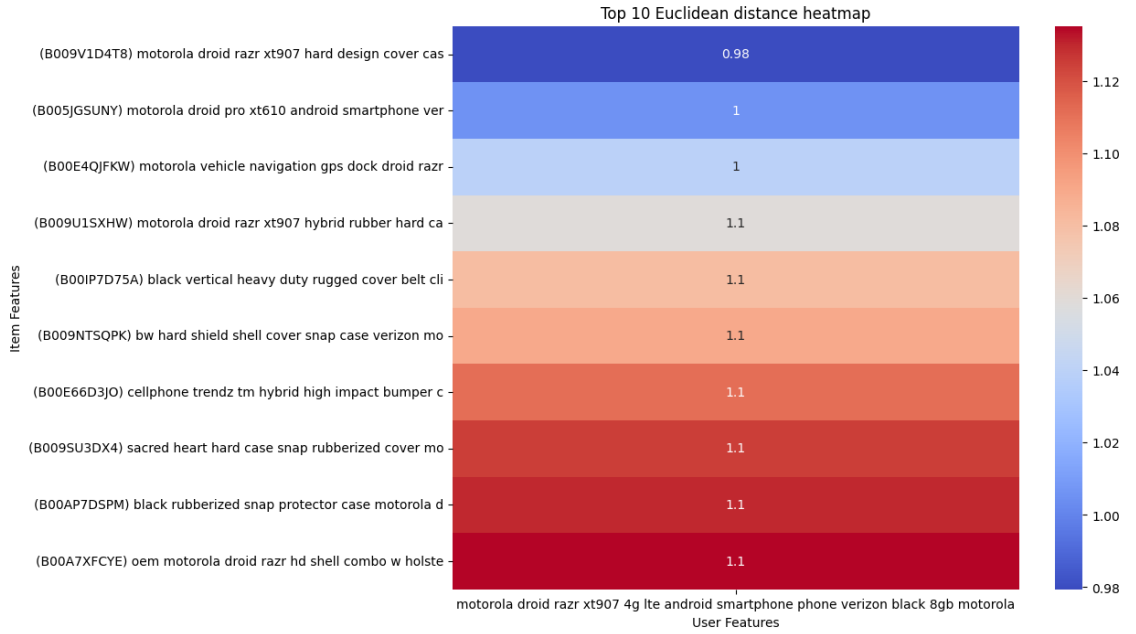

Figure 34 - Top 10 Euclidean distance heatmap

Manhattan distance results

The results of the Manhattan distance function, where a lower distance value indicates greater similarity, are presented in Figure 35. This figure lists the top 10 items with the lowest Manhattan distance scores, along with their corresponding titles.

These results are further visualized in Figure 36 through a heatmap, which graphically represents the distance values

```
Top 10 Recommendations using Manhatan distances:
1. 3.01 - (B005JGSUNY) motorola droid pro xt610 android smartphone verizon
2. 3.79 - (B009NTSQPK) bw hard shield shell cover snap case verizon motorola droid razr xt907 tool black
3. 3.8 - (B007PJEZ32) motorola mb632 android unlocked gsm phone black
4. 3.82 - (B014RJJXUW) verizon wireless lte prepaid smartphone black
5. 3.88 - (B009V1D4T8) motorola droid razr xt907 hard design cover case green flower
6. 3.98 - (B00E4QJFKW) motorola vehicle navigation gps dock droid razr xt907 non retail packaging
7. 4.0 - (B006KAKZTK) mirror screen protector motorola droid razr maxx xt913 verizon
8. 4.03 - (B00Y66EQUO) lg leon lte h340n mobile 4g android smartphone
9. 4.06 - (B009U1SXHW) motorola droid razr xt907 hybrid rubber hard case pink black shap stand
10. 4.11 - (B000CNTEG6) motorola q car charger
```

Figure 35 - Top 10 Manhattan distances and titles

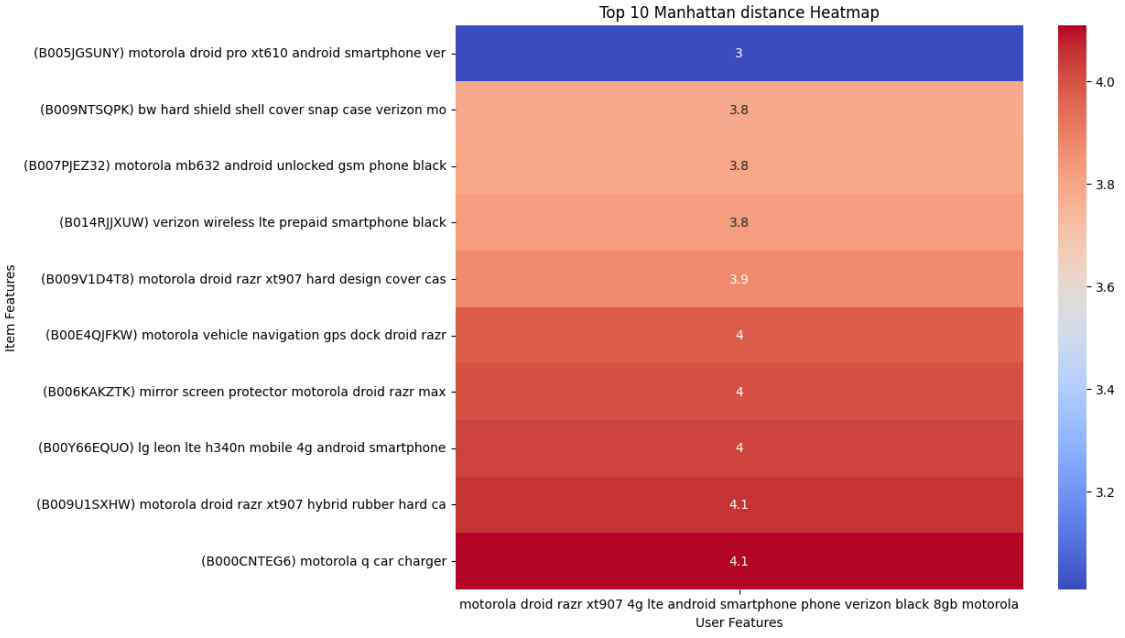

Figure 36 - Top 10 Manhattan distance heatmap

Conclusion

Upon examining the top 10 recommendations generated by each similarity function, it is evident that the Cosine similarity and Euclidean distance functions produce closely matching results, effectively validating each other.

In contrast, the Manhattan distance function not only alters the ranking of some items common to the other two functions but also introduces new items. However, the items recommended by the Manhattan distance function exhibit higher distance values, indicating less precise similarity measurements compared to Euclidean distances.

The Cosine similarity function, due to its intuitiveness, where a higher value directly correlates with higher similarity and reliable validation through comparable results with the Euclidean distance, was chosen to provide recommendations for the content-based recommender system.

4.4.3 Recommender evaluation with mocked user interactions

The content-based recommender was evaluated using precision and recall metrics, as described in the Recommender systems results evaluation section.

The evaluation process involved creating simulated user interactions to assess the system's performance. Specifically, it was simulated that 10 users each purchased one item from each of the top 10 most popular brands identified during the Amazon Review Data EDA step.

To introduce novelty into the recommendations and avoid recommending items that the user has already interacted with, items that are the same, were excluded from the recommendations. A minimum similarity threshold was also defined to ensure that only sufficiently relevant items were considered valid recommendations.

The evaluation process involved generating recommendations based on the items that users interacted with and then calculating the average precision and recall for these recommendations. To account for the dynamic and simulated nature of the user interactions, the evaluation was automatically repeated 10 times, and the results were averaged to obtain robust estimates of precision and recall.

To understand the impact of different similarity thresholds on the system's performance, the recommender was evaluated using various thresholds ranging from 0.1 to 0.9. Allowing to observe how the precision and recall metrics changed as the similarity threshold increased.

The evolution of precision and recall as the similarity threshold increases is represented in Figure 37. Based on these results, it was selected a minimum similarity value of 0.3 to balance achieving relatively high precision and recall.

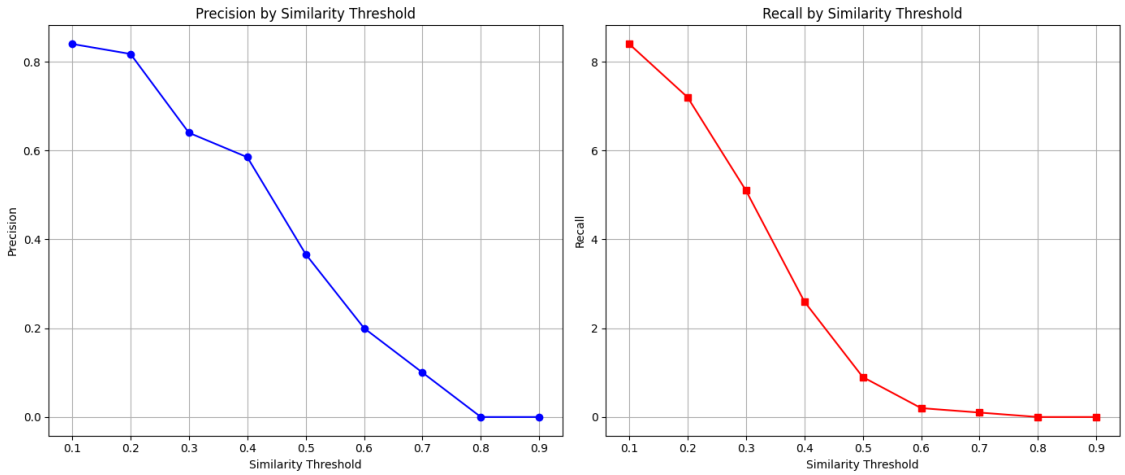

Figure 37 - Precision and Recall by similarity evolution

4.4.4 Content-based final recommender

The content-based recommender, during its initialization, connects to the MongoDB instance loads the stored pre-processed entries and prepares them for use. After loading the entries, it creates a list of all entries' features to initialize the instance TF-IDF vectorizer and its matrix.

The public *recommend_items()* function, Code snippet 13, receives the user profile as a parameter. When this function is called, the user profile, containing the item in the basket title and brand, is pre-processed and vectorized using the instance TF-IDF vectorizer, it then checks the cosine similarity between the user item and the existing items, the ones that are above the minimum similarity threshold defined as 0.3 are returned as a list.

```
def recommend_items(self, userProfile, n_recommendations=10):
    userProfile = pd.DataFrame(userProfile)

    firstItem = userProfile.loc[0]
    preprocessed_user_title = preprocess_text(firstItem["title"])
    preprocessed_user_brand = preprocess_text(firstItem["brand"])

    user_feature = preprocessed_user_brand + " " + preprocessed_user_title
    user_tfidf_vector = self.tfidf_vectorizer.transform([user_feature])

    similarities = cosine_similarity(self.tfidf_matrix, user_tfidf_vector).flatten()
    similar_indices = similarities.argsort()[::-1]

    input_index = (self.item_features.index(user_feature) if user_feature in self.item_features else None)

    recommendations = []
    for i in similar_indices:
        if i != input_index:
            recommendations.append(
                (self.item_ids[i], self.item_titles[i], similarities[i])
            )
        if len(recommendations) == n_recommendations:
            break

    return [item[0] for item in recommendations if item[2] > self.similarity_threshold]
```

Code snippet 13 - Content-based recommender recommend_items() function

4.5 Hybrid Recommender System Integration

A hybrid recommender system combines multiple recommendation techniques to improve the accuracy and relevance of suggestions. The Python-implemented “HybridRecommender” class integrates Content-based filtering, Item-based collaborative filtering, and Association Rules assigning default weights to each: 40% to Content-Based, 40% to Collaborative Filtering, and 20% to Association Rules. By leveraging these weighted techniques, the system delivers more personalized recommendations. The following sections detail the initialization and recommendation processes within this hybrid system.

4.5.1 Initialization

Using Python, a “HybridRecommender” class was implemented, which integrates the final recommender for Content-Based, Collaborative Filtering, and Association Rules methods.

To optimize the initialization process, the ThreadPoolExecutor is utilized for concurrent loading of each recommender, significantly speeding up the overall system setup.

During initialization, the system also assigns the default weights to the results from each recommender:

- Association Rules: 20%
- Collaborative Filtering: 40%
- Content-Based: 40%

This flow is represented in the Figure 38 sequence diagram.

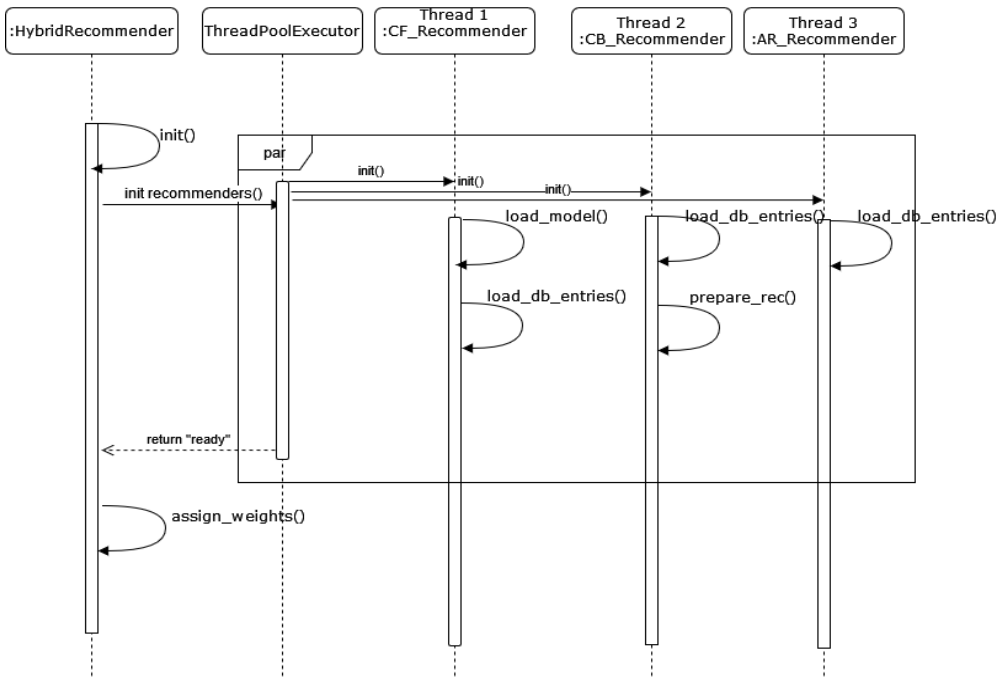

Figure 38 - Hybrid recommender initialization sequence diagram

4.5.2 Getting recommendations

The “HybridRecommender” class also provides a public method “recommend_items()” that accepts the user’s profile as a parameter. Based on this profile, the necessary data is mapped for each recommender, such as using the “userId” for Collaborative Filtering. This method also leverages ThreadPoolExecutor for concurrency, enabling efficient retrieval of recommendations from each recommender.

Once all recommenders have returned their results, the method applies the respective weights to each recommendation index and then combines the results. Finally, the combined recommendations are sorted by score, and the top results are returned as a list. Figure 39 represents the described flow when using a web client.

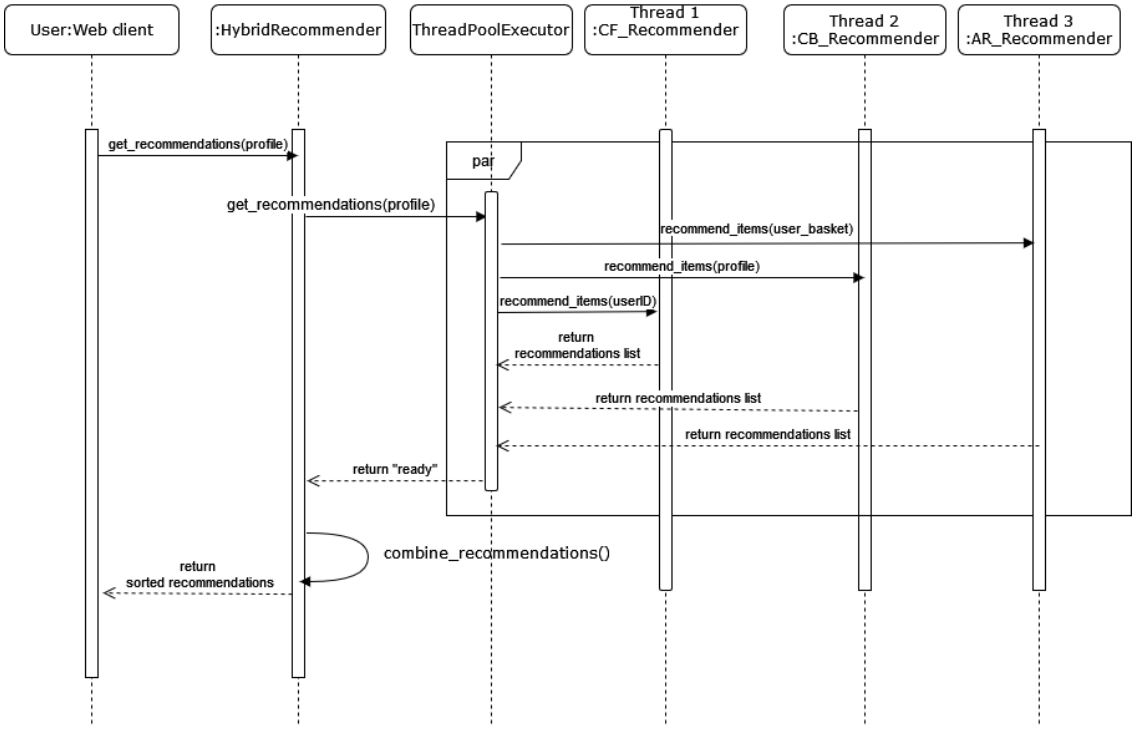

Figure 39 - Get recommendations diagram sequence flow

4.6 Evaluation of recommendations relevance

To assess the relevance of the system's recommendations, a quick client web page was developed using Streamlit Python library (*Streamlit Documentation*, 2024), which instantiates the "HybridRecommender" class, described on the previous section, and displays the top 10 recommendations based on a selected user profile and a product.

Two rounds of interviews were conducted with 10 participants, including friends, family members, and professional colleagues who are familiar with e-commerce platforms. After choosing a product, each participant was asked to rate the recommendations on a scale of 1-5, where:

1 = "Not at all relevant"
2 = "Not very relevant"
3 = "Relevant"
4 = "Very relevant"
5 = "Extremely relevant"

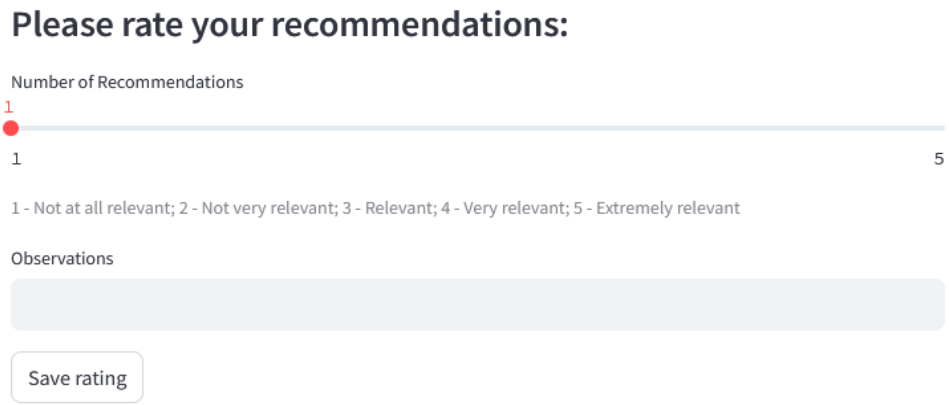

Figure 40 – Recommender client feedback form

Participants were also encouraged to provide additional observations or feedback on the system's recommendations, the feedback form is visible in Figure 40.

In addition to gauging the relevance of the recommendations, it's also a goal to evaluate whether the default weights of each module in the system was appropriate for new users. The defaults weights contain a small, 20%, contribution from the Association Rules module.

Test Scenario

A new profile was created for each participant, simulating a user who is new to the system. Participants were allowed to select one product from a set of 75 products, which were curated from the dataset used to develop the recommendation system.

These 75 products were divided into three categories:

- 25 products that appear as antecedents in association rules with the highest confidence levels.
- 25 products with the highest number of user ratings.
- 25 products chosen randomly from the LG brand.

After selecting a product, participants reviewed the system's top 10 recommendations and assigned a relevance score based on their impressions. Where possible, participants also provided qualitative feedback to explain their ratings and offer insights into the system's performance. The system's profile selection is represented in Figure 41.

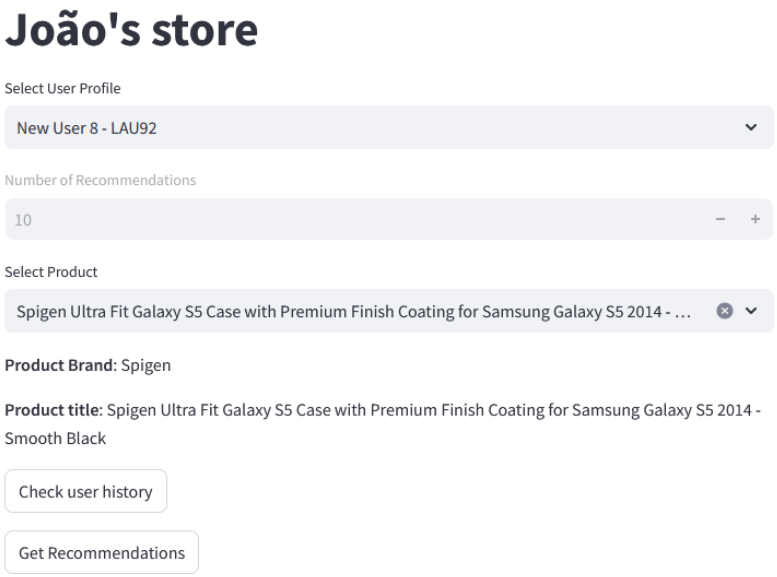

Figure 41 – Recommender client profile selection

4.6.1 First round results

Following interviews with the 10 participants, the initial analysis revealed that users generally found the recommendations to be moderately relevant, with an average rating of 3.2 out of 5.

However, a recurring observation among the participants was that some of the top-ranked recommendations were unrelated to the product they had chosen, Figure 42 displays an example of irrelevant provided recommendations, highlighted in red.

Upon further investigation, the source of this issue was traced to the collaborative filtering module. Since the users were new and lacked a history of reviews or interactions, they were excluded from the model, leading to what is commonly known as the "cold start" problem as described on “Known Recommender systems challenges” subsection.

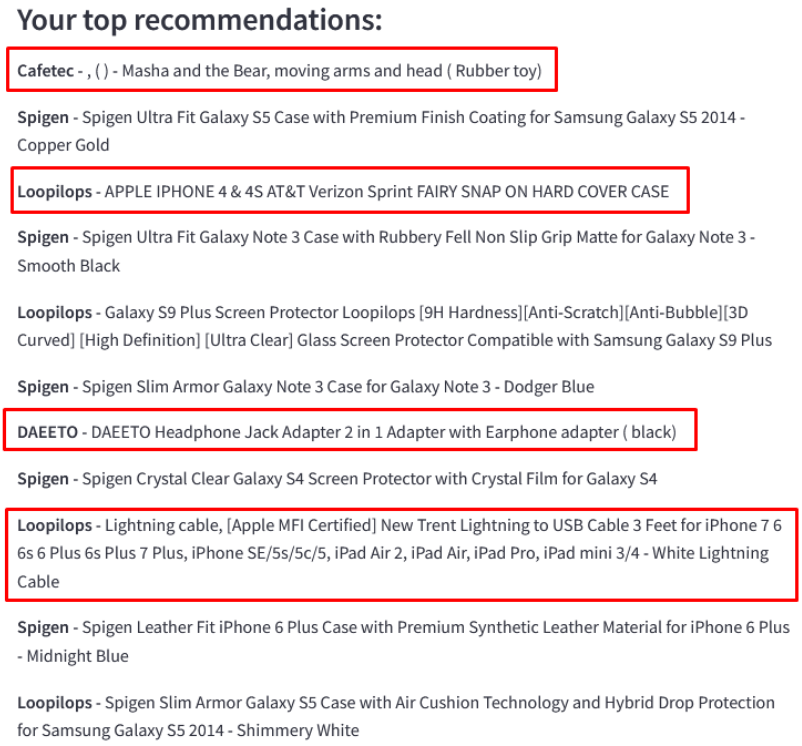

Figure 42 – Irrelevant recommendations highlighted

4.6.2 Adjustments and second round results

Following feedback from the first round of user interviews, adjustments were made to address the identified user cold start problem. Specifically, the system's weighting mechanism was changed so that input from other modules, such as the association rules and content-based filtering, would have a greater influence than the collaborative filtering module when interacting with new users.

This adjustment is only applied to users created after the model. The proposed solution involves adding a timestamp to the database, recording the last time the model was trained. Additionally, each user's account creation date is logged. If the user's creation date is more recent than the model's last training date, indicating that insufficient data may be available on that user, the system dynamically adjusts the module weights. For all other users, the original weighting remains unchanged. Figure 43 illustrates the flow of this dynamic weight adjustment process.

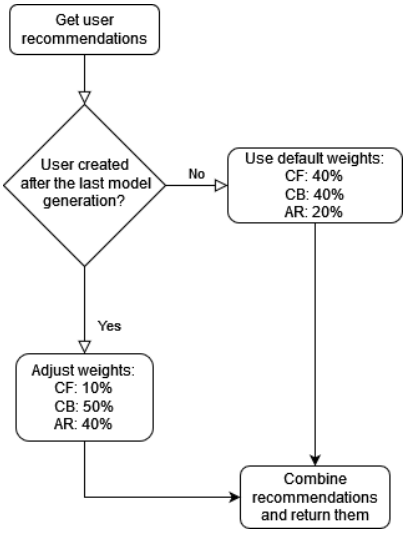

Figure 43 – Dynamic weights based on user creation flowchart

In the second round of interviews, users were presented with new recommendations based on their previously selected product. The adjustments to the weighting mechanism reordered the recommendations, giving higher priority to results from the content-based filtering and association rules components.

Even though Item-based Collaborative filtering has less priority, the recommendations it provides still appear within the top 10 list for certain products. Since not all products are linked to association rule precedents or are present but in a small portion of association rules, the association rules component is unable to completely fill the list with relevant suggestions.

As a result of the reduced influence of irrelevant recommendations from item-based collaborative filtering component, users reported that the overall relevance of the suggestions improved, rating 4.33 out of 5, however still making some observations about some irrelevant suggested products. However, some users still noted irrelevant suggested products. This improvement, driven by placing more weight on the association rules component, demonstrates the potential of this approach to address the user cold start problem.

4.7 Chapter remarks

This chapter introduced a hybrid recommender system, incorporating weights to balance the influence of each component on the system's outcomes. The development process for each component, along with its integration into the final system, was discussed in detail. Results from the user studies indicate the potential of association rules in addressing the user cold-start problem. However, the effectiveness of this approach is dependent on the availability of transaction data to generate enough association rules.

5 Conclusions

This chapter concludes the project by evaluating the achievement of the initial objectives and exploring potential directions for future improvements.

5.1 Fulfilled objectives

At the beginning of this dissertation, the main research question aimed to explore how association rules could be integrated into e-commerce hybrid recommender systems to address their limitations. To answer this, several key steps were set.

The first step involved analysing the current state of e-commerce hybrid recommender systems. Through a systematic review using the PRISMA methodology, various recommender system techniques and their challenges or limitations were identified, as well as different hybrid recommender approaches. This review also introduced how these systems are evaluated, which was crucial for understanding how to assess each module of the proposed approach described in Implementation, thereby completing the second step of examining evaluation strategies.

Finally, to analyse the impact of association rules in recommender systems, a weighted hybrid recommender system prototype combining collaborative filtering, content-based filtering, and association rules modules was developed and documented. Each module was evaluated individually using state-of-the-art metrics.

The relevance of the system's recommendations was further assessed through two rounds of test user interviews. The first round used default weight settings, and after gathering feedback, adjustments were made to improve the user experience for the second round, resulting in the re-ordering of the final recommendations, allowing for the

5.2 Limitations and future work

Given the vast landscape of recommender systems and their widespread use, it was not feasible to cover all relevant concepts and approaches within this dissertation. Additionally, it was not possible to fully explore all sources identified for PRISMA, as this is a meticulous and time-consuming process.

In retrospect, the availability of a dataset that enabled the development of the three modules in the hybrid recommendation system was crucial. The 2018 version of the Amazon Reviews dataset was utilized, as the recently released 2023 version lacked some important properties (McAuley-Lab, 2024). If these missing details become available, it would be valuable to use the 2023 version in a future iteration and compare the results to those presented in this dissertation.

However, there are several limitations and potential areas for improvement in each module:

Association Rules: The effectiveness of this approach heavily depends on the availability of a rich transaction history dataset to generate a larger and more meaningful set of association rules. In this study, it was adopted a creative but less-than-ideal approach due to the absence of sufficient transaction data, which may have impacted the quality of recommendations. Additionally, privacy concerns surrounding transaction records restricted the use of real-world data, as these records are often sensitive and not for public use. During the recommendation relevance interviews stage it was found that some association rules were created but the metadata for the item itself didn't exist.

Item-based Collaborative Filtering: While the collaborative filtering model was trained on ratings from existing, but anonymous, users, it was not feasible to gather feedback from actual users of the system. Although the model produced promising evaluation metrics, the absence of real-time user feedback may limit its practical applicability and the evaluation of its relevance for specific users. It was however possible to easily reproduce the user cold-start problem.

Content-Based Filtering: The content-based filtering module was constrained to using only item titles and brands for comparisons. Incorporating additional product attributes, such as weight, price, or subcategories, could have enriched the recommendation process. This limited feature set may have reduced the system's ability to differentiate between items effectively.

Only a single type of hybrid recommender system was implemented in this approach. As future work, other types of hybrid systems can be explored, allowing for a comparison of results between different approaches.

Furthermore, during the recommendation relevance evaluation phase, some recommended items were neither cell phones nor accessories, but rather misclassified products that slipped through the pre-processing phase without being filtered out. These misclassifications lacked a consistent pattern, making them difficult to handle. To address this, future improvements could involve applying more robust filtering techniques, such as keyword-based filtering on item titles,

to ensure that only relevant products, specifically cell phones and accessories, are included in the recommendation set.

It was also observer during the recommendation relevance interviews that new users consistently received the same recommendations from the item-based collaborative filtering module. Since this model operates as a "black box", it wasn't possible to fully understand or debug the reasoning behind these recommendations, particularly in the context of a user cold-start scenario. This highlights the need for future implementations to incorporate explainability into the recommendation process, not only to enhance user trust and understanding but also to facilitate debugging and improve model transparency.

Another limitation was the restricted ability to gather feedback on the relevance of the recommendations. The interviews were conducted in August and early September which coincided with a common Portuguese holiday period, limiting the number of potential participants and their availability. Although a web page was developed, it was not deployed in time to mitigate this issue.

5.3 Final notes

The initial idea for this dissertation showed up during the first-year lectures of this master's program, where topics such as Association Rules, the Apriori, and FP-Growth algorithms were introduced. This sparked a curiosity about their potential applications in suggesting items to users or influencing store layouts, which motivated a deeper exploration of the subject.

Throughout the process, the idea evolved through numerous iterations, driven by feedback and challenges from this dissertation supervisors. These contributions enriched the educational value of the work and increased its complexity. Personally, it was gratifying to dive into the inner workings of recommender systems and expand my understanding of their components. These techniques are not only academically significant but are also widely used in domains that impact my daily life, such as e-commerce, social media, and digital advertising. Additionally, developing a recommender system exposed me to new technologies that are outside the scope of my current professional experience, making this journey both educational and professionally rewarding.

6 References

Aggarwal, C. C. (2016). *Recommender Systems: The Textbook* (1st ed.). Springer Publishing Company, Incorporated.

Agrawal, R., & Srikant, R. (1994). Fast Algorithms for Mining Association Rules in Large Databases. *Proceedings of the 20th International Conference on Very Large Data Bases*, 487–499.

Alam, I., Khusro, S., & Khan, M. (2021). Personalized content recommendations on smart TV: Challenges, opportunities, and future research directions. *Entertainment Computing*, *38*, N.PAG-N.PAG. 10.1016/j.entcom.2021.100418

Alhijawi, B., & Kilani, Y. (2020). A collaborative filtering recommender system using genetic algorithm. *Information Processing & Management*, *57*(6), 102310. https://doi.org/10.1016/j.ipm.2020.102310

Association for Computing Machinery. (2024, January 31). *ACM Digital Library*. Https://Dl.Acm.Org/.

Balasubramaniam, K. A., & Chidambaram, M. (2018). A Novel Approach in Clustering Supervised Learning (CSL) for Recommender Systems. *International Journal of Simulation -- Systems, Science & Technology*, *19*(4), 25.1-25.4. 10.5013/IJSSST.a.19.04.25

Biblioteca do Conhecimento Online. (2024, January 31). *b-on.pt*. Https://Www.b-on.Pt/.

Bird, S., Klein, E., & Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc."

Bobadilla, J., Ortega, F., Hernando, A., & Bernal, J. (2012). A collaborative filtering approach to mitigate the new user cold start problem. *Knowledge-Based Systems*, *26*, 225–238. https://doi.org/10.1016/j.knosys.2011.07.021

Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems*, *46*, 109–132. https://doi.org/10.1016/j.knosys.2013.03.012

Burke, R. (2002). Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction*, *12*(4), 331–370. https://doi.org/10.1023/A:1021240730564

Du, Y., Fang, M., Yi, J., Xu, C., Cheng, J., & Tao, D. (2019). Enhancing the Robustness of Neural Collaborative Filtering Systems Under Malicious Attacks. *IEEE Transactions on Multimedia, Multimedia, IEEE Transactions on, IEEE Trans. Multimedia*, *21*(3), 555–565. https://doi.org/10.1109/TMM.2018.2887018

Ferrera, C. ;, & Kessedjian, E. (2019). Evolution of E-commerce and Global Marketing. *International Journal of Technology for Business*, *1*(1), 33–38. https://doi.org/10.5281/zenodo.2591544

Ghoshal, A., Mookerjee, V. S., & Sarkar, S. (2021). Recommendations and Cross-selling: Pricing Strategies when Personalizing Firms Cross-sell. *Journal of Management Information Systems*, *38*(2), 430–456. https://doi.org/10.1080/07421222.2021.1912930

Golub, G., & Kahan, W. (1965). Calculating the Singular Values and Pseudo-Inverse of a Matrix. *Journal of the Society for Industrial and Applied Mathematics Series B Numerical Analysis*, *2*(2), 205–224. https://doi.org/10.1137/0702016

Google. (2024, January 31). *Google Scholar*. Https://Scholar.Google.Com/.

Gupta, S., Dixit, V. S., Thampi, El-Alfy, Mitra, & Trajkovic. (2018). Scalable online product recommendation engine based on implicit feature extraction domain. *Journal of Intelligent & Fuzzy Systems*, *34*(3), 1503–1510. https://doi.org/10.3233/JIFS-169445

Hanafi. (2022). Enhance Rating Prediction for E-commerce Recommender System Using Hybridization of SDAE, Attention Mechanism and Probabilistic Matrix Factorization. *International Journal of Intelligent Engineering & Systems*, *15*(5), 427–438. 10.22266/ijies2022.1031.37

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

He, R., & McAuley, J. (2016). Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. *Proceedings of the 25th International Conference on World Wide Web*, 507–517. https://doi.org/10.1145/2872427.2883037

Henriques, R., & Pinto, L. (2023). A novel evaluation framework for recommender systems in big data environments. *Expert Systems with Applications*, *231*, N.PAG-N.PAG. 10.1016/j.eswa.2023.120659

Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design Science in Information Systems Research. *MIS Q.*, *28*(1), 75–105.

Hou, Y., Li, J., He, Z., Yan, A., Chen, X., & McAuley, J. (2024). *Bridging Language and Items for Retrieval and Recommendation*.

https://jupyter.org/. (2024). *Jupyter.org*. Https://Jupyter.Org/.

Hug, N. (2020). Surprise: A Python library for recommender systems. *Journal of Open Source Software*, *5*(52), 2174. https://doi.org/10.21105/joss.02174

Hunter, J. D. (2007). Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Institute of Electrical and Electronics Engineers. (2024, January 31). *IEEE Xplore*. Https://Ieeexplore.Ieee.Org/Search/Advanced.

Jannach, D., & Jugovac, M. (2019). Measuring the Business Value of Recommender Systems. *ACM Transactions on Management Information Systems*, *10*(4), 1–23. https://doi.org/10.1145/3370082

Jannach, D., Zanker, M., Felfernig, A., & Friedrich, G. (2010). *Recommender Systems*. Cambridge University Press. https://doi.org/10.1017/CBO9780511763113

Jha, G. K., Gaur, M., Ranjan, P., & Thakur, H. K. (2023). A trustworthy model of recommender system using hyper-tuned restricted boltzmann machine. *Multimedia Tools & Applications*, *82*(6), 8261–8285. 10.1007/s11042-021-11575-8

Joblib: running Python functions as pipeline jobs. (2024). Https://Joblib.Readthedocs.Io/En/Stable/.

Julian McAuley, U. of C. S. D. (2018). *Amazon review data*. Https://Cseweb.Ucsd.Edu/~jmcauley/Datasets/Amazon_v2/.

Kaggle Inc. (2024). *Kaggle Competitions*. Https://Www.Kaggle.Com/Competitions.

Kalidindi, A., Yavanamandha, P., & Kunuku, A. N. (2019). Discrete Deep Learning Based Collaborative Filtering Approach for Cold Start Problem. *International Journal of Intelligent Engineering & Systems*, *12*(3), 68–75. 10.22266/ijies2019.0630.08

Karakolis, E., Oikonomidis, P. F., & Askounis, D. (2022). Identifying and Addressing Ethical Challenges in Recommender Systems. *2022 13th International Conference on Information, Intelligence, Systems & Applications (IISA)*, 1–6. https://doi.org/10.1109/IISA56318.2022.9904386

Karthik, R. V, & Ganapathy, S. (2021). A fuzzy recommendation system for predicting the customers interests using sentiment analysis and ontology in e-commerce. *Applied Soft Computing*, *108*, N.PAG-N.PAG. 10.1016/j.asoc.2021.107396

Katarya, R., & Verma, O. P. (2017). Effectual recommendations using artificial algae algorithm and fuzzy c-mean. *Swarm & Evolutionary Computation*, *36*, 52–61. 10.1016/j.swevo.2017.04.004

Ko, H., Lee, S., Park, Y., & Choi, A. (2022). A Survey of Recommendation Systems: Recommendation Models, Techniques, and Application Fields. *Electronics*, *11*(1), 141. https://doi.org/10.3390/electronics11010141

Koohi, H., & Kiani, K. (2021). Two new collaborative filtering approaches to solve the sparsity problem. *Cluster Computing*, *24*(2), 753–765. 10.1007/s10586-020-03155-6

Li, J., Lu, K., Huang, Z., & Shen, H. T. (2021). On Both Cold-Start and Long-Tail Recommendation with Social Data. *IEEE Transactions on Knowledge and Data Engineering, Knowledge and Data Engineering, IEEE Transactions on, IEEE Trans. Knowl. Data Eng.*, *33*(1), 194–208. https://doi.org/10.1109/TKDE.2019.2924656

Liu, H., Lin, H., Fan, W., Ren, Y., Xu, B., Zhang, X., Wen, D., Zhao, N., Lin, Y., & Yang, L. (2022). Self-supervised learning for fair recommender systems. *Applied Soft Computing*, *125*, N.PAG-N.PAG. 10.1016/j.asoc.2022.109126

Lourenco, J., & Varde, A. S. (2020). Item-Based Collaborative Filtering and Association Rules for a Baseline Recommender in E-Commerce. *2020 IEEE International Conference on Big Data (Big Data)*, 4636–4645. https://doi.org/10.1109/BigData50022.2020.9377807

Lucas, J. P., Luz, N., Moreno, M. N., Anacleto, R., Almeida Figueiredo, A., & Martins, C. (2013). A hybrid recommendation approach for a tourism system. *Expert Systems with Applications*, *40*(9), 3532–3550. https://doi.org/10.1016/j.eswa.2012.12.061

Maarten Grootendorst. (2021, December 7). *Medium - 9 Distance Measures in Data Science*. Https://Towardsdatascience.Com/9-Distance-Measures-in-Data-Science-918109d069fa.

McAuley, J., & Leskovec, J. (2013). Hidden factors and hidden topics. *Proceedings of the 7th ACM Conference on Recommender Systems*, 165–172. https://doi.org/10.1145/2507157.2507163

McAuley-Lab. (2024, September). *AmazonReviews2023 Github issues*. Https://Github.Com/Hyp1231/AmazonReviews2023/Issues/4. https://github.com/hyp1231/AmazonReviews2023/issues/4

McKinney, W. (2010). *Data Structures for Statistical Computing in Python*. 56–61. https://doi.org/10.25080/Majora-92bf1922-00a

Mingxun, Z., Jiewu, Y., Zhigang, M., & Yanping, W. (2023). Deep Forest-Based E-Commerce Recommendation Attack Detection Model. *Security & Communication Networks*, 1–16. 10.1155/2023/8413247

mongodb.com. (2024). *PyMongo Tutorial: MongoDB And Python | MongoDB*. Https://Www.Mongodb.Com/Resources/Languages/Pymongo-Tutorial.

Murty, C. S. V. V. S. N., Saradhi Varma, G. P., & Satyanarayana, Ch. (2022). Content-Based Collaborative Filtering with Hierarchical Agglomerative Clustering Using User/ Item based Ratings. *Journal of Interconnection Networks*, *22*, 1–18. https://doi.org/10.1142/S0219265921410267

Ni, J., Cai, Y., Tang, G., & Xie, Y. (2021). Collaborative Filtering Recommendation Algorithm Based on TF-IDF and User Characteristics. *Applied Sciences*, *11*, 9554. https://doi.org/10.3390/app11209554

Ni, J., Li, J., & McAuley, J. (2019). Justifying Recommendations using Distantly-Labeled Reviews and Fine-Grained Aspects. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 188–197. https://doi.org/10.18653/v1/D19-1018

Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., Shamseer, L., Tetzlaff, J. M., Akl, E. A., Brennan, S. E., Chou, R., Glanville, J., Grimshaw, J. M., Hróbjartsson, A., Lalu, M. M., Li, T., Loder, E. W., Mayo-Wilson, E., McDonald, S., … Moher, D. (2021). The PRISMA 2020 statement: an updated guideline for reporting systematic reviews. *Systematic Reviews*, *10*(1), 89. https://doi.org/10.1186/s13643-021-01626-4

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel V. and Thirion, B., Grisel, O., Blondel, M., Prettenhofer P. and Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Peffers, K., Tuunanen, T., Rothenberger, M., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*, *24*, 45–77.

Pirasteh, P., Jung, J. J., & Hwang, D. (2014). *Item-Based Collaborative Filtering with Attribute Correlation: A Case Study on Movie Recommendation* (pp. 245–252). https://doi.org/10.1007/978-3-319-05458-2_26

Rahim, A., Durrani, M. Y., Gillani, S., Ali, Z., Hasan, N. U., & Kim, M. (2022). An efficient recommender system algorithm using trust data. *Journal of Supercomputing*, *78*(3), 3184–3204. 10.1007/s11227-021-03991-2

Raschka, S. (2018). MLxtend: Providing machine learning and data science utilities and extensions to Python's scientific computing stack. *The Journal of Open Source Software*, *3*(24). https://doi.org/10.21105/joss.00638

RecSys Community. (2024). *RecSys 2024 - Challenge - RecSys - RecSys*. Https://Recsys.Acm.Org/Recsys24/Challenge/.

Ricci, F., Rokach, L., & Shapira, B. (2011). Introduction to Recommender Systems Handbook. In *Recommender Systems Handbook* (pp. 1–35). Springer US. https://doi.org/10.1007/978-0-387-85820-3_1

Salter, J., & Antonopoulos, N. (2006). CinemaScreen Recommender Agent: Combining Collaborative and Content-Based Filtering. *IEEE Intelligent Systems*, *21*(1), 35–41. https://doi.org/10.1109/MIS.2006.4

Schröder, G., Thiele, M., & Lehner, W. (2011). *Setting Goals and Choosing Metrics for Recommender System Evaluations*. *811*.

Shambour, Q. Y., Turab, N. M., & Adwan, O. Y. (2021). An Effective e-Commerce Recommender System Based on Trust and Semantic Information. *Cybernetics and Information Technologies*, *21*(1), 103–118. https://doi.org/10.2478/cait-2021-0008

Shao, R. (2022). Improvement of Business Analysis Method of E-Commerce System from the Perspective of Intelligent Recommendation System. *Advances in Multimedia*, *2022*, 1–13. https://doi.org/10.1155/2022/7860718

Sharma, R., & Singh, R. (2016). Evolution of recommender systems from ancient times to modern era: A survey. *Indian Journal of Science and Technology*, *9*(20). https://doi.org/10.17485/ijst/2016/v9i20/88005

Simon Funk. (2006, December 11). *Netflix update: Try this at home*. https://sifter.org/simon/journal/20061211.html

Smith, B., & Linden, G. (2017). Two Decades of Recommender Systems at Amazon.com. *IEEE Internet Computing*, *21*(3), 12–18. https://doi.org/10.1109/MIC.2017.72

Streamlit documentation. (2024). Https://Docs.Streamlit.Io/.

Taneja, A., & Arora, A. (2018). Cross domain recommendation using multidimensional tensor factorization. *Expert Systems with Applications*, *92*, 304–316. 10.1016/j.eswa.2017.09.042

Tewari, A. S., & Barman, A. G. (2017). Collaborative Recommendation System Using Dynamic Content based Filtering, Association Rule Mining and Opinion Mining. *International Journal of Intelligent Engineering & Systems*, *10*(5), 57–66. 10.22266/ijies2017.1031.07

Union, E. (2016). Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). *Official Journal of the European Union*, *L 119*(1), 1–88.

Vellaichamy, V., & Kalimuthu, V. (2017). Hybrid Collaborative Movie Recommender System Using Clustering and Bat Optimization. *International Journal of Intelligent Engineering & Systems*, *10*(5), 38–47. 10.22266/ijies2017.1031.05

vom Brocke, J., Hevner, A., & Maedche, A. (2020). *Introduction to Design Science Research* (pp. 1–13). https://doi.org/10.1007/978-3-030-46781-4_1

Wang, C., Deng, Z., Lai, J., & Yu, P. S. (2019). Serendipitous Recommendation in E-Commerce Using Innovator-Based Collaborative Filtering. *IEEE Transactions on Cybernetics, Cybernetics, IEEE Transactions on, IEEE Trans. Cybern.*, *49*(7), 2678–2692. https://doi.org/10.1109/TCYB.2018.2841924

Waskom, M., Botvinnik, O., O'Kane, D., Hobson, P., Lukauskas, S., Gemperline, D. C., Augspurger, T., Halchenko, Y., Cole, J. B., Warmenhoven, J., de Ruiter, J., Pye, C., Hoyer, S., Vanderplas, J., Villalba, S., Kunter, G., Quintero, E., Bachant, P., Martin, M., … Qalieh, A. (2017). *mwaskom/seaborn: v0.8.1 (September 2017)*. Zenodo. https://doi.org/10.5281/zenodo.883859

Widiyaningtyas, T., Ardiansyah, M. I., & Adji, T. B. (2022). Recommendation Algorithm Using SVD and Weight Point Rank (SVD-WPR). *Big Data and Cognitive Computing*, *6*(4), 121. https://doi.org/10.3390/bdcc6040121

Yang, Q. (2019). A robust recommended system based on attack detection. *Concurrency & Computation: Practice & Experience*, *31*(12), N.PAG-N.PAG. 10.1002/cpe.4660