ARISTOTLE UNIVERSITY OF THESSALONIKI

A Comprehensive Benchmark of Product Recommendation Algorithms

by

Christos Maskaleris - 3174

A thesis submitted in partial fulfillment for the
Undergraduate degree

in the
Faculty of Sciences
School of Informatics

Supervising Professor: Dr. Athena Vakali
Co-Supervisor: Dr. Petros Drineas

June 2023

Declaration of Authorship

I, Christos Maskaleris, declare that this thesis titled, A Comprehensive Benchmark of Product Recommendation Algorithms and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

ARISTOTLE UNIVERSITY OF THESSALONIKI

Abstract

Faculty of Sciences
School of Informatics

Undergraduate Degree

by Christos Maskaleris - 3174

In the world of e-commerce, product recommendation algorithms play a pivotal role in enhancing user experience and boosting sales. The thesis titled ”A Comprehensive Benchmark of Product Recommendation Algorithms” dives deeply into a broad examination of these algorithms with a focus on understanding their strengths, weaknesses, overall performance and rank them accordingly.

To conduct this analysis, the widely recognized Amazon Dataset for products and reviews was utilized. Leveraging Python for data manipulation and algorithm implementation, an wide variety of recommendation algorithms, from traditional popularity based recommender to various collaborative filtering and content-based methods, were examined..

The study aims to provide insights into how different factors such as data sparsity, scalability, and cold-start problems impact the performance of these recommendation algorithms. It also explores how these systems handle real-world complications such as how they can be integrated into real production systems.

By providing a comprehensive comparison of recommendation algorithms, this thesis aims to serve as a valuable guide for researchers and practitioners in the field of e-commerce. It strives to assist in the selection of the most suitable algorithm for specific application scenarios, thereby maximizing the potential of personalized product recommendations to enhance customer satisfaction and drive business growth.

Acknowledgements

Undertaking this extensive benchmarking of product recommendation algorithms and formulating this thesis has been a rigorous yet rewarding experience. Throughout this journey, I've had the privilege of being accompanied by an incredibly supportive network.

Foremost, I'd like to extend my deepest gratitude to my supervisor, Professor Athena Vakali, Director of Data and Web Science Lab of Aristotle University Thessaloniki. Her guidance, support, and trust in me have been pivotal in this endeavor. The opportunities she extended to me throughout our collaboration have contributed to my personal and professional growth.

I'm also immensely thankful to Professor Petros Drineas, Associate Head of Department of Computer Science in Purdue University, Indianapolis, who has been an additional source of support and inspiration throughout this project. His expert guidance has immensely enriched my work and understanding of the subject.

Additionally, my heartfelt thanks go out to my friends and colleagues at the School of Computer Science: Thomas Alavanos, Dimosthenis Tsormpatzoudis, and George Kynigopoulos. All three, being deeply involved in the field of AI, have offered their insights and expertise, which has greatly contributed to the quality of this research. Their camaraderie and intellectual companionship have made this journey enjoyable and enlightening.

Lastly, I'd like to express my sincerest gratitude to my family, who have consistently prioritized my education and ensured that I never missed any opportunity, even during the most challenging times. Their unwavering faith and support have been my pillars of strength throughout this journey.

Contents

List of Figures

List of Tables

Abbreviations

Acronym	What (it) Stands For
CBF	**Content Based Filtering**
CF	**Collaborative Filtering**
CSV	**Comma Separated Value**(s)
e-commerce	**electronic commerce**
e.g.	**exempli gratia** (“for the sake of an example”)
etc.	**et cetera** (“and other things”)
JSON	**JavaScript Object Notation**
NDCG	**Normalized Discounted Cumulative Gain**
NLP	**Natural Language Processing**
PBF	**Popularity Based Filtering**
PCA	**Principal Component Analysis**
SDK	**Software Development Kit**
SVD	**Singular Value Decomposition**
TF-IDF	**Term Frequency - Inverse Document Frequency**
UI	**User Interface**
URL	**Uniform Resource Locator**
VSM	**Vector Space Model**
WWW	**World Wide Web**
XML	**eXtensible Markup Language**

Dedicated to my brother, in the hope that this thesis will serve as a source of motivation and inspiration for his ongoing university studies...

Chapter 1

Introduction

The Digital Age, characterized by the rapid development of technology, has revolutionized countless aspects of everyday life. One area that has seen profound change is the world of retail. With the advent of e-commerce, the way we shop has evolved dramatically, offering consumers the ability to browse and purchase goods from the comfort of their own homes. However, this convenience comes with the challenge of navigating an ocean of options to find the products most relevant to us.

To address this, businesses have turned to product recommendation algorithms. These tools are designed to suggest products based on a dozen of factors such as user behavior, past purchases, and the preferences of similar users [4]. They create personalized shopping experiences that are as unique as the consumers themselves, enabling them to discover what they want, even when they don't know they want it.

"If I had asked people what they wanted, they would have said faster horses." – Henry Ford

This saying, attributed to the great innovator Henry Ford, is more than just a commentary on his vision that led to the birth of the automobile industry. It has grown to become a beacon of innovation, highlighting the idea that true breakthroughs often go beyond addressing known needs and instead, conceive and create solutions that people didn't know they needed, but quickly become essential once introduced.

Now, let's take a moment and translate this into the world of online retail, specifically into the sphere of product recommendation systems. You might not think a century-old quote about horses and cars could be relevant here, but there's a profound connection. In the same way Ford's automobiles revolutionized transportation, recommendation algorithms are reshaping how we discover and buy products online.

These advanced algorithms don't just serve up more of what we know we like, similar to offering faster horses. Instead, they unveil products that we might never have thought of exploring,

but could end up finding useful or even essential - the metaphorical 'automobiles' of our shopping experience. In this sense, product recommendation systems are doing exactly what Ford was hinting at. They're not just giving us faster horses, they're showing us the automobiles we didn't even know we wanted.

However, like any technological innovation, product recommendation algorithms are not without their challenges. As the volume and complexity of available data grow, so does the need for more advanced and efficient algorithms. Furthermore, there is a pressing need for comprehensive benchmarks to evaluate and compare the performance of these algorithms, in order to guide future developments in the field [72].

This thesis is motivated by the need for such a benchmark. It seeks to provide a comprehensive evaluation of various product recommendation algorithms, aiming to shed light on their performance, strengths, and weaknesses. By doing so, it hopes to provide valuable insights to both businesses looking to optimize their platforms and researchers seeking to advance the field.

1.1 Background and Significance of the Study

In our current digital era, marked by a vast number of online transactions, businesses globally are harnessing cutting-edge technologies to customize and elevate the customer experience. A significant force driving this evolution is product recommendation algorithms. The influence of these algorithms on business metrics is profound. They considerably boost user engagement, increase the average order value, and ultimately, drive sales growth. According to a report by McKinsey [58], 35% of Amazon's revenue and 75% of what Netflix viewers watch come from their recommendation engines, underscoring the impact these systems can have on business performance. Furthermore, they foster a deeper connection between businesses and their customers, enhancing brand loyalty and promoting customer retention. Given their critical role in contemporary e-commerce, understanding their workings, effectiveness, and potential areas for improvement becomes essential. This study is designed to provide an in-depth examination of these facets.

1.2 Evolution of E-commerce and the Role of Product Recommendations

The rise of online retail traces back to the 1990s, marking the onset of e-commerce. Winning online marketplaces like Amazon and eBay introduced a new paradigm of shopping that offered unmatchable convenience in shopping experience and variety [82] of products. Their success

laid the foundation for other businesses, leading to a surge in the e-commerce sector, now giving drive to a diverse array of product categories and services.

Alongside the expansion of online retail, the concept of product recommendations began to gain traction. Initial recommendation systems were relatively primitive, primarily employing user-item matrices and straightforward algorithms. However, with technological progression and the exponential growth in accessible data, these systems evolved into advanced recommendation engines. Today, powered by state-of-the-art machine learning algorithms and artificial intelligence, these engines can process vast volumes of data, producing highly personalized product suggestions. As a result, they enhance customer satisfaction and significantly boost business performance, further highlighting the necessity and significance of this research.

1.3 Categories of Recommendation Algorithms

Product recommendation algorithms can be categorized broadly into collaborative filtering, content-based filtering, and hybrid systems[46].

Collaborative filtering forecasts a user's interests by gathering preferences from numerous users, assuming that if two users align on some interests, they will likely align on others as well.

Content-based filtering, on the other hand, recommends items by comparing the content of the product with a user profile. Each product's content is represented by a set of descriptors, such as the words in a document.

Hybrid systems combine the strengths of both methods mentioned above and often display better prediction performance than either of the approaches on their own.

1.4 Progress in Recommendation Algorithms

Product recommendation algorithms have seen significant advancements, particularly over the last decade. With the integration of machine learning and artificial intelligence, these algorithms have become increasingly efficient at analyzing large volumes of data, identifying patterns, and making insightful product recommendations.

Deep learning, a subset of machine learning, has been particularly influential in this respect. These models employ neural networks with several layers to learn representations of data with multiple levels of abstraction, making them potent tools for recommendation systems.

1.5 Challenges in Product Recommendation Algorithms

While product recommendation algorithms have brought significant advances in the world of online retail, they are not without their challenges. As the field continues to evolve, new obstacles arise that require innovative solutions. Some of the most significant challenges faced by these algorithms include:

- **Scalability**: As the user base and product catalogs of online retail platforms grow , scalability becomes a significant challenge. Algorithms need to handle vast amounts of data while maintaining their efficiency and accuracy[12].

- **Cold Start Problem**: This problem occurs when a new user or a new product enters the system. With new users, there is a lack of historical data to base recommendations on. For new products, the lack of user interaction data makes it difficult for the algorithm to identify potential interested customers[76].

- **Sparsity**: User-item interactions in online retail are often sparse because a user typically interacts with a tiny fraction of the total items. This sparsity can make it challenging to identify meaningful patterns for recommendation[1].

- **Diversity and Serendipity**: While recommendation algorithms aim to provide relevant suggestions, they often end up recommending popular items or items very similar to those the user has already interacted with. This lack of diversity and serendipity can lead to a less satisfying user experience[18].

- **Privacy and Security**: Personalized recommendation systems often rely on collecting and analyzing vast amounts of personal data, which can raise privacy concerns. Ensuring the user's data is secure and their privacy is respected is a significant challenge in building effective recommendation systems[67].

- **Dynamic Environment**: The online retail environment is highly dynamic, with user preferences, item availability, and trends changing rapidly. Algorithms need to be able to adapt quickly to these changes to provide relevant recommendations[26].

Overcoming these challenges is an area of ongoing research and development in the field of product recommendation algorithms.

1.6 Aims of the Study

The main objective of this study is to provide a comprehensive benchmark of various product recommendation algorithms. It seeks to evaluate their performance, highlight best practices, and identify potential areas for improvement.

Furthermore, the study aims to address existing challenges in recommendation algorithms, such as dealing with sparse data, scalability , the cold start problem, while also investigating how these systems maintain a balance between personalization and user privacy.

Lastly, the study will propose potential strategies and directions for future research in the realm of recommendation algorithms , with the goal of facilitating the development of more effective, efficient, and user-friendly systems for e-commerce platforms.

1.7 Research Questions

This research aims to answer several key questions regarding product recommendation algorithms:

- How do different types of recommendation algorithms perform in terms of accuracy, precision, recall, and other metrics? [80]
- What are the most effective strategies for dealing with common challenges in recommendation algorithms, such as handling sparse data, ensuring scalability, and addressing the cold start problem? [6]
- How do recommendation algorithms balance the need for personalization but also combining popularity and similarity metric? [84]
- What future trends and potential research directions can be anticipated in the field of product recommendation algorithms?

1.8 Structure of the Thesis

This thesis is structured as follows:

- **Chapter 2**: Literature Review - A comprehensive review of existing literature on product recommendation algorithms, their types, advancements, and challenges.

- **Chapter 3**: Benchmark Studies - Discusses the importance of benchmarking and presents general and product-specific recommendation benchmarks.
- **Chapter 4**: Methodology & Implementation - Outlines the data collection, preparation, and processing methods, describes the algorithms used, and discusses potential limitations.
- **Chapter 5**: Results & Discussion - Presents the dataset overview and the results obtained from the different types of recommenders. It includes a comparison of results and a discussion on the findings.
- **Chapter 6**: Experimentation - Discusses experiments conducted with dataset sparsity and presents the findings.
- **Chapter 7**: Conclusions & Future Work - Summarizes the findings, acknowledges the limitations of the study, and proposes potential directions for future research in the field of product recommendation algorithms.

Chapter 2

Literature Review

The purpose of the Literature Review section in this thesis is to provide an extensive overview of the existing body of knowledge related to recommendation systems. This serves as the foundation for the study, offering a thorough understanding of the topic and showcasing the breadth and depth of research already conducted in this field.

This section will explore and summarize key research papers, studies, and sources that have contributed to our understanding of recommendation systems. We will delve into the theoretical research, methodologies, and findings of these works, critically evaluating their contributions and identifying gaps where further research is needed. An important aspect of this review is to understand the evolution of recommendation systems over time. We will trace the advancements in the field, from the earliest systems to the cutting-edge technologies of today, highlighting the innovations and improvements that have shaped their current state.

It's important to note that the literature review is not merely a summary of previous works but a critical evaluation that sets the stage for the research questions and objectives of this thesis. As suggested by Jesson, J. Matheson and L. Lacey, . in [25], a well-executed literature review can illuminate the path for new research directions and methodologies.

2.1 Recommendation Process Phases

In the realm of product recommendation systems, the process can be splited as a sequence of stage. It is critical to understand this sequence to effectively harness and improve the capabilities of such systems. The recommendation process usually includes three important stages: Information Collection, Learning, and Prediction/Recommendation.

The Information Collection stage involves gathering important user data to construct a comprehensive user profile or model, which serves as the basis for upcoming predictions. The

Learning phase utilizes this collected data, applying suitable algorithms to identify patterns and user preferences. Lastly, the Prediction/Recommendation phase uses these insights to suggest items the user may find appealing.This process is not linear, but rather a cyclical one, constantly evolving and adapting to the user's changing preferences and interactions with the system. This dynamic nature of the recommendation process ensures that the system continually provides relevant and personalized suggestions to the user.

It is essential to note that each stage presents its unique challenges and opportunities, and an effective recommendation system must address these to optimize its performance. The following sections delve deeper into each phase, exploring their implications and highlighting their significance in the overall recommendation process. As stated by Burke, R. in his study [16] understanding the interplay of these stages is crucial to the design and enhancement of recommendation systems. Furthermore, Tuzhilin, A. in their work [1] emphasize the continual evolution and refinement of these stages as key to the future development of more precise and user-centric recommendation systems.

2.1.1 Information Gathering Phase

The first step in the recommendation process is all about collecting user data. This data helps build a detailed user profile or model, which includes user traits, behaviors, and the resources they use. The goal is to learn as much as we can about the user. A well-built user profile is key for a recommendation tool to work well and give the right recommendations from the start.

Recommendation tools use different kinds of input. The most common are explicit and implicit feedback. Explicit feedback is when a user directly shows their interest in a specific item. Implicit feedback is more subtle and is gathered by watching user behavior. There's also a mix of the two, known as hybrid feedback.[42].

Let's take Netflix as an example. Here, a user profile has a lot of personal data about a specific user. This can include their favorite genres, what they've watched, how they interact with the system, and even things like how long they usually watch and the devices they use to stream. This profile acts as a basic user model and is key in gathering the data needed to build a more detailed user model.

A recommendation system works well when it can accurately show a user's current interests. That's why having precise models is so important for giving relevant and accurate recommendations.[42].

1. **Explicit Feedback**

   Explicit feedback is a way for the system to understand user preferences directly. This typically comes through the system interface, where users are invited to rate items. These

ratings contribute significantly to constructing and enhancing the user's model. It's worth noting that the credibility of the recommendation is proportional to the number of ratings provided by the user.

However, one drawback of this approach is that it requires user effort. Not all users are willing or able to provide this information. Despite the extra effort required from users, explicit feedback is often seen as more reliable since it doesn't rely on interpreting user actions. Moreover, the transparency it provides into the recommendation process tends to increase user confidence in the recommendations, thus potentially enhancing the perceived quality of the recommendations. [42]

2. **Implicit Feedback**

   Implicit feedback is another method through which the system learns about the user's preferences. In this approach, the system tracks user actions like purchase history, browsing patterns, time spent on certain pages, the links they click, and even the content of their emails. The system then infers user preferences based on these behaviors. The beauty of implicit feedback is that it doesn't demand any direct effort from the user.

   However, this method isn't as accurate as explicit feedback. It's more like reading between the lines, which leaves room for interpretation errors. Yet, some argue that implicit feedback might actually offer a more honest reflection of user preferences. This is because it's not subject to the same biases as explicit feedback, such as users giving socially desirable responses [15] [32].

3. **Hybrid Feedback**

   A hybrid approach blends both explicit and implicit feedback, drawing on their respective strengths and mitigating their drawbacks. This fusion results in a high-performing system that delivers more reliable recommendations. In essence, hybrid feedback uses implicit data to validate explicit ratings, or it gives the user the option to offer explicit feedback only when they're eager to express a specific interest. The synergy of these two feedback types in a hybrid system optimizes the recommendation process.

2.1.2 Learning Phase

After the data collection stage, we enter the learning phase. In this phase, complex learning algorithms are employed, making the most of the user attributes derived from the previously collected feedback. This phase converts raw user data into valuable insights. Although a full explanation of these learning algorithms and their individual methods is too detailed for this section, we will explore them more deeply in subsequent chapters of this thesis.

2.1.3 Prediction Phase

After the learning phase, we move to the prediction or recommendation stage. At this point, the system starts to suggest or forecast items that might be of interest to the user. These recommendations are either directly taken from the dataset collected during the data collection phase, or are based on the system's observation of user activities. The datasets used might be memory-based, model-based, or a mix of both. While the details of these processes are extensive and intricate, we will examine them more in future sections of this thesis. This phase represents the culmination of the recommendation phases, as shown in Figure 1.

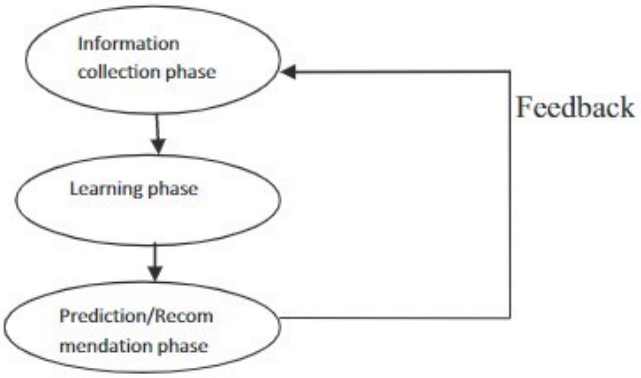

FIGURE 2.1: Recommendation Phases

2.2 Recommendation Filtering Techniques

The effectiveness of any recommendation system depends on the use of robust and accurate recommendation techniques. These are crucial for delivering valuable and personalized recommendations to each user. As a result, understanding the unique features and capacities of various recommendation techniques is vital. This knowledge enables us to shape the system to maximize user satisfaction and system performance. While these techniques can be complex and diverse, understanding them becomes easier when they're broken down into their basic elements, as shown in Figure 2.2.

In this section, we'll examine the specifics of recommendation filtering techniques, exploring their strengths, weaknesses, and uses.

2.2.1 Content-Based Filtering

Content-Based Filtering (CBF), also known as cognitive filtering, works mainly on the semantics of the items, engaging in information retrieval[7]. This method is particularly effective

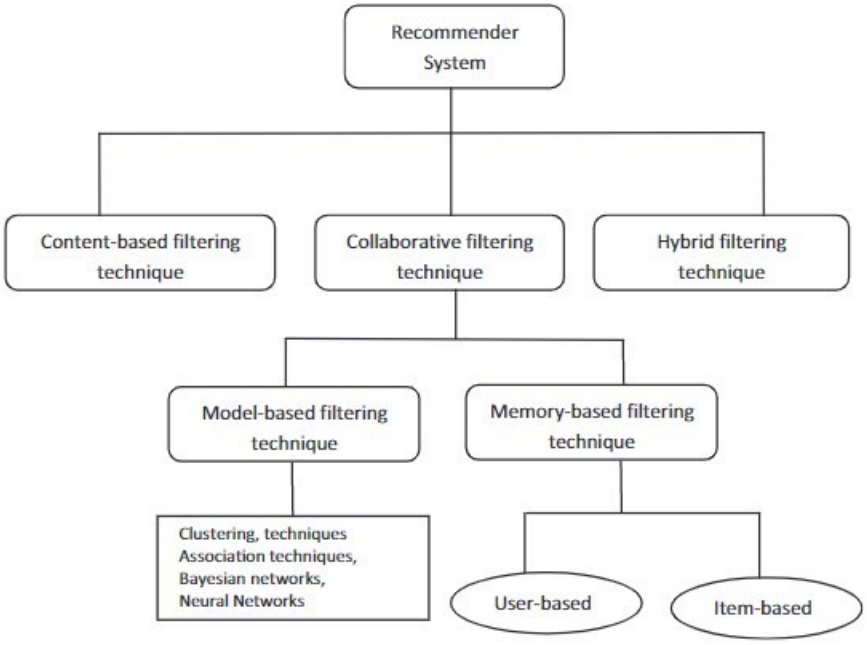

Figure 2.2: Recommendation Techniques

when recommending documents like web pages, publications, and news items. The underlying principle is straightforward yet robust: recommendations are created based on a user's profile and the attributes of the items, which are drawn from the content of items the user has previously evaluated[16], [12].

The user profile is established when the user initially interacts with the system, collecting the user's interests and then recommending items after a thorough analysis of the item features and user characteristics. Therefore, the items recommended by the CBF technique are similar to those that the user has expressed a liking for in the past, aligning with the user's attributes.

To find these similarities, CBF uses different models. Some use the **Vector Space Model** [61], such as the **Term Frequency-Inverse Document Frequency (TF/IDF)** [19] , while others use probabilistic models like **Naïve Bayes Classifier, Decision Trees**, and **Neural Networks** [78]. These models aim to identify the relationship between different documents within a collection, using either statistical analysis or machine learning techniques to understand the underlying model.

One of the key features of CBF is its independent operation; it doesn't depend on the profiles of other users. This means that other users' preferences or opinions have no bearing on the recommendations offered to a specific user. Moreover, CBF systems are quick to adapt their recommendations if a user's preferences evolve over time.

However, CBF isn't without its challenges. A significant hurdle is the requirement for a deep understanding and detailed description of the item's features in the user's profile. This need

for comprehensive knowledge about item characteristics can sometimes hinder the effective implementation of this method. Thus, the clarity and accuracy of item descriptions and user profiles play a crucial role in the success of CBF.

With a strong understanding of both the user's profile and the item's attributes, CBF's main function is to accurately predict whether the active user would appreciate the recommended item or not [78]. This underscores the importance of the comparison between the active user's preferences and the item's attributes, leading to recommendations tailored to the user's interests.

2.2.1.1 Advantages and Disadvantages of Content-Based Filtering

Pros:

- Content-Based Filtering (CBF) techniques offer several advantages that help overcome some of the issues faced by Collaborative Filtering (CF). Firstly, CBF can recommend new items even without previous user ratings, ensuring the quality of recommendations even when user preference data is limited. This flexibility also extends to adapting to changing user preferences, with the ability to quickly adjust recommendations as the user's interests change.
- Another advantage of CBF is its ability to handle situations where different users do not share the same items. Instead, it focuses on recommending items that are similar based on their inherent features. This feature-focused approach allows for more personalized recommendations, even when users' tastes vary greatly.
- In terms of user privacy, CBF stands out as it can provide recommendations without needing to share user profiles, thereby maintaining the privacy of user data. Also, CBF systems can provide clear explanations to users on how recommendations are generated, fostering trust and understanding in the recommendation process.

Cons:

- Despite its advantages, CBF is not without challenges. One of the most significant drawbacks is its reliance on item metadata. The method requires a rich description of items and a well-organized user profile before it can make useful recommendations. This reliance on detailed descriptive data is often referred to as limited content analysis. Therefore, the effectiveness of CBF is tied to the availability and quality of such data.

- Another concern with CBF is the issue of overspecialization. Because it mainly recommends items similar to those already in a user's profile, users can find themselves in a sort of "filter bubble," receiving recommendations that are perhaps too narrowly aligned with their past preferences. This could potentially limit the diversity of the recommendations, depriving users of the opportunity to discover new items outside their usual interests.
- Given these pros and cons, it's clear that the success of CBF depends on the balanced interaction of user profiles, item metadata, and the system's ability to diversify its recommendations while still respecting user preferences. More research and development in this area could help to further improve the performance and versatility of CBF techniques.

2.2.2 Collaborative Filtering

Collaborative Filtering (CF) is vital in product recommendation systems due to its domain-neutral nature, making it ideal for recommending products that can't be easily characterized by metadata. The underlying principle of CF involves creating a preference database, usually represented as a user-item matrix. Here, the preferences of each user for various products are captured[39].

Once this matrix is formed, CF identifies users with similar interests and preferences by comparing similarities between their profiles. These users with similar preferences form a 'neighborhood'. The system then curates product recommendations for a user based on the items they haven't rated yet, but have been positively rated by other users in their neighborhood. The recommendations generated by CF can be either predictions or recommendations. A prediction, denoted as Rij, gives a numerical estimate of the expected rating for a product j by user i. In contrast, a recommendation offers a list of the top N products that the system predicts the user will find most appealing.

Collaborative filtering techniques can be primarily divided into two types: memory-based and model-based[12]. Each type has unique advantages and considerations, which will be examined in the following sections. This will provide a comprehensive understanding of the different aspects of collaborative filtering and their specific applications in product recommendation systems.

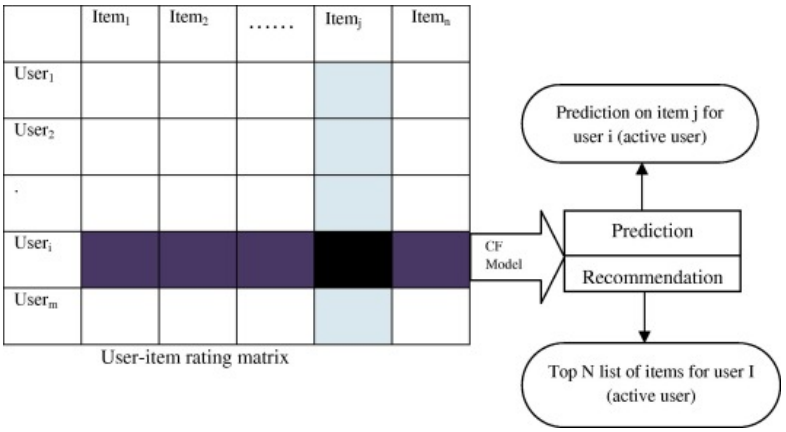

FIGURE 2.3: Collaborative Filtering Process.

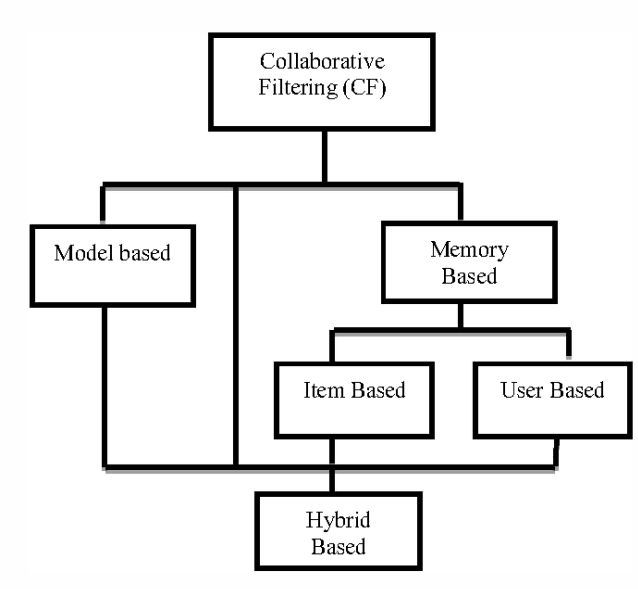

FIGURE 2.4: Collaborative Filtering Techniques

2.2.2.1 Model-Based

Model-based techniques are crucial in improving the performance of collaborative filtering for product recommendation systems. They use past ratings to create a predictive model, thus significantly enhancing the efficiency and accuracy of recommendations. While collaborative filtering typically involves analyzing a large user-item matrix, model-based methods simplify this process by using machine learning or data mining techniques to learn and update the model. [65]

Compared to memory-based approaches that are analyzed later, model-based techniques use the pre-computed model for recommendations, increasing speed and scalability [60]. They are especially effective in addressing the sparsity problem in recommendation systems, often caused by a large number of items but relatively few user interactions. Model-based methods, such as Matrix Factorization [14], have been widely used in the field due to their effectiveness in prediction and their ability to handle high-dimensionality and sparsity of the user-item matrix.

Other notable model-based techniques include Dimensionality Reduction techniques (e.g., SVD) [68], Matrix Completion Techniques [13], Latent Semantic methods [40], and Regression and Clustering. These methods analyze the user-item matrix to discover relationships between items, comparing the list of top-N recommendations based on these relations.

Model-based collaborative filtering methods have evolved in recent years, not only suggesting products a user may like but also indicating when the user might be most receptive to consuming a product. Various learning algorithms play a crucial role in enhancing the capabilities of model-based recommender systems. Some popular techniques and algorithms used in model-based CF include:

- **Matrix Factorization**: Techniques such as Singular Value Decomposition (SVD) [68], Probabilistic Matrix Factorization (PMF) [70], Non-negative Matrix Factorization (NMF) [56], and Alternating Least Squares (ALS) [38], are widely used in model-based Collaborative Filtering. These methods break down the user-item interaction matrix into smaller-dimensional matrices, thereby revealing hidden attributes and connections within the data. ALS, in particular, is recognized for its scalability and efficiency, making it a great choice for large-scale recommendation problems.

  Clustering: Clustering techniques, such as K-Means [30] and Self-Organizing Maps (SOM) [3], can be employed as a pre-processing step to segment users into distinct groups based on shared characteristics or behaviors. Once these groups are formed, other model-based techniques can be applied to each group individually. This allows the system to make tailored recommendations for each user group, often based on the

average preferences of users within the same cluster. This approach can improve the relevancy of recommendations, leading to enhanced user satisfaction and engagement.

• **Artificial Neural Networks (ANN)**: ANNs are complex structures composed of numerous interconnected nodes, or neurons, systematically arranged in different layers. The influence one neuron exerts on another is represented through weighted connections between them. ANNs offer several advantages in tackling complex problem scenarios. For instance, an ANN, with its numerous neurons and weighted connections, exhibits significant resilience when faced with noisy or flawed datasets. This is primarily due to its ability to estimate nonlinear functions and discern complex relationships within the data sets. Moreover, the design of ANNs enables efficient operation, even when parts of the network fail or falter. One of the primary challenges when working with ANNs, however, is determining the optimal network topology for a specific problem. Once the network's structure is established, it sets a minimum limit for the classification error, which can limit the potential accuracy of the model. Despite this limitation, the ability of ANNs to analyze intricate datasets and extract meaningful insights makes them an invaluable tool in modern recommendation systems [22].

- **Association Rule**: Association rule mining algorithms are particularly effective in the context of "customers also bought" scenarios, a common feature in many online shopping platforms. These algorithms sift through a vast array of transactions to identify patterns, predicting the likelihood of an item being purchased based on the presence of other items within the same transaction. In this context, an association rule takes the form $A \rightarrow B$, with A and B representing distinct item sets [42]. This rule suggests that when items in set A are purchased, items in B will also likely be of interest to the buyer. These algorithms provide a compact representation of preference data, enhancing storage efficiency and system performance. They have proven effective in revealing hidden patterns, informing personalized marketing strategies, and driving significant business growth. Although association rule mining is not yet a mainstream technique in recommendation systems, its effectiveness, particularly in the "customers also bought" feature, has been crucial in boosting sales. By identifying and suggesting items that are likely to be of interest based on past purchases, businesses can stimulate additional sales, thereby increasing their overall revenue.

- **Bayesian Classifiers**: Bayesian classifiers leverage conditional probability and Bayes' theorem to address classification problems. These classifiers are resilient to isolated noise points and irrelevant attributes, and handle missing values effectively. They are particularly useful when user preferences change slowly over time. A popular Bayesian classifier used in recommendation systems is the Naive Bayes Classifier [33]. This classifier presumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. By adopting this assumption, it simplifies computation and is

able to handle high-dimensional data effectively, making it suitable for recommendation systems dealing with large amounts of user data and product features.

These model-based techniques and learning algorithms, when harnessed effectively, can enable collaborative filtering to produce more personalized, efficient, and accurate product recommendations. The ultimate goal is to elevate the user experience by providing pertinent and timely product suggestions, thus driving user engagement and business growth.

2.2.2.2 Memory-Based

Memory-Based Collaborative Filtering is a straightforward method that zeroes in on finding similarities between users or items to suggest products. It's rooted in the idea that users who like the same things are probably quite similar [65]. The heart of this strategy is to spot users who like and dislike the same stuff as the user we're focusing on. After finding these similar users, the system guesses what the user we're focusing on might like based on what these similar users have liked. In essence, users who like similar things are lumped together in a group [14].

This technique is pretty simple and can be set up quite easily, which is why it's a go-to choice for a lot of product recommendation systems. However, it's important to highlight that how well this method works largely depends on the effectiveness and trustworthiness of the similarity measures used. As we continue our discussion, we'll dig deeper into the ins and outs of these measures and why they're so crucial in creating dependable and personalized suggestions [60].

User-Based

User-Based Collaborative Filtering is another essential strategy in the world of collaborative filtering. This method works by measuring how similar users are, based mostly on the ratings they've given to specific products [5]. With this strategy, the system finds users who have similar likes and dislikes based on their product ratings. Once these "taste buddies" are found, the system can guess how a certain user might rate a product they haven't seen before, based on how their taste buddies have rated it.

This method works by making sense of a large grid of users and their product ratings, forming a user-user matrix. Each row in this matrix stands for a user, while each column stands for a product. The points where the rows and columns intersect show the rating a user gave for that product. If a user hasn't rated a product, that intersection point is left blank, indicating a potential spot for a product suggestion.

Though the user-based method is generally effective and has been the backbone of many successful product suggestion systems, it does come with its own set of challenges. For example, it

can require a lot of computational resources, particularly for platforms with many users. It also struggles when dealing with new users (a problem often referred to as the cold-start problem) and changes in user tastes over time [87].

In the sections that follow, we'll dig deeper into these challenges and explore possible solutions to make user-based collaborative filtering more effective in product recommendation systems.

Item-based

Item-based collaborative filtering is a kind of memory-based method specifically created for making product suggestions [60]. The item-based collaborative filtering process can be split into two main parts.

The first part involves figuring out how similar different products are. This is done using various measures of similarity, which are designed to show how alike two items are based on how users have interacted with them. The second part involves using these similarity values to guess ratings for products that the user hasn't seen or used before [5]. By finding products that are similar to ones the user has liked in the past, this method can create personalized product suggestions that are likely to appeal to the user [75].

This strategy can be very effective when there's a wide variety of different products, making it a useful tool in any product suggestion system. However , it's important to note that how well this method works can depend on how reliable the measures of similarity are. We'll be looking more closely at this later in the discussion.

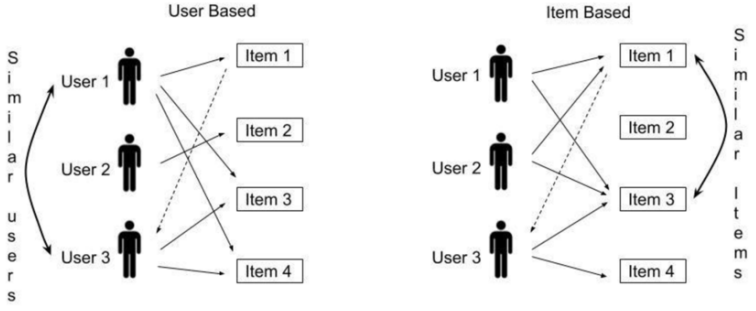

Figure 2.5: Difference between User-Based and Item-Based Collaborative Filtering

2.2.2.3 Hybrid Collaborative Filtering

Hybrid Collaborative Filtering provides an enriched approach to recommendation systems by incorporating various collaborative filtering techniques. This combination aims to provide a more holistic solution to the limitations often encountered in traditional Collaborative Filtering, such as information loss and data sparsity [45].

Essentially, the Hybrid Collaborative Filtering method capitalizes on the strengths of multiple filtering techniques, effectively compensating for the weaknesses of one method with the strengths of another. This blending of techniques can result in more accurate and comprehensive recommendations, leading to an overall enhancement in the quality of the recommendation system.

However , it's worth noting that while Hybrid Collaborative Filtering offers significant benefits, it also introduces its own set of challenges. These systems can be quite complex and costly to implement, considering the resources and computational power required to simultaneously manage and integrate multiple filtering methods. Yet, despite these challenges, the potential benefits of Hybrid Collaborative Filtering in delivering superior recommendations often outweigh these considerations [53].

2.2.2.4 Similarity Metrics in Collaborative Filtering

A crucial aspect of collaborative filtering is the computation of similarity between users or items, which directly impacts the quality of recommendations. The aim is to find like-minded users or similar items based on their ratings or interactions. There are several methods used for computing similarity, and the choice of method can significantly influence the recommendation results.

- **Euclidean Distance** is a commonly used distance metric for measuring similarity in recommendation systems. It calculates the straight line distance between two points (users or items) in a multi-dimensional space. The points represent users or items, and each dimension corresponds to a feature or attribute (for example, a rating for a particular item).

  The formula for Euclidean Distance between two points x and y in an n-dimensional space is given by:

  $$d(x, y) = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2}$$

  Here, n is the number of attributes, and x_k and y_k are the kth attributes of data objects x and y, respectively.

  The square root ensures that the distance is always a positive value. The smaller the Euclidean Distance, the more similar the two points are. Conversely, larger distances imply less similarity.

  In the context of a recommendation system, each user or item can be represented as a point in a multi-dimensional space, where each dimension corresponds to a feature or attribute (for example, a rating for a particular item). The Euclidean Distance between

two points can then be used to measure the similarity between two users or items. This can be particularly useful for methods like k-nearest neighbors (k-NN), where the goal is to find the k points that are most similar to a given point.

- **Cosine Similarity** is another commonly used metric for measuring similarity in recommendation systems. It calculates the cosine of the angle between two vectors , which represent users or items in a multi-dimensional space . Each dimension corresponds to a particular item (in user-based) or a user (in item-based).

  The formula for Cosine Similarity between two vectors x and y is given by:

  $$\text{cosine\_similarity}(x, y) = \frac{x \cdot y}{||x|| \cdot ||y||}$$

  Here, $x \cdot y$ denotes the dot product of vectors x and y, and $||x||$ and $||y||$ are the norms (or lengths) of vectors x and y, respectively.

  The dot product $x \cdot y$ is calculated as:

  $$x \cdot y = \sum_{k=1}^{n} x_k \cdot y_k$$

  And the norm of a vector x is calculated as:

  $$||x|| = \sqrt{\sum_{k=1}^{n} x_k^2}$$

  The Cosine Similarity ranges between -1 and 1. A value of -1 means the vectors are diametrically opposite, 0 means they are orthogonal (no similarity), and 1 means they are identical.

  In the context of a recommendation system, each user or item can be represented as a vector in a multi-dimensional space, where each dimension corresponds to a feature or attribute (for example, a rating for a particular item). The Cosine Similarity between two vectors can then be used to measure the similarity between two users or items. This can be particularly useful for methods like k-nearest neighbors (k-NN), where the goal is to find the k vectors that are most similar to a given vector. [45].

- **Pearson Correlation Coefficient** is a measure that calculates the statistical correlation between two vectors, which can represent users or items in a recommendation system. The correlation coefficient ranges between -1 and 1, where -1 signifies a perfect negative correlation, 0 signifies no correlation, and 1 signifies a perfect positive correlation.

The formula for the Pearson Correlation Coefficient between two vectors x and y is given by:

$$\rho_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

Here, x_i and y_i are the ith elements of vectors x and y, $\bar{x}$ and $\bar{y}$ are the means of vectors x and y, and n is the number of elements in each vector.

The numerator of the formula calculates the covariance of x and y, which measures how much x and y vary together. The denominator calculates the product of the standard deviations of x and y, which measure the amount of variation in each vector.

In the context of a recommendation system , each user or item can be represented as a vector, where each element corresponds to a rating for a particular item (for user vectors) or a user (for item vectors). The Pearson Correlation Coefficient can then be used to measure the similarity between two users or items.

One advantage of the Pearson Correlation Coefficient over other similarity measures like Cosine Similarity is that it adjusts for the mean ratings of each user, accounting for the bias in user ratings. For example, some users might tend to give higher ratings than others, and the Pearson Correlation Coefficient can adjust for this bias when calculating similarities [31].

- **Jaccard Index**, also known as the Jaccard similarity coefficient , is a measure typically used in the case of binary ratings (like or dislike, buy or not buy). It is defined as the size of the intersection divided by the size of the union of two sets. These sets represent the preferences of two users or items.

  The formula for the Jaccard Index between two sets u and v is given by:

  $$J(u, v) = \frac{|u \cap v|}{|u \cup v|}$$

  Here, $|u \cap v|$ represents the size of the intersection of sets u and v (i.e., the number of items that are in both sets), and $|u \cup v|$ represents the size of the union of sets u and v (i.e., the number of items that are in either set or in both).

  The Jaccard Index ranges from 0 to 1. A value of 0 means that the two sets have no items in common, and a value of 1 means that the two sets are identical.

  In the context of a recommendation system, each user or item can be represented as a set, where each element corresponds to a liked item (for user sets) or a user who liked the item (for item sets). The Jaccard Index can then be used to measure the similarity between two users or items.

One advantage of the Jaccard Index is that it is easy to understand and calculate. However, it is only suitable for binary ratings and does not take into account the magnitude of ratings or the order of items [31].

- **Hamming Distance** is a measure used to calculate the difference between two binary vectors, typically of equal length. It is defined as the number of positions at which the corresponding values are different.

  Given two binary vectors x and y of length n, the Hamming Distance d between x and y is given by:

  $$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$

  Here, x_i and y_i are the ith elements of vectors x and y, respectively, and n is the number of elements in each vector.

  In the context of a recommendation system, each user or item can be represented as a binary vector, where each element corresponds to a binary rating for a particular item (for user vectors) or a user (for item vectors). The binary rating could represent like or dislike, buy or not buy, etc. The Hamming Distance can then be used to measure the dissimilarity between two users or items [31].

  One advantage of the Hamming Distance is that it is easy to understand and calculate. However, it is only suitable for binary ratings and does not take into account the magnitude of ratings or the order of items. It's also worth noting that a smaller Hamming Distance indicates higher similarity, which is the opposite of some other distance measures like Euclidean Distance.

Choosing the right similarity measure is vital for the effectiveness of a collaborative filtering system. The choice often depends on the nature of the data, the sparsity of the user-item matrix, and the computational resources available.

2.2.2.5 Advantages and Disadvantages of Collaborative Filtering

Like any other technique, CF is not exempt from challenges. Understanding its strengths and limitations is crucial for maximizing its potential in recommendation systems.

Pros:

- **Absence of a knowledge database**: CF does not require a specific database to store the characteristics of the items or users. The recommendations are entirely based on the users' ratings, eliminating the need for additional information about the users or items. This makes CF more flexible and adaptable to various contexts and domains.

- **Improving quality over time**: As more users interact with the system and provide ratings, the quality of CF recommendations continually improves. The system becomes increasingly accurate and effective as it learns from an expanding pool of user feedback.
- **No need for domain knowledge**: CF does not require explicit domain knowledge to generate recommendations. It solely utilizes users' ratings, freeing it from the need to understand the specifics of the items or the industry. This makes CF highly versatile and applicable to a broad array of fields and sectors.
- **Unexpectedly Suitable Recommendations**: Collaborative Filtering (CF) excels at presenting suggestions that, although surprising, are still relevant. It has the capacity to propose items that may not be evident in a user's profile, yet might still spark their interest, creating a pleasing sense of surprise and exploration.
- **Handling less content-rich items**: CF shines in areas where there is limited content associated with items, or where the content is difficult for a computer system to analyze, such as personal opinions or ideals.

Cons:

- **Cold start problem**: When a new user visits with no prior records, the CF approach struggles, as it requires a substantial amount of information to offer accurate recommendations.
- **Scalability issues**: As the number of users and items grows exponentially, the computational demands for recommendations increase. Systems must provide suggestions even when there's a lack of past history or ratings, requiring considerable computational power.
- **Data sparsity**: On e-commerce sites, while a large number of items are sold, not every item is rated by every user, resulting in a small portion of the database being rated, which leads to sparse data.
- **Synonymy**: This issue arises when the same item has different names, making it hard for the recommendation system to discover latent associations. As a result, these items are considered as different, disrupting the recommendation process.
- **Gray sheep problem**: Users who neither like nor dislike items within a group don't benefit from the recommendation system, as their preferences don't align with the majority.
- **Shilling attack**: This refers to 'profile injection' attacks , where fake profiles that can manipulate the recommendation process. Users might give positive reviews for their own products and negative reviews for their competitors, skewing the recommendations.

In conclusion, while CF techniques hold great promise and offer unique advantages, they also come with significant challenges that can impact their performance. These problems need to be addressed to maximize the effectiveness of recommendation systems and ensure they provide accurate, relevant suggestions to users. As recommendation systems evolve, finding new strategies to tackle these obstacles will remain a vital area of research and development.

2.2.3 Hybrid Recommendation Systems

Hybrid recommendation systems are born from a blend of two or more of the recommendation approaches we've discussed earlier. By uniting these traditional approaches, hybrid systems aim to overcome individual limitations and capitalize on their collective strengths [16]. Depending on the specific needs for recommendations, various combinations can be applied.

Indeed, many of today's applications leverage hybrid approaches to address the shortcomings of both collaborative filtering (CF) and content-based filtering (CBF), forming the most commonly used combination [2]. Hybrid recommendation systems promise to elevate the accuracy and efficiency of the recommendation process by mitigating the drawbacks of both filtering techniques [23].

These hybrid strategies can be brought to life in several ways [16]:

- Employing content-based and collaborative filtering algorithms independently, and then merging their recommendations. This combination offers a well-rounded view for users, addressing the unique benefits of each technique.
- Integrating elements of the content-based method into the collaborative filtering technique. This approach can help address the "cold start" issue in collaborative filtering, where a new user lacks a history of likes and dislikes for the system to base its recommendations on.
- Incorporating some collaborative features into the content-based filtering method. This approach helps to tackle the overspecialization problem in CBF, where the system might only recommend items too similar to what the user has already shown interest in.
- Creating a model that includes features from both content-based and collaborative filtering techniques. This method enhances the accuracy and efficiency of the recommendations by using the best features of both CBF and CF [14].

By adopting such a hybrid approach, recommendation systems can provide more accurate, effective, and personalized suggestions, thereby significantly improving user experience.

2.2.3.1 Hybridization Types

- **Weighted Hybridization** in recommendation systems blends scores from various techniques, creating a unified prediction or recommendation list. An instance of this is P-tango [27], a system that uses both content-based and collaborative recommenders. Initially, these are equally weighted, but adjustments occur based on prediction confirmations or errors. This approach, simple yet efficient, taps into the strengths of all the involved recommendation systems.[35]

- **Switching Hybridization** smartly moves between different ways of making recommendations based on a rule that decides the best strategy for coming up with better product suggestions. This flexible model can cleverly deal with problems tied to one method by changing between techniques. For example, when faced with the challenge of new users in a content-based recommendation system [71], it can switch to a recommendation system that works together. A big plus of this method is its awareness of the strong and weak points of the recommendation techniques it uses. However, the extra difficulty in deciding the right time to change techniques is a big challenge, as it can increase the number of settings in the system.

- **Cascade Hybridization** is a method that improves recommendations step by step, creating a detailed ranking among different products. This process involves one recommendation method creating an initial list of suggestions, which is then improved by a second method. The result is a more refined and focused list of product recommendations. This hybridization approach is praised for its efficiency and ability to resist errors, thanks to its gradual improvement procedure. An example of a system that uses Cascade Hybridization is EntreeC [16], which uses a cascading knowledge-based and collaborative recommender to improve the quality of its product suggestions.

- **Mixed Hybridization** employs multiple recommendation techniques to deliver diverse suggestions for each item.Instead of a single recommendation, each product comes with a range of suggestions from various methods [57]. This method keeps working well even if one technique doesn't do so well. An example could be a system that combines user-user collaborative filtering, item -item collaborative filtering, and content-based filtering, offering a broad range of product suggestions. While it increases variety, this technique may make things more complicated because it needs to manage multiple recommendation algorithms.

- **Feature Combination Hybridization** in product recommendation systems allows for the analysis of cooperative data, reducing dependence solely on user product ratings [71]. This approach offers insight into the inherent similarities between products, which might not be evident through a solely collaborative method [16]. In this method, an

initial recommendation technique assigns rough ratings to potential products. Then, a second technique refines these recommendations. It is particularly effective for sparse datasets. It allows systems to bypass the use of a second, lower-priority strategy for products that have already been well differentiated by the first technique or for those that are so poorly rated they are unlikely to be recommended.

- **Feature Augmentation Hybridization** adds extra characteristics to the main recommender, instead of combining features. This method, often used to improve how well content-based filtering works, creates a ratio based on the classification of user or product profiles. This method uses the ratings and other data from the previous recommender and asks for more features from the recommendation systems [42].

- **Meta-Level Hybridization** uses the internal model formed by one recommendation technique as input for another in product recommendation systems. This process generates a model that's more information-rich than a single rating. By using the entire model from the first technique as input for the second, Meta-Level hybrids address the sparsity issue often found in collaborative filtering methods [17].

Hybridization Method	Description
Weighted	Merges scores (or votes) from multiple recommendation methods to generate a single suggestion.
Switching	The system shifts between recommendation methods depending on the current scenario.
Cascade	One recommender builds on the recommendations provided by another.
Mixed	Provides recommendations from multiple different recommenders simultaneously.
Feature Combination	Develops a single recommendation algorithm by integrating features from various recommendation data sources.
Feature Augmentation	One technique's output is employed as a feature in another's input.
Meta-Level	One recommender's model gets used as input for another recommender.

2.2.3.2 Advantages and Disadvantages of Hybrid Recommenders

The effectiveness of Hybrid recommendation systems largely depends on the particular blend of recommendation methods chosen.

Pros:

Hybrid recommendation systems leverage the strengths of the combined methodologies to create a system that reduces their individual drawbacks. For example, a system could integrate Collaborative Filtering and Content-Based Filtering strategies to capitalize on both the item representation advantages of CBF and the user similarity aspects of CF. This way, the system can counter the new-item cold-start issue typically encountered in CF systems by using CBF's item representation. At the same time, it can alleviate the Serendipity problem often seen in CBF systems by taking advantage of CF's user similarity feature.

Cons:

Despite their aim to address the issues associated with individual recommendation methods, Hybrid recommendation systems come with their own challenges. Selecting the most suitable combination for the recommendation system and merging these approaches is a complex task. This decision should be made after considering the advantages and disadvantages of each methodology and the specific needs of the application at hand. Thus, while hybrid systems can provide enhanced recommendation results, the complexity of designing and implementing such systems should not be underestimated.

2.3 Evaluation Metrics

Evaluating the performance of a recommendation system is crucial to understand its effectiveness and to identify areas for improvement . Several metrics are used to measure the accuracy , relevance, and diversity of the recommendations provided by the system. Here, we discuss some of the most commonly used evaluation metrics in recommendation systems [75] [42]

2.3.1 Mean Absolute Error

Mean Absolute Error (**MAE**) is a popular metric used in recommendation systems to measure the average absolute difference between the predicted and actual ratings. It is calculated as:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|r_i - \hat{r}_i|$$

Here, r_i is the actual rating, $\hat{r}_i$ is the predicted rating, and N is the total number of ratings. Lower values of MAE indicate higher accuracy of the recommendation system. In the context of product recommendation systems, MAE can be used to evaluate how well the system is able to predict the ratings a user would give to a product. For instance, if a user rated a product as 4 stars and the system predicted it as 5 stars, the absolute error would be 1. The MAE is then the average of these absolute errors across all predictions. A lower MAE would mean that the system is more accurate in predicting user ratings [66].

2.3.2 Root Mean Square Error

Root Mean Square Error (**RMSE**) is another commonly used metric in recommendation systems. It is similar to MAE but gives more weight to larger errors. It is calculated as:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(r_i - \hat{r}_i)^2}$$

Like MAE, lower values of RMSE indicate higher accuracy of the recommendation system. However, because the differences between the actual and predicted ratings are squared before they are averaged, RMSE gives a relatively high weight to large errors. This means that RMSE should be more useful when large errors are particularly undesirable. In the context of product recommendation systems, a large error could mean that a product is highly recommended to a user even though the user ends up disliking it. Therefore, a system with a lower RMSE would be more reliable in its recommendations [79].

2.3.3 Precision and Recall

Precision and Recall are two fundamental measures in information retrieval and are also used in recommendation systems. Precision measures the proportion of recommended items that are relevant, while Recall measures the proportion of relevant items that are recommended. They are defined as:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

Here, TP is the number of true positives (relevant items that are recommended), FP is the number of false positives (irrelevant items that are recommended), and FN is the number of false negatives (relevant items that are not recommended). In the context of product recommendation systems, Precision can be interpreted as the probability that a recommended product is liked by the user, while Recall can be interpreted as the probability that a product liked by the user is recommended by the system [43].

2.3.4 F1 Score

The F1 Score is the harmonic mean of Precision and Recall, providing a single measure that balances both values. It is particularly useful when you want to compare two or more models or when you want to find a balance between Precision and Recall. It is defined as:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

A higher F1 Score indicates a better balance between Precision and Recall and therefore a better overall performance of the recommendation system [80].

2.3.5 Precision-Recall Curve

The Precision-Recall curve is a plot of the precision (y-axis) and the recall (x-axis) for different thresholds, much like the **ROC Curve**. A high area under the curve represents both high recall and high precision, where high precision relates to a low false positive rate, and high recall relates to a low false negative rate. High scores for both show that the classifier is returning accurate results (high precision), as well as returning a majority of all positive results (high recall).

In the context of product recommendation systems, a high precision but low recall system would mean that the system recommends very few products but most of them are liked by the users. On the other hand, a low precision but high recall system would mean that the

system recommends many products, and most of them are not liked by the users. Therefore, the Precision-Recall curve provides a comprehensive view of the balance between precision and recall for different threshold levels [76].

2.3.6 Coverage

Coverage is a measure of the ability of a recommendation system to offer a recommendation to a user. It is often expressed as a percentage of users for whom the system can provide recommendations. In the context of product recommendation systems, coverage can be interpreted as the proportion of products that can be recommended by the system. A high coverage means that the system can recommend a large variety of products , which is particularly important in situations where the product catalog is large and diverse [42]. Coverage can be calculated as:

$$Coverage = \frac{Number\ of\ items\ recommended}{Total\ number\ of\ items}$$

2.3.7 Ranking Metrics

In some recommendation systems, the goal is not just to recommend a list of items, but also to rank them according to their relevance. In such cases, traditional metrics like precision and recall may not be sufficient, as they do not take into account the order of the items in the recommendation list. Ranking metrics such as **Normalized Discounted Cumulative Gain** (NDCG), **Precision** at K, and **Mean Reciprocal Rank** (MRR) are often used in these cases.

For example, NDCG is a measure of the effectiveness of a ranking algorithm. It calculates the gain of each recommendation , discounted at increasing logarithmic rate. The gain is accumulated from the top of the list to the bottom, with the gain of each result discounted at the corresponding rank. If all the recommended items are relevant , NDCG is equal to 1. If none of the recommended items are relevant, NDCG is equal to 0 [69].

Chapter 3

Benchmark Studies

3.1 Importance of Benchmarking

As we discussed, benchmark studies play a vital role in the domain of recommendation systems, contributing to both academic pursuits and practical business implementations. They are like the points that bind the theoretical improvements in recommendation algorithms to the real impacts they carry for e-commerce businesses, no matter their size . Recommendation systems bring substantial value to the e-commerce sector. They navigate users toward products of potential interest , thus enhancing user experiences, promoting customer engagement, and ultimately, increasing sales. However, the success of these systems is deeply rooted in the quality and relevance of the recommendations they provide, which is mostly steered by the employed recommendation algorithm.

The significance of benchmark studies cannot be underestimated when it comes to discussing this matter. They offer e-commerce companies a standardized platform to assess the effectiveness of different recommendation algorithms, empowering them to make well-informed decisions supported by data. These decisions are crucial in determining which algorithm to implement. Benchmark studies shine a spotlight on various aspects, such as algorithm scalability , their ability to handle sparse or cold-start issues, and their capacity to provide accurate and pertinent recommendations . By delivering vital insights, these studies assist e-commerce businesses in refining their recommendation systems.

For larger e-commerce enterprises with an extensive range of products and a substantial customer base, the choice of the right recommendation algorithm can make all the difference. It can create personalized and engaging shopping experiences, or it can overwhelm customers with unrelated product recommendations. The stakes are even higher for smaller e-commerce ventures. Due to limited resources, making the correct decision is critical in terms of cost-efficiency and growth potential.

Benchmark studies also play a vital role in facilitating ongoing improvements. By identifying the weaknesses and gaps in existing algorithms, these studies can inspire the development of innovative solutions tailored to the specific needs of e-commerce businesses. In the subsequent sections, we will delve into the existing benchmark studies within the domain of recommendation systems. We will highlight their discoveries, methodologies, and particularly focus on the unaddressed gaps, especially concerning product recommendation algorithms.

3.2 General Recommendation Benchmarks

Benchmark studies for recommendation systems span across various domains, each having their unique datasets and challenges. These domains extend from movies and music to news, books, and more.

3.2.1 MovieLens Bechmarks

4.2.1 MovieLens Benchmarks In the domain of movie recommendations, one of the most prominent datasets is the MovieLens dataset, which was created by the GroupLens Research Lab at the University of Minnesota. The MovieLens dataset is widely recognized for its various versions, each containing extensive user rating data. The dataset has been instrumental in studying and benchmarking algorithms, particularly collaborative filtering and matrix factorization methods [37].

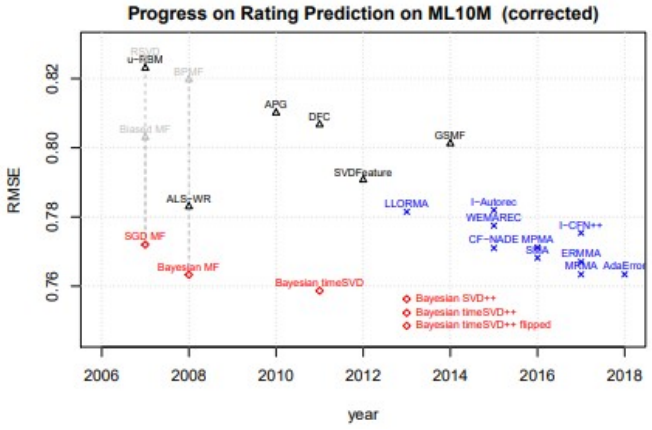

FIGURE 3.1: RMSE Comparison of recommendation algorithms on ML10M Dataset

Specifically, Movielens 10M has been a reliable benchmark for assessing rating prediction methods, especially through the measurement of Root Mean Square Error (RMSE) on a 90:10 split [49] [77] [21] [88]. Progress over time has shown that newer methods consistently outperform older ones, such as matrix factorization or Boltzman machines [74].

Upon closer inspection of Biased MF, RSVD, ALS-WR, and BPMF, several observations come to light. For instance, Biased MF [48] and RSVD are essentially the same method, differing only in aspects like hyperparameters, training data ordering, and implementations. Additionally, ALS-WR [89] and Biased MF/RSVD, though identical models, are learned by different algorithms and show similar results when optimally tuned. Surprisingly, BPMF [73] which shares its model with RSVD/ALS-WR but learned with a Gibbs sampler, underperforms in comparison.

Re-running the baseline models yielded interesting results. For instance, the SGD-MF model achieved an RMSE of 0.7720 for a 512-dimensional embedding and an RMSE of 0.7756 for 64 dimensions, outperforming the reported values for RSVD (0.8256) and Biased MF (0.803). Moreover, the Bayesian MF model using Gibbs sampling (similar to BPMF) achieved an RMSE score of 0.7633 for a 512-dimensional embedding, surpassing previously reported number for BPMF (0.8197) [52].

In summary, these findings question conclusions drawn from previous experimental results on ML10M and suggest the importance of revisiting established methods with a carefully set up baseline. This approach allowed for the outperforming of any result, even with a simple Bayesian matrix factorization, a method previously reported to perform poorly on ML10M.

3.2.2 Netflix Prize Benchmarks

The Netflix Prize was a contest aiming to decrease by 10% the RMSE of Netflix's recommender system, Cinematch, with an original score of 0.9514. This effort took approximately three years and utilized hundreds of ensembled models [11]. Participants worked with a dataset split into training, validation (probe set), and testing (qualifying set), where the test ratings were unknown during the competition. Ratings were separated based on user and time, introducing a forecasting challenge that complicated the task [10].

Matrix factorization algorithms were among the early promising methods, with traditional SVD solvers achieving RMSEs close to 0.94 [11]. A significant development was FunkSVD, a sparse matrix factorization algorithm that achieved a probe RMSE of 0.93. Continued work led to lower RMSE scores of 0.8985 for ALS and 0.8998 for SGD methods [89]. The community eventually shifted focus towards complex models, like SVD++ [47] and timeSVD++ , and ensemble models, which led to the winning solutions. A variety of models, like nearest neighbor models and Restricted Boltzmann Machines, were also employed in this combination approach.

In contrast to ML10M, the Netflix Prize encouraged re-running and refining methods. This approach motivated the community to understand which methods work best. Experiments revealed that poorly performing methods on ML10M actually did well, suggesting that the key is in careful tuning and significant experimentation effort. The issues raised are not specific

to ML10M; the lack of a standardized benchmark poses a risk of questionable experimental findings across recommendation systems research.

3.2.3 Diverse Applications Benchmarks

More generally, in the field of music recommendations, the **Million Song Dataset** (MSD) has been a significant resource. This dataset includes audio features and metadata for a million contemporary popular music tracks. The dataset has been extensively used in studies focusing on content-based filtering algorithms and provides a valuable contribution in this area by showcasing the effectiveness of hybrid techniques that combine audio and text data [62].

News recommendation is another area where benchmarks have played a critical role. The **Outbrain Click Prediction** dataset, for instance, contains user click data and document metadata, allowing for comprehensive experiments on news recommendation techniques. It has been used to benchmark several algorithms focusing on contextual bandits and sequence-aware recommendation techniques [44]

In the realm of advertisement recommendation, the **Criteo** dataset stands out. This dataset is used for display advertising and contains over a week's worth of Criteos real-world traffic data, including click-through records and conversion rates [63]. The dataset has facilitated research into click-through rate prediction models, which are pivotal for optimizing ad recommendations. This study [20] leverages this dataset to evaluate several machine learning algorithms, highlighting the importance of feature engineering and providing guidelines for model selection in this context.

While this review extensively delves into various domains of recommendation system benchmarks , it's imperative to remember that each area carries unique challenges and traits. As a result, the insights derived from these benchmarks should be contextualized within their unique application domain. Furthermore, we have curated this broad overview of diverse recommendation systems, studying how different benchmarks operate across various sectors. This exploration is designed to facilitate our ongoing effort to craft an effective benchmark for product recommendation systems. In the following section, we will further probe into existing benchmarks specific to product recommendations, deepening our understanding and strengthening our methodology.

3.3 Product Recommendation Benchmarks

Product recommendation systems are of paramount importance for e-commerce stores, enhancing the online shopping experience through personalization, thereby boosting customer

satisfaction and business revenue. Despite their significance, there is a noticeable scarcity of benchmark studies specifically focused on product recommendation. This domain presents a unique set of challenges and opportunities that are distinct from other recommendation system domains.

While there have been attempts to create benchmarks for product recommendation, such as using the Amazon product data, these efforts have not fully addressed the unique challenges in this domain . The Amazon product data, which includes product reviews, product metadata, and links to view product images, has been used in several studies to evaluate the performance of various recommendation algorithms but not on a precise manner that studies and tests both Collaborative Filtering and Content Based Filtering algorithms on the same dataset and compares each algorithm to find their advantages and where each one lacks at.

In this study, we aim to fill this gap by proposing a benchmark study specifically tailored for product recommendation systems on the Amazon dataset format. We will apply a range of recommendation algorithms, and evaluate their performance using relevant metrics. Our goal is to contribute to the ongoing research in product recommendation and provide valuable insights for e-commerce businesses and researchers alike. In the following sections, we will analyze the methodology and implementation of our benchmark study.

Chapter 4

Methodology & Implementation

In order to address the research questions, we designed and followed a methodology that is standard in the domain of recommendation systems. This included stages of data collection, data preparation and pre-processing, choosing and detailing algorithms , selecting evaluation metrics, specification the research design, performing statistical analysis and identifying probable limitations. Each of these phases are paramount in establishing an objective and trustworthy recommendation system.

This allowed us to draw upon the large quantity of available data from Amazon reviews, making use of machine learning techniques to build and evaluate the recommendation models. Furthermore, our methodology aims to fill in the gaps identified in the current benchmark literature. By utilizing the best practices and approaches in recommendation system research, and adapting them to address the specific challenges and opportunities identified in the literature, our study seeks to advance the field and contribute to a more nuanced and effective understanding of recommendation systems.

4.1 Data Collection

The data utilized in this study is the Amazon Product Reviews dataset, specifically focusing on the electronics category from the year 2018. This dataset is a rich source of customer feedback , having a wide array of information such as product ratings, review texts, and additional metadata about the products. The electronics category was chosen for this study due to its relevance and popularity among consumers. Electronics are a significant part of the e-commerce market, and the reviews and ratings provided by customers in this category can offer valuable insights into their preferences and purchasing behavior. By leveraging both the reviews and metadata from the Amazon Product Reviews dataset, this study aims to develop a comprehensive benchmark on product recommendation systems, focusing on both collaborative and content-based filtering methods.

Reviews Dataset:

Attribute	**Value**
Reviewer ID	AUI6WTTT0QZYS
Product ID	5120053084
Reviewer Name	Abbey
Vote	2
Review Text	I now have 4 of the 5 available colors of this shirt...
Overall Rating	5.0
Summary	Comfy, flattering, discreet–highly recommended!
Unix Review Time	1514764800
Review Time	01 1, 2018
Image	`https://images-na.ssl-images-amazon.com/images/I/71eG75FTJJL.SY88.jpg`

Table 4.1: Reviews Dataset Sample

Where:
reviewerID - ID of the reviewer, e.g. A2SUAM1J3GNN3B
asin - ID of the product, e.g. 0000013714
reviewerName - name of the reviewer
vote - helpful votes of the review
reviewText - text of the review
overall - rating of the product
summary - summary of the review
unixReviewTime - time of the review (unix time)
reviewTime - time of the review (raw)
image - images that users post after they have received the product

Product Metadata Dataset:

Attribute	Value
Product ID	0000031852
Title	Girls Ballet Tutu Zebra Hot Pink
Feature	Botiquecutie Trademark exclusive Brand, Hot Pink Layered Zebra Print Tutu, Fits girls up to a size 4T, Hand wash / Line Dry, Includes a Botiquecutie TM Exclusive hair flower bow
Description	This tutu is great for dress up play for your little ballerina. Botiquecute Trade Mark exclusive brand. Hot Pink Zebra print tutu.
Price	3.17
Image URL	`http://ecx.images-amazon.com/images/I/51fAmVkTbyL.SY300.jpg`
Also Buy	B00JHONN1S, B002BZX8Z6, B00D2K1M3O, 0000031909, B00613WDTQ, B00D0WDS9A, B00D0GCI8S, 0000031895, B003AVKOP2
Also Viewed	B002BZX8Z6, B00JHONN1S, B008F0SU0Y, B00D23MC6W, B00CEV8366, B003AVEU6G, B00HSOJB9M, B00B0AVO54, B00E95LC8Q, B0046W9T8C, B00E1YRI4C, B00EHAGZNA
Brand	Coxlures
Categories	Sports & Outdoors, Other Sports, Dance

Table 4.2: Product Metadata Dataset Sample

Where:

asin - ID of the product, e.g. 0000031852

title - name of the product

feature - bullet-point format features of the product

description - description of the product

price - price in US dollars (at time of crawl)

imageURL - url of the product image

related - related products (also bought, also viewed, bought together, buy after viewing)

brand - brand name

categories - list of categories the product belongs to

For this study, we considered specific attributes from the raw data before to prepare two CSV files: **electronics.csv** and **electronics_meta.csv**. The electronics.csv file includes the reviews dataset, comprising reviewerID (**user ID**), asin (**product ID**), overall (**rating number**), and unixReviewTime (**timestamp of review**) attributes. This dataset is intended to be used for collaborative filtering and understanding user relevance. The electronics_meta.csv file includes metadata for each product, including asin(**product ID**), **category**, **title**, **also_buy**, **brand**, **also_view**, main_cat (**main category**), and **price** attributes. This dataset will be utilized for content-based filtering and assessing product similarity.

4.2 Data Preparation and Processing

After the data collection phase, we proceeded with data preparation and processing, the key steps to making the raw data suitable for analysis. This involved cleaning, filtering, and manipulating the data to suit the needs of our research.

First, we performed a data cleaning process to remove unnecessary columns from the raw data. This was an important step because extraneous data can introduce noise and make it harder to identify useful patterns. Not all attributes in the raw data were relevant to our study. For example, fields such as 'image,' 'verified,' 'reviewTime' in the review dataset, and 'feature,' 'description,' 'imageURL,' 'imageURLHighRes,' 'salesRank,' 'categories,' and 'similar' in the metadata were not essential for our recommendation systems and therefore removed. Following the initial data cleaning, we proceeded to refine the datasets to suit our needs better.

For the **electronics.csv** dataset, designed for collaborative filtering, we retained only the most essential attributes for understanding user behavior and evaluating the product's worth. The columns maintained were:

'**reviewerID**': This is the unique identifier for each user, which we renamed to '**user_id**' for easier understanding and readability.
'**asin**': This is the unique identifier for each product, which we renamed to '**product_id**' to make it more intuitive.
'**overall**': This is the rating given by the user for a particular product, which we renamed to '**rating_number**' for clarity.
'**unixReviewTime**': This is the time the review was made, which we renamed to '**timestamp**' for better interpretability.
By focusing only on the data that provides valuable insights into user-product interactions, we can build a more efficient recommendation system. After data cleaning and renaming columns, the data preparation stage was completed, and we were ready to advance to the next phase of our research methodology.

The **electronics_meta.csv** dataset, on the other hand, contains detailed metadata for each product and is intended for content-based filtering and building similarity-based recommendations. In this dataset, the columns maintained were:

'**asin**': This is the unique identifier for each product, which we renamed to '**product_id**' to maintain consistency across datasets.
'**brand**': This specifies the product's brand.
'**title**': This contains the name of the product. We maintained this column as it is crucial for identifying and describing the product.
'**also_buy**': This column lists IDs of products that are often bought together with the product

in question. It could be useful for generating "**Users also bought**" recommendations.
'also_view': This lists product IDs that are often viewed together with the product in question, which could be valuable for offering "Users also viewed" suggestions based on browsing behavior.
'**main_cat**': This is the main category of the product. We preserved this attribute, considering that similar categories may imply similar products.
'**price**': This is the price of the product, which could play a role in recommendations, especially when considering users' affordability.

By retaining these columns, we ensured the availability of relevant information to support the content-based recommendation algorithm and provide diverse and accurate suggestions to users.

Upon completing the selection and renaming process for the electronics_meta.csv dataset, we performed a sanity check on both datasets to ensure the data's consistency and reliability. This step is paramount to confirm the datasets' readiness for the subsequent stages in the methodology. Also, for collaborative filtering, we divided the electronics.csv dataset into a **75:25 ratio**, forming our training and test sets respectively. This ratio is frequently used in machine learning as it ensures a substantial amount of data is available for training the model (75%), while still leaving a good portion for testing the model's performance (25%).

The process of data preparation and processing is crucial in recommendation systems, as it shapes the data into a suitable format for further analysis and model building . The goal is to make the data as clear and relevant as possible for the algorithms to generate meaningful and accurate recommendations.

4.3 Description of Algorithms

Due to the differences in datasets and algorithmic requirements, I decided to separate the discussion of these two categories of algorithms. This separation allows for a more focused exploration and understanding of each type of recommendation system and their respective intricacies.

In the following sections, we'll start by detailing the algorithms used in the Collaborative Filtering recommendation system and then move onto those used in the Content-Based Filtering system. Subsequently, we will present and discuss the results obtained from each recommendation system.

4.3.1 Collaborative Filtering Algorithms

As we previously discussed, CF is a popular recommendation type of algorithms that bases its predictions and recommendations on the past behavior of users. The underlying assumption of the CF approach is that those who agreed in the past tend to agree again in the future.

4.3.1.1 Memory-Based Algorithms

Memory-based collaborative filtering approaches can be thought of as a "neighborhood" based approach. In this case, the neighborhood is formed by selecting a set of users (in user-based CF) or a set of items (in item-based CF) based on their similarity to the user or item for which we need to make a prediction. These methods are primarily heuristic-based, relying on past user-item interactions to generate recommendations. They include the commonly used User-Based and Item-Based collaborative filtering techniques. Later, We will benchmark both User-Based and Item-Based implementation of these algorithms.

- **KNNBaseline**: This is a collaborative filtering algorithm that considers a baseline rating. The baseline rating is a combination of the user bias, item bias, and the global mean rating. The predicted rating is then a weighted sum of the baseline ratings and the similarities of the users/items. The user bias is the deviation of a user's average rating from the global average rating, while the item bias is the deviation of an item's average rating from the global average rating. The baseline rating for a user-item pair is calculated as:

$$b_{ui} = \mu + b_u + b_i$$

  where b_{ui} is the baseline rating, μ is the global average rating, b_u is the user bias, and b_i is the item bias.

- **KNNBasic**: This is a straightforward application of the k-nearest neighbors approach for collaborative filtering. It calculates the rating prediction for a user-item pair by taking the average of the ratings given by the k most similar users/items to the given user/item. The predicted rating r_{ui} is calculated as:

$$r_{ui} = \frac{1}{k} \sum_{j \in N_i^k(u)} r_{uj}$$

  where $N_i^k(u)$ is the set of k users most similar to user u who have rated item i, and r_{uj} is the rating of user u for item j.

- **KNNWithMeans**: This is a modification of the basic k-nearest neighbors approach that takes into account the mean ratings of each user. The predicted rating is calculated by adding the user's mean rating to the average of the deviation of the k most similar users' ratings from their mean ratings. The predicted rating r_{ui} is calculated as:

$$r_{ui} = \bar{r}_u + \frac{1}{k} \sum_{j \in N_i^k(u)} (r_{uj} - \bar{r}_j)$$

  where $\bar{r}_u$ is the mean rating of user u, and $\bar{r}_j$ is the mean rating of user j.

- **KNNWithZScore**: This is another modification of the k-nearest neighbors approach. It normalizes the ratings of a user using the z-score normalization, which subtracts the mean rating and divides by the standard deviation. The predicted rating r_{ui} is calculated as:

$$r_{ui} = \bar{r}_u + \sigma_u \frac{1}{k} \sum_{j \in N_i^k(u)} \frac{(r_{uj} - \bar{r}_j)}{\sigma_j}$$

  where σ_u is the standard deviation of the ratings of user u, and σ_j is the standard deviation of the ratings of user j.

Algorithm	**Key Difference**
KNNBaseline	Incorporates user and item biases in addition to the global mean rating
KNNBasic	Applies the k-NN approach without any additional adjustments
KNNWithMeans	Adjusts for the mean ratings of each user
KNNWithZScore	Normalizes ratings using z-score normalization

TABLE 4.3: Differences among k-NN based algorithms

These algorithms are all implemented in the Surprise library, which is a Python scikit for building and analyzing recommendation systems. The library provides various ready-to-use prediction algorithms including these k-NN based algorithms, and also offers tools for evaluating, analysing, and comparing the performance of different algorithms.

4.3.1.2 Model-Based Algorithms

Model-based collaborative filtering methods involve a step to learn or train a model from the given data before making a prediction. The model is a mathematical representation that captures patterns in the data. The advantage of this approach is that the learned model can generalize to new data and can provide a rating prediction for previously unseen user-item pairs.

Model-based CF methods tend to be more scalable and can deal with higher sparsity level than memory-based models, but they also involve more complex computations. This makes them

particularly suitable for large-scale datasets, such as the Amazon product dataset used in our research.

In our research, we use several model-based collaborative filtering algorithms:

- **Singular Value Decomposition (SVD)** is a matrix factorization technique that is widely used in collaborative filtering for product recommendation systems[85]. The mathematical representation of SVD is as follows[55]:

  Given a matrix M, it can be decomposed into three matrices U, Σ, and V^T such that:

  $$M = U \cdot \Sigma \cdot V^T$$

  Where:

  - U is an $m \times m$ orthogonal matrix
  - Σ is an $m \times n$ diagonal matrix with non-negative real numbers on the diagonal
  - V^T (the transpose of V) is an $n \times n$ orthogonal matrix

  The diagonal entries σ_i of Σ are known as the singular values of the original matrix M . The columns of U and the columns of V are called the left-singular vectors and right-singular vectors of M, respectively.

  In the context of product recommendation, the original matrix M represents the user-item interactions , where each entry m_{ij} represents the rating given by user i to item j . The matrix U represents the latent features of the users, and the matrix V represents the latent features of the items. The singular values in Σ represent the strength of each latent feature.

  The SVD algorithm predicts the missing entries in M by calculating the dot product of the three matrices U, Σ, and V^T. These predicted ratings can then be used to recommend products that a user has not yet rated but is likely to rate highly based on their latent features.

  It's important to note that SVD assumes that the missing ratings in the user-item matrix are missing completely at random, which may not always be the case. Furthermore, SVD can be computationally expensive, particularly for large datasets. Despite these challenges, SVD remains a popular and effective method for product recommendation[48].

- **SVD++ (SVDpp)** is an extension of the Singular Value Decomposition (SVD) algorithm that incorporates implicit feedback in addition to the explicit ratings. Implicit feedback refers to the indirect signals of user preference, such as browsing history, click patterns, purchase history, etc. These implicit signals can provide additional information about user preferences, especially when explicit ratings are sparse. Since our dataset doesn't provide these feedback, the algorithm doesnt differentiate a lot to the original.

  The key idea behind SVDpp is to model the implicit feedback by introducing additional factors in the SVD model. These factors capture the effect of items that a user has rated on the user's overall rating behavior. The inclusion of these factors allows SVDpp to capture more complex user-item interactions and often leads to more accurate rating predictions[47].

  The mathematical representation of SVDpp is similar to SVD, but with an additional term to account for the implicit feedback:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T(p_u + |N(u)|^{-0.5} \sum_{j \in N(u)} y_j)$$

  Where:

  - $\hat{r}_{ui}$ is the predicted rating of user u for item i
  - μ is the global average rating
  - b_u and b_i are the user and item biases, respectively
  - q_i and p_u are the item and user factors, respectively
  - $N(u)$ is the set of items that user u has rated
  - y_j is the implicit feedback factor for item j

  In the context of product recommendation, SVDpp can be particularly effective when there is a wealth of implicit feedback available, such as browsing history or click patterns. By incorporating this implicit feedback into the recommendation model, SVDpp can provide more personalized and accurate recommendations.

  However, like SVD, SVDpp can be computationally expensive, particularly for large datasets with a lot of implicit feedback. Despite this, SVDpp remains a powerful tool for product recommendation, especially in scenarios where implicit feedback is abundant and informative.

- **Non-negative Matrix Factorization (NMF)** is a collaborative filtering algorithm that is based on matrix factorization, similar to SVD. However, the key difference is that NMF only allows non-negative elements in its factor matrices. This constraint makes NMF particularly useful for datasets where the matrix elements represent non-negative quantities, such as counts or physical quantities that cannot be negative.

  The basic idea behind NMF is to approximate a given non-negative matrix X as the product of two lower-rank non-negative matrices W and H. Mathematically , this can be represented as [54]:

  $$X \approx WH$$

  Where:

  - X is the user-item matrix with non-negative elements
  - W is the user-feature matrix
  - H is the item-feature matrix.

  The dimensions of W and H are chosen such that their product approximates X as closely as possible.

  In the context of product recommendation , NMF works by decomposing the user-item rating matrix into two lower-rank matrices: one representing the users and the other representing the items. Each row of the user matrix represents a user's preferences towards different features, and each row of the item matrix represents an item's composition in terms of these features. The predicted rating for a user-item pair is then calculated as the dot product of the corresponding user and item vectors.

  A study by Zhang (2006) [86] demonstrated the effectiveness of NMF in collaborative filtering. They found that NMF outperformed other state-of-the-art recommendation algorithms in terms of recommendation accuracy and computational efficiency. This suggests that NMF can be a powerful tool for product recommendation, especially for large-scale datasets with non-negative ratings.

  However, like other matrix factorization methods, NMF also has its limitations. For instance, it assumes that the missing ratings in the user-item matrix are missing completely at random , which may not always be the case. Furthermore, NMF can be sensitive to the choice of the rank of the factor matrices, which can affect the quality of the approximation.

- **BaselineOnly** algorithm is a fundamental yet effective method in the realm of collaborative filtering algorithms used in recommendation systems. It predicts the baseline estimate for a given user and item, which is a combination of the global mean rating, the user bias, and the item bias [83].

  In the context of product recommendation systems, such as the one we're developing for the Amazon product dataset, the BaselineOnly algorithm can be particularly beneficial. It provides a basic but often reasonably accurate prediction of a user's rating for a product, which can then be used to compare to other algorithms.

  The BaselineOnly algorithm operates as follows:

  1. **Global Mean Rating (μ)**: The algorithm first calculates the global mean rating. This is the average rating across all user-product pairs in the dataset. It serves as the starting point for the baseline estimate.

  2. **User Bias (b_u)**: The user bias is the deviation of the mean rating of a given user from the global mean rating. It represents the tendency of a user to consistently rate products higher or lower than the average.

  3. **Item Bias (b_i)**: The item bias is the deviation of the mean rating of a given product from the global mean rating. It represents the tendency of a product to be consistently rated higher or lower than the average.

  The baseline estimate for a user-product pair is then calculated as the sum of the global mean rating, the user bias, and the item bias:

  $$b_{ui} = \mu + b_u + b_i$$

  Where: - b_{ui} is the baseline estimate for user u and product i, - μ is the global mean rating, - b_u is the user bias, - b_i is the item bias.

  The BaselineOnly algorithm is a good starting point for a product recommendation system. It provides a basic level of personalization by taking into account the individual biases of users and products. However, it does not consider the interactions between users and products, which is where more complex collaborative filtering algorithms come into play.

  Despite its simplicity, the BaselineOnly algorithm can often provide reasonably accurate predictions, especially in scenarios where the dataset is sparse and many user-product pairs do not have associated ratings. It serves as a baseline against which the performance of more complex recommendation algorithms can be compared.

- **CoClustering** is a collaborative filtering algorithm that is based on co-clustering. Co-clustering is a data mining technique that simultaneously clusters both the rows and columns of a matrix. In the context of a product recommendation system, the matrix would represent the user-item interactions, with users as rows, items as products, and the entries as ratings. I t simultaneously groups similar users and similar items together, forming clusters and predicts the rating of a user-item pair by considering the average ratings of the user's cluster and the item's cluster. It essentially assumes that users in the same cluster have similar tastes, and items in the same cluster are similarly rated. Therefore, a user is likely to rate an item similarly to how other users in their cluster have rated items in the item's cluster.

  Now, let's represent the CoClustering algorithm mathematically. Given a user-item rating matrix R, the CoClustering algorithm aims to find two co-clusters C_u and C_i for users and items, respectively. The predicted rating r_{ui} of user u for item i is then calculated as follows:

  $$r_{ui} = \bar{r} + (\bar{r_{C_u}} - \bar{r}) + (\bar{r_{C_i}} - \bar{r})$$

  Where:
  - $\bar{r}$ is the global average rating
  - $\bar{r_{C_u}}$ is the average rating of user cluster C_u
  - $\bar{r_{C_i}}$ is the average rating of item cluster C_i

  A study by Thomas George (2005) [34] demonstrated the effectiveness of CoClustering in collaborative filtering. They proposed a co-clustering model that simultaneously clusters users and items to predict missing ratings. The study found that their model outperformed other state-of-the-art recommendation algorithms in terms of prediction accuracy, especially on sparse and large-scale datasets.

  The advantage of this algorithm is that it can capture more complex user-item interactions than traditional collaborative filtering algorithms. By considering both user-user and item-item similarities , it can make more accurate predictions, especially when the user-item interaction data is sparse.

  However, it's important to note that while CoClustering is a powerful tool for product recommendation, it's not without its limitations. For instance, the quality of the co-clusters can significantly affect the prediction accuracy. Moreover, this algorithm can be computationally expensive, particularly for large datasets. Despite these challenges, it remains a promising method for product recommendation which results will discuss on next sections.

- The **SlopeOne** algorithm computes the deviation between pairs of items to predict user preferences. It is a straightforward and efficient method that works by calculating the average difference in ratings between each pair of items. This average difference , also known as the deviation, is then used to predict a user's rating for an item they have not yet rated. The predicted rating is calculated by adding the deviation to the user's rating for similar items.

  For example, if a user has rated item A and we want to predict their rating for item B, we would add the average difference in ratings between item A and item B to the user's rating for item A. This gives us a predicted rating for item B.

  The mathematical representation of the SlopeOne algorithm is as follows:

  Given a set of items I, for each pair of items $i, j \in I$, the average deviation $dev(i, j)$ is computed as:

$$dev(i, j) = \frac{1}{n_{ij}} \sum_{u \in U} (r_{ui} - r_{uj})$$

  Where:
  - U is the set of users who have rated both items i and j
  - n_{ij} is the number of such users
  - r_{ui} and r_{uj} are the ratings given by user u to items i and j, respectively

  The predicted rating $\hat{r}_{ui}$ for an item i by a user u is then computed as:

$$\hat{r}_{ui} = \frac{1}{n} \sum_{j \in I} (r_{uj} + dev(i, j))$$

  where n is the number of items rated by user u.

  The strength of SlopeOne lies in its simplicity and efficiency. It does not require complex computations or extensive training , making it a practical choice for large-scale recommendation systems. Moreover, it can handle sparse data and high-dimensional spaces, which are common challenges in product recommendation. But it also has its limitations. It assumes that the differences in ratings between items are consistent across all users, which may not always be the case. Furthermore, it does not take into account other factors that may influence a user's rating, such as the user's personal preferences or the context in which the rating was given. Despite these limitations, SlopeOne remains a popular choice for product recommendation due to its simplicity, efficiency, and performance. In a study by S. Dooms (2015) [28], a hybrid recommender system incorporating SlopeOne and other algorithms was found to significantly **outperform** individual algorithms, demonstrating the potential of SlopeOne in a hybrid recommendation context.

Algorithm	Description	Advantages	Disadvantages
SVD	SVD is a matrix factorization technique that decomposes a matrix into three other matrices. It predicts missing ratings by calculating the dot product of the three matrices.	Effective in dealing with sparse data and scalability. Can generalize to new data and provide a rating prediction for previously unseen user-item pairs.	Assumes that the missing ratings in the user-item matrix are missing at random. Can be computationally expensive, particularly for large datasets.
SVDpp	SVDpp is an extension of SVD that takes into account implicit ratings. Implicit ratings refer to user behavior, such as viewing an item. Including these implicit ratings can often increase the accuracy of the prediction.	Takes into account implicit ratings which can often increase the accuracy of the prediction.	More computationally expensive than basic SVD due to the inclusion of implicit ratings.
NMF	NMF is a collaborative filtering algorithm based on Non-negative Matrix Factorization. It is very similar to SVD in that it also performs a matrix factorization, but it only allows non-negative elements in its factor matrices.	Can handle sparsity of the user-item matrix. Provides a parts-based representation due to non-negativity constraints.	Difficulty in interpreting the components. No unique solution exists due to the rotational ambiguity inherent in its formulation.
BaselineOnly	BaselineOnly is an algorithm that predicts the baseline estimate for given user and item. The baseline estimate is a combination of the global mean rating, the user bias, and the item bias.	Simple and fast. Takes into account the user and item biases.	Does not take into account the interaction between user-item pairs beyond the biases.
CoClustering	CoClustering is a collaborative filtering algorithm based on co-clustering. It simultaneously clusters users and items and uses these clusters to make predictions.	Can capture more complex structures in the user-item matrix by clustering not only the items but also the users.	The quality of the recommendation can be highly dependent on the quality of the co-clusters.
SlopeOne	SlopeOne is a straightforward implementation of the SlopeOne collaborative filtering algorithm. It is a model-based approach that computes the deviation between pairs of items to predict user preferences.	Simple and efficient. Can handle new user or new item scenario.	May not be as accurate as other more complex models.

Table 4.4: Comparison of Collaborative Filtering Model-Based Algorithms

4.3.2 Content-Based Filtering Algorithms

Content-Based Filtering (CBF) algorithms primarily leverage specific item attributes or features to recommend items similar to what a user has expressed an interest in before . These algorithms make recommendations by understanding the content of the items and a user's profile. In the context of our study, where we examine the Amazon product dataset for electronics, the metadata of each product in the 'electronics_meta.csv' file is used to generate similar product recommendations.

We used specificly title, description, brand, and category to create a description for each product to present the item content. These descriptors are the foundation for developing a profile or a multi-dimensional vector representation for each item. The process of implementing CBF involves various stages such as **Feature Extraction**, **Dimensionality Reduction**, and **Similarity Metrics**. Each stage plays a vital role in interpreting the content effectively and providing precise similarity-based recommendations.

4.3.2.1 Feature Extraction

Feature Extraction is the initial step in content-based filtering. This step involves processing the raw data to extract useful features that can be used to represent the items in a more structured and meaningful way.

- **Inverse Document Frequency (IDF)** is a measure of how much information the word provides, i.e., if it's common or rare across all documents. It is the logarithmically scaled inverse fraction of the documents that contain the word, obtained by dividing the total number of documents by the number of documents containing the word, and then taking the logarithm of that quotient. Mathematically, it is represented as:

$$IDF(t) = log\left(\frac{N}{df(t)}\right)$$

  - t is the term or word
  - N is the total number of documents in the corpus
  - $df(t)$ is the number of documents in the corpus that contain the term t

  IDF gives a high value for a term that appears rarely across documents and gives a lower value to a term that appears frequently across documents. This helps in reducing the weight of words that appear very frequently, hence, offering little to no information (such as "is", "the", "and", etc.), and increasing the weight of words that appear rarely, thus, offering a lot of information.

- **TF-IDF**, short for **Term Frequency-Inverse Document Frequency,** is a numerical statistic used in information retrieval and text mining. It reflects how important a word is to a document in a collection or corpus[51]. The TF-IDF value increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general.

  The TF-IDF is composed of two terms: the Term Frequency (TF) and the Inverse Document Frequency (IDF).

  The Term Frequency is a measure of the frequency of a term in a document. It is usually the raw count of a term in a document, i.e., the number of times that term t occurs in document d. If we denote the raw count by $f_{t,d}$, then the simplest choice for the TF is $TF = f_{t,d}$.

  Mathematically, the TF-IDF is calculated as [36]:

$$TFIDF(t, d, D) = TF(t, d) \cdot IDF(t, D)$$

  Where:
  - $TF(t, d)$ is the term frequency of term t in document d
  - $IDF(t, D)$ is the inverse document frequency of term t in the set of all documents D, calculated as $IDF(t, D) = log \frac{|D|}{|\{d \in D : t \in d\}|}$
  - $|D|$ is the total number of documents in the set
  - $|\{d \in D : t \in d\}|$ is the number of documents where the term t appears

  In the context of product recommendation systems, TF-IDF can be used to extract features from product descriptions or reviews [8]. These features can then be used to calculate the similarity between different products based on their descriptions or the reviews they received. This can be particularly useful for recommending products that are similar to those a user has shown an interest in or has positively rated.

  TF-IDF's limitation lies in its 'bag of words' approach, ignoring word order and semantics. This can distort text meaning where context is key. Both TF-IDF and IDF are used in the bag of words model, assessing word importance in documents. While IDF focuses on term rarity across all documents, TF-IDF combines Term Frequency and Inverse Document Frequency for a composite score per word. Their use in text mining, information retrieval, and user modeling is extensive , with TF-IDF often being more insightful due to its consideration of term frequency within specific documents as well as overall rarity.

- **Bag-of-Words** (BoW) model is a simple and commonly used method to represent text data in Natural Language Processing (NLP). It's particularly useful in the context of product recommendation systems where the description or reviews of products are used as input data.

  In the context of the Amazon product dataset, particularly the electronics metadata, the BoW model would involve creating a 'vocabulary' of all unique words present in the product descriptions or reviews. Each unique word becomes a feature in the dataset. The frequency of each word in a particular product description or review is then recorded. This results in a matrix where each row represents a product and each column represents a unique word from the 'vocabulary'. The value in each cell is the frequency of the word (column) in the product description or review (row).

  This model is simple to understand and implement , and it can handle large amounts of text data efficiently. However, it does have its limitations. The BoW model does not take into account the order of words in the text, which means it loses any semantic meaning or context. For example, it cannot distinguish between "this product is not good" and "this product is good". Despite these limitations, the BoW model serves as a fundamental model in text-based product recommendation systems. It provides a baseline against which more complex models can be compared and evaluated. By understanding the performance of the BoW model on our Electronics dataset, we can gain insights into the potential improvements that more sophisticated models might offer. Furthermore, the simplicity and computational efficiency of the BoW model make it a practical choice for initial exploratory analysis and prototyping in the development of product recommendation systems.

- **Word2Vec** is a popular word embedding method developed by Google [24]. It represents words in a dense vector space using continuous bag-of-words and skip-gram architectures. The vectors are created in such a way that words with similar meanings have similar vectors. Word2Vec can capture semantic and syntactic similarities and certain relational meanings. Words in the same product category often appear together in product descriptions and hence, will have similar vectors. This enables the recommendation system to capture and suggest products that are contextually similar [29].

  The Word2Vec model is trained on a large corpus of text where each word is predicted based on the words that surround it in a text window. The model generates a dense vector for each word, which is designed to represent its context in the training corpus. For instance, in the context of the Amazon product dataset on electronics metadata, the product descriptions can be used to train a Word2Vec model. The resulting word vectors can capture the semantic relationships between different words in the product descriptions. For example, words like "laptop" and "notebook" might be found to have

similar vectors, reflecting their similar meanings in the context of electronics products [59].

Word2Vec uses two architectures, **Continuous Bag of Words** (CBOW) and **Skip-Gram**, both of which are used to generate the word embeddings.

1. **CBOW**: This method predicts the target word (center word) from source context words (surrounding words). Given a context window of size m, the model is trained on an objective to maximize the average log-probability:

$$\frac{1}{T}\sum_{t=1}^{T}\log p(w_t|w_{t-(m/2)},...,w_{t+(m/2)})$$

where w_t is the target word, and $w_{t-(m/2)},...,w_{t+(m/2)}$ are the context words.

2. **Skip-Gram**: This method predicts source context words from the target word. The objective here is to maximize the average log-probability:

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0}\log p(w_{t+j}|w_t)$$

where w_{t+j} are the context words and w_t is the target word.

In both cases, the probability $p(w_{t+j}|w_t)$ is computed using softmax:

$$p(w_{t+j}|w_t)=\frac{\exp({v'_{w_{t+j}}}^T v_{w_t})}{\sum_{w=1}^{W}\exp({v'_w}^T v_{w_t})}$$

where v_w and v'_w are the "input" and "output" vector representations of w, and W is the number of words in the vocabulary.

These equations represent the mathematical underpinnings of the Word2Vec model . The model is trained by adjusting the word vectors to maximize the likelihood of the context words given the target word (or vice versa, depending on the architecture). The resulting word vectors can then be used to compute similarity scores between different words.

One of the advantages of Word2Vec is that it can capture certain types of semantic relationships between words. For example, the vector difference between "laptop" and "desktop" might be similar to the vector difference between "portable" and "stationary". This property can be useful in a recommendation system, as it allows the system to capture nuanced similarities between different products. However, it's important to note that this algorithm is an unsupervised learning method, which means it does not require any labeled training data. This can be both an advantage and a disadvantage. On one hand, it means that it can be trained on large amounts of unlabeled text data. On the

other hand, it also means that the resulting word vectors might not always align with human intuition about word similarities.

In a study by SeJoon Park, Chul-Ung Kang, and Yung-Cheol Byun [64], they used Word2Vec to enhance recommendation accuracy. They collected user shopping history with personal click preference information of product items as data, representing a document for natural language processing. The sequence of product item clicks was fed into the Word2Vec technology algorithm to obtain the vectors symmetrically representing all of the product items clicked by users. They found that the recommendation accuracy increases when they use multiple dimensions of Word2Vec vectors.

In another study by Kyoung Min Lee [50], they proposed an extrapolative collaborative filtering (ECF) system that improves recommendation performance for small and medium-sized companies that lack data through the extrapolation of data. They developed PP2Vec using Word2Vec by utilizing purchase information only, excluding personal information from payment company data. They found that for the SME merchants with fewer data, the multi-merchant model's performance was higher.

These studies show that Word2Vec can be a powerful tool for product recommendation systems, especially when used in combination with other machine learning techniques.

- **FastText**, developed by Facebook's AI Research lab, is a method that assumes a word to be an n-grams of characters. This approach allows it to understand suffixes and prefixes and also the meaning of shorter words. Unlike Word2Vec, which disregards the morphology of words, FastText uses a more refined approach that considers the internal structure of words. This can be particularly useful for languages or datasets where the order of characters is important for the meaning of words.

  In the context of product recommendation systems, FastText can be particularly useful for capturing the intricacies of product descriptions, especially for products with complex or technical names or descriptions. For instance, in our Amazon product dataset, product descriptions often contain specific model numbers or technical specifications. FastText's ability to understand the internal structure of words can help it capture these details more effectively than other methods [41].

  FastText can be especially useful for recommending products in the **electronics category**, where product names and descriptions often contain a mix of letters and numbers that represent specific product features or specifications. For example, a laptop might be described as having an "Intel Core i7-9750H" processor. It's ability to understand the internal structure of words can help it recognize that "i7-9750H" is a specific type of processor, and that other products with the same processor might be similar.

However, it's important to note that FastText, like Word2Vec, is an unsupervised learning method. This means that it does not require any labeled training data, and the resulting word vectors might not always align with human intuition about word similarities [9]. Despite this, FastText's ability to capture the internal structure of words makes it a valuable tool for product recommendation systems.

Also both these word embedding algorithms, FastText and Word2Vec, learn vector representations for words. However, FastText treats each word as composed of character n-grams. So instead of learning a vector for each word directly, FastText learns vectors for these character n-grams. A word is represented by the sum of the vector representations of its n-grams. For example, for the word "laptop" and n=3, FastText would consider the following n-grams: "lap", "apt", and "top". It would then learn vector representations for each of these n - grams. The vector representation for "laptop" would be the sum of these n - gram vectors.

This approach allows FastText to generate meaningful representations for words not seen during training, which is a significant advantage over other word embedding methods like Word2Vec. This also enabled it to capture more information about the structure of words, which can be particularly useful for languages with complex morphology. However, it's important to note that FastText requires more computational resources than Word2Vec , as it needs to learn vectors for many more n-grams. Despite this, FastText can be a powerful tool for text-based product recommendation systems, as it can capture more nuanced similarities between different products based on their descriptions [81].

Algorithm	Description	Advantages	Disadvantages
TF-IDF	A numerical statistic that reflects how important a word is to a document in a corpus. It calculates the occurrence of a word in a document (Term Frequency) and inverses the document frequency part, which dampens the effect if this word is common in the entire document set.	Understands the context and semantics of the words.	Treats the sentence as a bag of words, and the order of the words in the document or the semantics of the word is not preserved.
Bag of Words	Describes the occurrence of words in a document or a corpus. It creates a vocabulary of all the unique words in the text corpus and represents each word as a feature.	Simple to understand and implement.	Does not consider the order of the words in the document and the semantic meaning is also lost.
Word2Vec	Represents words in a dense vector space using continuous bag-of-words and skip-gram architectures. The vectors are created in such a way that words with similar meanings have similar vectors.	Can capture semantic and syntactic similarities and certain relational meanings.	Does not consider the order of the words in the document and the semantic meaning is also lost.
FastText	Assumes a word to be an n-grams of characters, which allows it to understand suffixes and prefixes and also the meaning of shorter words.	Understands the morphology of words and considers the internal structure of words.	Can be computationally expensive for large vocabularies.

Table 4.5: Comparison of Feature Extraction Algorithms

4.3.2.2 Dimensionality Reduction

Content-based filtering often deals with high-dimensional data. By employing dimensionality reduction, we streamline these data into a simpler, lower-dimensional space, thus boosting the effectiveness of the algorithm and making data handling more manageable. Notably, some algorithms have built-in dimensionality reduction capabilities, eliminating the necessity for this process. However, when applicable, dimensionality reduction is a must since it boosts processing speed, counteracts the 'curse of dimensionality' (a scenario where high-dimensional data compromised performance), resolves overlapping information, and makes data visualisation more possible. This contributes to a more efficient and optimized recommendation system.

- **Principal Component Analysis (PCA)** is a statistical procedure that is used to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. The number of principal components is less than or equal to the number of original variables. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components.

  In the context of our Amazon electronics product recommendation system, PCA can be used to reduce the dimensionality of the feature extraction matrix. This matrix is a high-dimensional matrix where each dimension corresponds to a unique word in the product descriptions. However, not all of these dimensions are equally important. Some words might be very common and not provide much information about the product, while others might be very rare and not appear in enough product descriptions to be useful. PCA can be used to identify the most important dimensions, "principal components" , that capture the most variability in the product descriptions.

  The mathematical formulation of PCA involves a series of steps:

  1. Standardize the data.
  2. Compute the covariance matrix.
  3. Compute the eigenvalues and eigenvectors of the covariance matrix.
  4. Sort the eigenvalues and their corresponding eigenvectors.
  5. Select the first n eigenvectors, where n is the number of dimensions of the new feature subspace.
  6. Transform the original dataset via the selected eigenvectors to obtain a d-dimensional feature subspace.

- **Truncated Singular Value Decomposition (Truncated SVD)** is a variant of the Singular Value Decomposition (SVD) method. It is a linear algebra method that decomposes a matrix into three other matrices. The purpose of this method is to reduce the dimensionality of the dataset, but unlike PCA, this method does not require the input matrix to be centered.

  In the context of product recommendation systems, especially for the Amazon electronics dataset, Truncated SVD can be used to reduce the dimensionality of the TF-IDF matrix. This is particularly useful when dealing with high-dimensional data, as it allows us to retain the most important information (in terms of variance in the data) while reducing the dimensionality.

  The mathematical representation of SVD is as follows:

  Given a matrix M, it can be decomposed into three matrices U, Σ, and V^T such that:

  $$M = U \cdot \Sigma \cdot V^T$$

  Where:

  - U is an $m \times m$ orthogonal matrix
  - Σ is an $m \times n$ diagonal matrix with non-negative real numbers on the diagonal
  - V^T (the transpose of V) is an $n \times n$ orthogonal matrix

  Elements on the diagonal in Σ are known as singular values of M. In Truncated SVD, we select the first k largest singular values and only keep these dominant features, effectively reducing the dimensionality of our data.

  In the context of the Amazon electronics dataset, the Truncated SVD method can be applied to the TF-IDF matrix to reduce its dimensionality while preserving the most important features. This can help in improving the efficiency of the recommendation system , especially when dealing with large datasets .

4.3.2.3 Similarity Measurement

Similarity Measurement is the final phase in content-based filtering. Once the data has been processed and transformed, matrix dimension has been reduced, this step is used to compute the similarity between various items, which is subsequently utilized to generate recommendations. Given that the primary heavy-lifting work of understanding the textual data has been performed during the feature extraction stage, we only require a relatively straightforward

algorithm to conduct the similarity measurement. Since we analyzed a great number of Similarity Measurement algorithms in previous chapter of Literature Review but for Collaborative Filtering, we now briefly describe which algorithms are used on this final phase of content-based filtering similarity recommendation system.

• Cosine Similarity is a metric employed to determine how similar two entities are. Essentially, it computes the cosine of the angle between two vectors. This metric is especially useful when dealing with text data, as the magnitude of the vectors is often less important than their orientation, i.e., their relative distributions of features.

• k-Nearest Neighbors (k-NN) is an instance-based learning technique where the classification of an object is determined by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors. While effective as a similarity measure in collaborative filtering systems, the main drawback is that it can be computationally expensive, as it needs to compute the distance of each query instance to all training samples.

By appropriately combining these steps and algorithms, a content-based filtering system can generate personalized recommendations for users, leveraging the detailed information about the content or features of the items that we've extracted in the previous steps.

4.4 Evaluation Metrics

Evaluating the performance of a recommendation system is a critical aspect of building a successful one. Different metrics allow us to measure various aspects of the system's accuracy and relevance. This section explores several metrics that will be used to evaluate our algorithms throughout this benchmark.

4.4.1 Precision and Recall

Precision and recall are the most common metrics for evaluating the performance of recommendation systems . They form a balance between ensuring the system proposes relevant recommendations (precision) and ensuring it finds most of the items a user might be interested in (recall). These are one of the metrics we utilize to compare our algorithms on Collaborative Filtering methods.

• **Precision@k**: This metric is the proportion of recommended products in the top-k positions that are relevant. It is a measure of the quality of our recommendation algorithm. High Precision@k means the recommendation system is performing well in terms of making correct recommendations.

• **Recall@k**: This metric measures the proportion of relevant products that are found in the top-k recommendations. A high Recall@k score indicates that the algorithm is successfully identifying a large fraction of the relevant products. If the Recall@k is low, it means the system is missing a significant number of items that the user would have liked.

4.4.2 Mean Absolute Error and Root Mean Square Error

As we previously discussed, Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) are widely used metrics in the field of machine learning, including recommendation systems. They play a crucial role in evaluating the performance of recommendation algorithms, particularly in the context of Amazon product dataset reviews that are rated on a scale of 1 to 5. These metrics measure the average magnitude of the errors in a set of predicted product ratings, without considering their direction.

• **MAE** quantifies the average difference between the predicted ratings and the actual ratings. It provides a measure of how wrong the predictions are, on average. For example, an MAE of 0.5 would mean that, on average, the predicted ratings deviate from the actual ratings by 0.5 units on the scale of 1 to 5. A lower MAE indicates better accuracy, as it implies that the predicted ratings are closer to the actual ratings.

• **RMSE** is similar to MAE but gives relatively higher weight to large errors. It measures the square root of the average of the squared differences between the predicted ratings and the actual ratings. RMSE is particularly useful in identifying the impact of larger errors or outliers in the prediction.

These metrics are particularly useful in systems that use collaborative filtering, where we have actual user-item interaction data, and we can make numerical predictions about this data. We can then compare these predictions with the actual values to calculate MAE and RMSE.

4.4.3 Limitation on Content and Popularity Based Recommendations

For content-based filtering methods, traditional numerical evaluation metrics like Precision, Recall, MAE, and RMSE present a challenge due to the nature of our dataset and the recommendation methods employed. The Amazon Products dataset only provide metrics that can be statistically measured with the discussed methods only for reviews subset. Neither electronics subcategory nor the summary of all products on metadata dataset provide any numerical variable that can be utilized to extract a meaningful statistical evaluation. The solution is discussed on next subsection.

In the absence of conventional evaluation metrics, we rely on popularity-based metrics, such as the number of reviews or average ratings to provide predictions of no other user data or product data is available. Although these metrics offer a measure of general product popularity, they do not provide a personalized measure of recommendation relevance to a specific user. Thus, we will use popularity-based metrics as a basic threshold to compare with our more advanced collaborative filtering techniques.

4.4.4 Visual Evaluation of Recommendations

Given the limitations of conventional evaluation metrics in Content-Based filtering techniques due to the Amazon Product Dataset, we resort to visual evaluation of the recommendations. We generate HTML pages that display five random products and the five most similar products for each combination of algorithms. By visually inspecting the quality of the recommendations provided by each algorithm, we aim to assess the effectiveness of our recommendation system. In the following sections, we'll go deeper into each of these aspects and showcase the generated HTML pages with product images and titles, which serve as visual evidence for evaluating the performance of our various recommendation algorithms.

4.4.5 Fit Time and Test Time

These metrics relate to the computational efficiency of the recommendation algorithms:

- **Fit Time**: This is the time taken by the algorithm to learn from the training data. It is an important metric as it indicates the efficiency of the algorithm in terms of resource utilization during the model building phase.
- **Test Time**: This refers to the time taken by the algorithm to generate recommendations for the test data. A shorter test time means the algorithm can quickly generate recommendations, which is particularly important in real-time recommendation scenarios.

These times are crucial for evaluating the practicality of recommendation systems. For online e-commerce stores, a system that takes a long time to generate recommendations may result in a poor user experience, as users typically expect to receive recommendations quickly. Therefore, it is crucial to balance accuracy and computational efficiency when implementing a recommendation system.

The time taken for an algorithm to run not only depends on the efficiency of the algorithm itself but also on the computing power of the system it runs on. Therefore, it's essential to

consider the available hardware and infrastructure when choosing a recommendation algorithm. Powerful recommendation algorithms that provide highly accurate results may require significant computing resources, making them unsuitable for systems with limited computing power. On the other hand, simpler algorithms may run efficiently on modest systems but may not provide the same level of recommendation accuracy.

4.5 Potential Limitations

Any research project, no matter how well-conceived and executed, inevitably encounters certain limitations. Acknowledging these limitations is essential as it allows us to consider their potential impact on our findings and future research directions. Here are some of the potential limitations in our study:

- **Limited Scope of the Dataset** : The study uses an Amazon dataset focused on electronics. Despite its diversity, it's limited in scope. A large portion of users only rate a handful of products and vice versa, creating a sparse matrix which makes generating high-quality recommendations challenging. The preprocessing strategy further narrows down the dataset by retaining only users with more than 50 reviews, affecting the dataset size and potentially the accuracy of the recommendation algorithms. This could limit the generalizability of our findings to other product categories or different e- commerce platforms. Despite the use of methods like collaborative filtering to tackle data sparsity, the constraints of the dataset may still impact the effectiveness of the recommendation system.
- **Data Quality**: Although our dataset, derived from Amazon, is reliable and robust, it isn't completely immune to potential inaccuracies or missing data, which are real characteristics of real-world data. Despite our best efforts in preprocessing to ensure data integrity and consistency, these imperfections may impact the performance of our recommendation algorithms and consequently, the validity of our results. Ideally, to validate the universality and robustness of our recommendation algorithms, it would be advantageous to include numerous datasets from different types of e-commerce businesses and of various data formats. This would provide a broader and more holistic perspective of the performance of the algorithms and allow for a more generalized application of the findings. The use of a single-source dataset, although rich, may limit the comprehensiveness of our insights, making them more specific to the Amazon electronics category, rather than being universally applicable.
- **Scalability Concerns**: While the dataset used in this study is large, with **20 million reviews** and **700,000 products** , the efficiency and effectiveness of the implemented

recommendation techniques might not remain constant when applied to even larger datasets. Recommendation systems are expected to perform in real-time or near real-time in a production environment , often with dynamically expanding product portfolios and user bases. The scalability of our algorithms to support such high demands without compromising the quality of recommendations remains an open question. Therefore, potential concerns about the scalability and performance of our chosen techniques could limit their applicability for businesses with rapidly growing datasets. This aspect is crucial for businesses considering the implementation of these techniques as part of their user experience strategy.

- **Subjective Evaluation**: Our research's significant limitation lies in the subjective nature of evaluating our content-based filtering recommendations. To assess relevance, we visually inspect the recommended products, resulting in human based rather than statistical evaluation. The results are presented in HTML format for easier visual comparison but lack an objective numerical measure of quality, potentially limiting the interpretability and reproducibility of our findings. Despite our efforts for objectivity, the personalized nature of recommendations adds to the complexity.

To sum up, while we believe our research provides valuable insights into various product recommendation algorithms' performance, these potential limitations need to be taken into consideration. These limitations do not invalidate our research but rather frame its findings within a certain context. They also highlight areas where future research could delve deeper and further our understanding of recommendation systems in e-commerce.

Chapter 5

Results & Discussion

In this chapter, we will evaluate and interpret the results obtained from the application of the methodologies of the benchmark discussed in the previous sections. The goal is to critically analyze the output of our recommendation system algorithms and their performance.

We begin with an in-depth overview of the dataset. Here, we will examine its unique characteristics and discuss their potential impacts on the modeling process. This includes a look at the data distribution, unique users and products, rating distribution, data sparsity, and the outcomes of our preprocessing.

We will then focus on presenting the results from our recommendation system models. The chapter will also discuss the impact of different factors on the performance of the algorithms. This could range from the size and quality of the dataset to the tuning of the model parameters and the handling of sparse data. After presenting and analyzing the results, we will discuss our findings.

The discussion will provide interpretations of the results and relate them back to the objectives of our research. It will also reflect on the effectiveness of the different approaches used, highlighting any limitations or challenges encountered during the implementation.

5.1 Dataset Overview

The primary dataset utilized for this analysis, 'Electronics.csv', contains crucial information relevant to our study such as '**userId**', '**productId**', 'rating', and 'timestamp'. This dataset is notably large, consisting of 20,994,353 product reviews.

- **Data Distribution**: The distribution of reviews within this dataset is expansive. The quantity of unique users and products represented within the dataset stands at **756,489** and **9,838,676** respectively. This considerable volume showcases a broad representation of users and product s, establishing the dataset's diversity and thus its potential for generating a robust recommendation system.

```
Opened 'Electronics.csv' with columns:
['userId', 'productId', 'rating', 'timestamp']
Number of reviews    : 20994353
Count of unique Users    : 756489
Count of unique Products : 9838676
```

FIGURE 5.1: Data distribution numbers printed

- **Unique Users and Products**: The number of unique users and products underscores the heterogeneity inherent in the dataset . The high count of unique users (756,489) suggests a wide array of user preferences and behaviors to be captured by our models. Meanwhile, the large number of unique products (9,838,676) indicates a vast array for the recommendation system to draw from, enhancing its capacity to provide diverse and personalized recommendations.

- **Rating Distribution**: Within the dataset, the rating distribution is skewed towards higher ratings:

  5-star ratings make up the majority, with **12,602,917** instances.
  4-star ratings follow at **3,306,379**.
  1-star ratings account for **2,415,650** of the total.
  3-star ratings come next, with **1,529,818** instances.
  Lastly, 2-star ratings are the least common, with **1,139,589** instances.

  The distribution of these ratings is pivotal to our analysis as it not only reflects user intent but also shapes the behavior of our recommendation system. A predominance of higher ratings might make the system more inclined to recommend highly-rated products. However, the presence of moderate and low ratings ensures that the system also recognizes and learns from less satisfactory product experiences. Consequently, the system should be capable of making more nuanced recommendations that take into account the varied range of user experiences captured in these ratings.

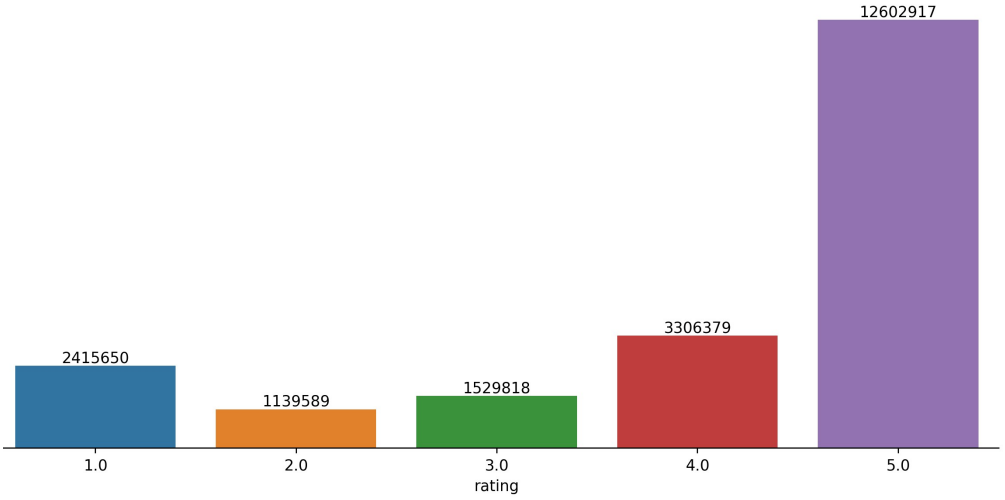

FIGURE 5.2: Plotting the rating Distribution with total review number on each bar

- **Data Sparsity**: Despite the substantial total of 20,994,353 observed ratings in the dataset, when compared to the possible total ratings of 7,442,850,168,564 (calculated by multiplying unique users by unique products), the density of the dataset is extremely sparse, at just 0.00028%. Such sparsity is common in recommendation systems and poses a significant challenge, as it can complicate model training and prediction.

```
Total observed ratings in the dataset  : 20994353
Total ratings possible for the dataset : 7442850168564
Density of the dataset                 : 0.00028%
```

FIGURE 5.3: Terminal with data sparsity data printed

- **Enhancing Data Density** : One of the primary goals of our preprocessing stage was to face the challenges posed by the sparsity of our initial dataset. We tailored our approach to focus on the more engaged users, specifically those who had rated 100 or more products. This decision led to a considerable transformation of our data:

  - The number of **unique users** was reduced to **49,296** from the original 756,489.
  - The final length of the dataset post-preprocessing came down to **107,795 ratings**.
  - The shape of the final ratings matrix was adjusted to **49,296 (users)** by **711 (products)**.

  As a result, the total observed ratings in our dataset post-preprocessing amounted to 107,795. When we compared this to the total possible ratings of 35,049,456 (unique users x unique products), we found that the density of our dataset increased significantly to **0.31**

  The increase in data density and the reduction in size were crucial for our modeling process. By focusing on the more engaged users, we ensured the relevance of the information we were working with, which in turn increased the potential for our recommendation system to deliver meaningful and personalized recommendations.

Importantly, we will later experiment with the **minimum number of rated products per user**, as this plays a pivotal role in our recommendation system's performance, directly influencing the level of data sparsity and the relevance of the user-product interactions we consider. The data was then divided into a training set and a test set in a 70:30

```
Unique users who have rated 100 or more products : 49296

Final length of the dataset : 107795
Shape of final_ratings_matrix: (49296, 711)
Total observed ratings in the dataset  : 107795
Total ratings possible for the dataset : 35049456
Density of the dataset                 : 0.31%
```

FIGURE 5.4: Enhancing Data Density by removing users with less than 100 reviews

ratio, ensuring enough data for the model to learn from while also having an adequate amount to evaluate the model's performance.

In the following sections, we will explore how these characteristics of the dataset influenced the performance of the different recommendation system models implemented in this study.

5.2 Popularity-Based Recommender Results

In this section, we present the findings from our exploration into popularity-based recommendation systems. Such systems recommend products that have been liked, purchased, or positively rated by the most users. Although these systems do not provide personalized suggestions, their simplicity and efficiency make them a preferred choice in numerous places. The greatest strength of these systems is their effectiveness in dealing with the cold start problem, which refers to the challenge of making recommendations for new or anonymous users with no previous interaction history. By recommending popular items that have been highly rated by many users, these systems can offer reliable recommendations to new users visiting the store for the first time and contributes to their growth and revenue.

For our investigation, we used a combined measure of popularity: the sum of ratings, standardized and scaled to match the 1-5 ratings scale, to evaluate the popularity of each product. This method offers a more balanced and robust representation of popularity, considering not only the quantity of ratings (number of ratings) but also their quality (the actual rating values). Using our measure of popularity, products were assigned a score that reflects both the number of users who have rated the product and the quality of these ratings. Our popularity-based recommendation system returned results that will be recognised as a basic threshold to compare with the results of personalized algorithms within the Collaborative Filtering spectrum.

```
RMSE: 1.0580027625612443
MAE: 0.8457907851500148
Precision: 0.8454404346138922
Recall: 0.33660352872107036
FCP: 0.5404762571004841
```

Figure 5.5: Basic Metrics of Popularity Based Filtering algorithm

We evaluated the effectiveness of our popularity-based recommendation system using various metrics, including the Root Mean Squared Error (**RMSE**), Mean Absolute Error (**MAE**), **precision**, **recall**, and Fraction of Concordant Pairs (**FCP**). Our system reported an RMSE of **1.058** and an MAE of **0.846**, indicating a fairly accurate performance in terms of predicting ratings. However, as noted earlier, these metrics can be influenced by the skewed distribution of ratings common in e-commerce platforms, where users tend to rate products mostly when they are extremely satisfied or dissatisfied, leading to a majority of ratings being in the edges of the rating scale.

In terms of precision and recall, our system achieved scores of **0.845** and **0.337**, respectively. These scores suggest that a good proportion of the recommended items were liked by the users (precision) an expected result since it is indeed more likely a fair amount of the users to have rated highly the most popular products since their popularity derives from these numerous reviews, The system wasn't as successful in recommending a diverse set of products liked by the users (recall). Additionally, the FCP score of **0.540** indicates a moderate agreement between the system's ranking of recommendations and the users' preferences.

In conclusion, while popularity-based recommendation systems may not be the most effective at providing personalized recommendations, its crucial to be included in our benchmark since they provide a fundamental baseline upon which more complex and personalized recommendation systems can be built and compared. While popularity-based methods may offer good rating prediction, they might fall short in providing personalized recommendations such as Collaborative Filtering algorithms.

5.3 Collaborative Recommender Results

In this section, we report on the performance of collaborative filtering algorithms. This method uses the collective behavior or preferences of users to make recommendations. We employed a variety of collaborative filtering algorithms, which as described on previous sections can be divided into two categories: model-based and memory-based.

Model-based algorithms generate a model from the ratings data and then use that model to make predictions. In contrast, memory-based algorithms calculate the similarity between users or items based on ratings, and then use these similarity scores to make recommendations. Each

method has its own strengths and weaknesses, and their performance can vary depending on the data and the specific requirements of the recommendation task.

To evaluate the effectiveness of these algorithms, we applied a number of evaluation metrics, including RMSE, MAE, precision, and recall, just like we did with the popularity-based recommendation system. Additionally, we also considered the computation time for each algorithm, splitting it into 'Fit Time' (time taken to train the algorithm) and 'Test Time' (time taken to make predictions).

5.3.1 Memory-Based Algorithms Results

Algorithm	Test RMSE	Test MAE	Fit Time	Test Time	Precision	Recall
SVD	0.940	0.684	0.480	0.050	0.853	0.984
SVDpp	0.943	0.681	0.503	0.109	0.848	0.993
BaselineOnly	0.942	0.689	0.077	0.066	0.854	0.985
CoClustering	1.054	0.743	2.534	0.037	0.868	0.944
SlopeOne	1.105	0.769	0.286	0.097	0.857	0.951
NMF	1.109	0.845	1.697	0.047	0.868	0.904

Table 5.1: Performance of Model-Based Algorithms

According to the results in the table, the SVD, SVDpp, and BaselineOnly algorithms achieved similar performances in terms of RMSE and MAE, with **RMSE** values around **0.94** and **MAE** values around **0.68**. This suggests that these three algorithms were similarly effective at predicting user ratings, with a small average error.

Looking at the Precision and Recall metrics, it is evident that all the model-based algorithms under consideration showed strong performance. All algorithms achieved a **Precision** score greater than **0.84** and a **Recall** score greater than **0.95**, barring the NMF algorithm. Specifically, the SVDpp algorithm achieved an extraordinary **Recall** score of over **0.99**. This score indicates that the SVDpp algorithm was especially adept at recommending items that users would rate highly, successfully capturing nearly all the items of interest to each user. It further implies that the algorithm's recommendations are well-aligned with user preferences, and it effectively uncovers items that users would appreciate but might not have discovered on their own.

The high Precision and Recall values collectively suggest that these algorithms excel not only at predicting items that users would like but also at ensuring that a substantial proportion of these recommended items meet the users' preferences. The blend of high precision and recall denotes a balance between recommending relevant items and covering a broad set of items that users would find interesting. Thus, these algorithms demonstrate both accuracy and breadth in their recommendations, providing a diverse yet personalized set of product suggestions to the users.

It is also important to take note of the Fit Time and Test Time metrics, as these can give us an indication of the computational efficiency of the algorithms. Here, we see a significant variation in the Fit Time, ranging from a minimum of 0.077 for BaselineOnly to a maximum of 2.534 for CoClustering. Test Time was more consistent, with all values being relatively low.

Algorithm	Test RMSE	Test MAE	Fit Time	Test Time	Precision	Recall	Algorithm Type
SVD	0.940	0.684	0.480	0.050	0.853	0.984	Model-Based
SVDpp	0.943	0.681	0.503	0.109	0.848	0.993	Model-Based
BaselineOnly	0.942	0.689	0.077	0.066	0.854	0.985	Model-Based
CoClustering	1.054	0.743	2.534	0.037	0.868	0.944	Model-Based
SlopeOne	1.105	0.769	0.286	0.097	0.857	0.951	Model-Based
NMF	1.109	0.845	1.697	0.047	0.868	0.904	Model-Based

FIGURE 5.6: Performance Comparison of Model-Based algorithms

While it is challenging to rank the algorithms without considering specific application requirements, if we consider a balance of prediction accuracy, computational efficiency, and recommendation relevance, we could propose the following overall ranking:

1. **SVDpp**: This algorithm offers excellent prediction accuracy and the highest recall, indicating it provides the most relevant recommendations to users. While it's not the most efficient in terms of computation time, the high relevance of recommendations makes it a strong choice for many applications.

2. **SVD**: The SVD algorithm achieves the best balance between prediction error and recommendation relevance. It performs slightly better than SVDpp in terms of prediction accuracy, but slightly lower in recall.

3. **BaselineOnly**: This algorithm is the most computationally efficient, which makes it an excellent choice for large-scale applications. Although its prediction accuracy is slightly lower than the top two, it still provides reliable recommendations with decent precision and recall.

4. **CoClustering**: Despite requiring more computational time, this algorithm offers fairly high precision and recall. Its prediction error is higher than the top three, but it still performs reasonably well.

5. **SlopeOne**: This algorithm has a higher prediction error compared to the top four, but still manages to offer decent precision and recall.

6. **NMF**: This algorithm has the highest prediction error among the model-based algorithms, its notably one of the less computationally efficient and also have by far the worst result in Recall metric. Its performance across all three aspects considered makes it less preferred than the other model-based algorithms.

It is important to note that the optimal algorithm may vary based on specific use-cases and constraints. These rankings are generalized and are based on the results from the dataset and variables used in this analysis. Different results may be generated on experimentation with different **Sparsity** on next sections.

5.3.2 Memory-Based Algorithms Results

Algorithm	Test RMSE	Test MAE	Fit Time	Test Time	Precision	Recall
KNNWithMeans	1.097	0.769	16.228	0.622	0.858	0.947
KNNBaseline	0.981	0.687	16.952	0.635	0.855	0.972
KNNBasic	1.042	0.751	15.956	0.744	0.848	0.984
KNNWithZScore	1.107	0.775	16.942	0.595	0.856	0.949

Table 5.2: Performance of Memory-Based Algorithms

On memory-based algorithms, we used the user-based version instead of the item-based for the main testing. We will discuss the item-based on experiementation section. When it comes to prediction accuracy, measured by RMSE and MAE, it is apparent that **KNNBaseline** algorithm outperforms other memory-based algorithms, with the lowest RMSE of **0.981** and MAE of **0.687**. It is followed by **KNNBasic** with a slightly higher RMSE and MAE. Inspecting the Precision and Recall metrics, the **KNNBasic** algorithm achieved the highest recall of **0.984**, indicating that it was successful in suggesting a large portion of relevant items to the users.

1. **KNNBaseline**: This algorithm offers the best RMSE and MAE and one of highest precision and recall, indicating it provides the most relevant recommendations to users. While it's not the most efficient in terms of computation time, the high relevance of recommendations makes it a strong choice among the tested memory-based algorithm.

2. **KNNBasic**: This algorithm achieves the best recall but also the worst precision but the second best prediction error. It performs the best on Fit time but worst on Test time. The basic variant of k-NN algorithm performs decently among the other variants and its simple implementation shall be recognised as a bonus.

3. **KNNWithMeans**: Although this algorithm's prediction accuracy is slightly lower than the top two, it still provides reliable recommendations with decent precision and recall.

4. **KNNWithZScore**: Despite requiring more computational time on training, this algorithm offers the lowest test time. Its prediction error is higher than the top three, but it still performs reasonably well.

The optimal choice depends on the specific constraints and requirements of the application. This evaluation suggests that the different variations of the k-NN a lgorithm that are widely

Algorithm	Test RMSE	Test MAE	Fit Time	Test Time	Precision	Recall	Algorithm Type
KNNWithMeans	1.090	0.764	21.377	1.149	0.850	0.952	Memory-Based
KNNBaseline	0.982	0.687	18.888	0.712	0.854	0.976	Memory-Based
KNNBasic	1.042	0.749	18.601	0.839	0.848	0.985	Memory-Based
KNNWithZScore	1.103	0.773	24.033	0.760	0.860	0.950	Memory-Based

FIGURE 5.7: Anoter Run of the same dataset with different results of k-NN based algorithms

used for memory-based collaborative filtering recommendation, do not imply any serious difference in the prediction accuracy, computational efficiency and prediction relevancy.

5.3.3 Comparison of Results

When comparing the results of memory-based algorithms and model-based algorithms, one significant difference that stands out is the computational efficiency. In the given evaluation, the model-based algorithms, generally have much lower fit times compared to the memory-based algorithms.

The fit time measures how long it takes to train the algorithms on the given dataset. In the case of memory-based algorithms, the fit times are considerably higher, ranging aroung 16 seconds each. On the other hand, the model-based algorithms have fit times that are significantly lower, ranging from 0.077 to 2.534s. BaselineOnly algorithm stands out as the most computationally efficient with a fit time of 0.077. This suggests that model-based algorithms require less time to train on the dataset compared to memory-based algorithms.

The test time, which measures how long it takes to make predictions on the test dataset, is more consistent across both types of algorithms. In the given evaluation, the test times for all algorithms, both memory-based and model-based, are relatively low. The difference in computational efficiency between memory-based and model-based algorithms is an essential consideration in real-world applications. Memory-based algorithms, particularly the k-NN variants, often involve calculating similarity measures between users or items , which can be computationally intensive and time-consuming. As the dataset grows larger, the computational cost of memory-based algorithms tends to increase significantly.

On the other hand, model-based algorithms, such as SVD, SVDpp, and BaselineOnly , typically involve matrix factorization or optimization techniques, which can be more computationally efficient. These algorithms derive latent factors or biases from the dataset during the training phase, allowing for faster predictions during the test phase. Therefore, if computational efficiency is a critical factor, model-based algorithms generally undeniably offer a favorable option. They can provide reasonably accurate predictions while requiring less time for training and making predictions. Memory-based algorithms, while effective in certain scenarios, may become less practical as the dataset size grows.

5.4 Content-Based Recommender Results

In this section, we will present the performance evaluation of our content-based algorithms. The primary goal of these algorithms is to identify similar products based on Amazon's electronics product metadata. The performance evaluation is subjective due to the lack of an objective statistical metric in the metadata.

As we discussed in the previous sections, we split the recommendation system process into **three** phases: **Feature Extraction**, **Dimensionality Reduction**, and **Similarity Measurement** . Since our evaluation is inherently subjective, our strategy will be to choose the best-performing algorithm at each phase, moving step by step through the process. We will initially benchmark the various feature extraction algorithms using the default dimensionality reduction and similarity measurement algorithms. Once we've identified the best performing feature extraction algorithm, we will then proceed to test different dimensionality reduction and similarity algorithms, observing the impact they have on the results.

To visually demonstrate the effectiveness of our recommendation system, we will generate HTML pages displaying images and titles of five similar products recommended for each of five selected products, totaling four distinct pages of results corresponding to each algorithm combination. The subjective relevance of the recommended products is used as a key metric to assess the effectiveness of our algorithms. For instance, if a specific laptop model is the given product, relevant recommendations would include other laptop models from the same brand, laptops with similar specifications from other brands, or accessories compatible with the given laptop.

5.4.1 Feature Extraction Results

The goal is to generate Simiar Products recommendations based on the combined feature 'description' derived from the 'title', 'brand', and 'main_cat' columns. This combined feature can help capture the essence of each product and enable more accurate recommendations. After numerous benchmarks, the most suitable ones are collected and displayed below in order to discuss its findings. Due to the unorthodox format of the images provided by the Amazon Product metadata dataset, many benchmarks are dismissed due to some products unavailability of photos and some others, photos that do not match well the product description.

Benchmark 1

FIGURE 5.8: Similar Products with Bag-of-Words as Feature Extraction algorithm

FIGURE 5.9: Similar Products with TF-IDF as Feature Extraction algorithm

Recommendations using Word2Vec

FIGURE 5.10: Similar Products with Word2Vec as Feature Extraction algorithm

Recommendations using FastText

FIGURE 5.11: Similar Products with FastText as Feature Extraction algorithm

As the Figures suggest, this specific benchmark displays quite similar products to the randomly picked Retevis walkie-talkie. Even from algorithms we expect less similar results, Bag-of-Words, very similar products are being selected as the most similar. The two word embeddings algorithms Word2Vec and FastText found almost the same products but with different ranking. W ord2Vec is the only algorithm that found a product item similar enough that has the same photo, and after examination is the same product but in other version (6pack instead of 4pack).

Arguably difficult to rank the performance of these algorithms in this specific benchmark, the findings suggest that all the algorithms rank almost perfect, but I will give a small edge to **Word2Vec**.

Benchmark 2

FIGURE 5.12: Benchmark 2. Bag-of-Words as Feature Extraction algorithm

FIGURE 5.13: Benchmark 2. TF-IDF as Feature Extraction algorithm

FIGURE 5.14: Benchmark 2. Word2Vec as Feature Extraction algorithm

FIGURE 5.15: Benchmark 2. FastText as Feature Extraction algorithm

In the second benchmark, the figures indicate a significant variation in the similar products recommended by each algorithm for the AeroCool Window Edition Case. The Bag-of-Words algorithm's performance was notably subpar, with only one out of five suggestions being arguably similar to the product in question. The remaining suggestions included two tablet cases, a cooling hardware for a desktop processor, and a desktop accessory, which are not closely related to the original product.

On the other hand, the TF-IDF algorithm recommended five desktop cases, similar to the original product we tested. However, these products were not from the same brand, "AeroCool", which is the primary point of difference when compared to the recommendations from the word embeddings algorithms.

The Word2Vec and FastText algorithms demonstrated superior performance by accurately recommending five desktop cases, four of which were from the same brand as the original product. The only difference between the recommendations from these two algorithms was the ranking of the products. Given the subjective nature of determining better results, both Word2Vec and FastText are considered to have performed exceptionally well.

In summary, while the **TF-IDF** algorithm performed reasonably well, the **Bag-of-Words** algorithm's recommendations were largely irrelevant. The **Word2Vec** and **FastText** algorithms, on the other hand, provided the most accurate recommendations, demonstrating their effectiveness in this benchmark.

Benchmark 3

FIGURE 5.16: Benchmark 3. Bag-of-Words as Feature Extraction algorithm

FIGURE 5.17: Benchmark 3. TF-IDF as Feature Extraction algorithm

FIGURE 5.18: Benchmark 3. Word2Vec as Feature Extraction algorithm

FIGURE 5.19: Benchmark 3. FastTest as Feature Extraction algorithm

In a similar manner to the previous benchmark, a distinct variance is observed in the quality of product similarity suggested by the different algorithms. A product with a less popular brand, ”Tuff-Luv Leather Book Style case and Keyboard case for Transformer Prime TF201 Tablet - Black”, was randomly selected for testing. This product presents a challenging scenario due to the commonality of words in its title such as ”Leather”, ”Book”, ”Case”, ”Keyboard”, ”Tablet”, ”Black”. These words could potentially **mislead** a simplistic algorithm into recommending irrelevant products.

As anticipated, the **Bag-of-Words** algorithm, with its straightforward approach, was misled by these commonly used words. However, it managed to suggest three out of five tablet cases, albeit quite different from the original product. One recommendation was an oddly specific desktop case for a particular motherboard, and another was a backpack suitable for tablets and laptops. None of the recommended products were from the same brand as the original product. This benchmark further reinforces the notion that the Bag-of-Words algorithm may not be reliable for a real-world recommendation system. Its primary utility lies in serving as a comparative baseline for more sophisticated algorithms.

Next, the **TF-IDF** algorithm managed to recommend five out of five tablet cases. However, the style and subcategory of the tablet cases varied significantly, and only one of the recommendations was from the same brand. While this is an improvement over the Bag-of-Words algorithm, it still falls short of a professional solution. Such recommendations could potentially confuse users in a real-world e-commerce setting, potentially damaging the store’s reputation rather than boosting revenue.

Finally, both **Word2Vec** and **FastText** algorithms demonstrated a marked improvement over TF-IDF. They recommended highly similar Tuff-Luv book-style tablet cases, but in a different ranking. It’s important to note that the dataset provides multiple images for each product, and during data pre-processing, only the first image was retained. This could result in different

images for very similar products from the same category, brand, and version. However, this should not detract from the performance of these algorithms. A close examination of each product title reveals a high degree of similarity. Without a doubt, a well-structured e-commerce platform employing these methods could provide exceptionally accurate product similarity recommendations on specific product pages.

Benchmark 4

FIGURE 5.20: Benchmark 4. Bag-of-Words as Feature Extraction algorithm

FIGURE 5.21: Benchmark 4. TF-IDF as Feature Extraction algorithm

FIGURE 5.22: Benchmark 4. Word2Vec as Feature Extraction algorithm

FIGURE 5.23: Benchmark 4. FastText as Feature Extraction algorithm

In this benchmark, it is evident that all the tested algorithms perform exceptionally well, offering recommendations highly similar to the original product. This could be attributed to the highly specific title of the product, "EVGA GeForce GTX 780 Ti Superclocked, 3GB, 3072MB,GDDR5 384bit, Dual-Link DVI-I, DVI-D, HDMI,DP, SLI Ready Graphics Card (03G-P4-2883-KR)". The title contains unique and specific keywords that are not commonly used, reducing the likelihood of misleading even the simplest algorithms. As anticipated, the Word2Vec and FastText algorithms delivered strong results. However, what's particularly noteworthy in this benchmark is the performance of the **Bag-of-Words** algorithm. This demonstrates that even basic algorithms can yield accurate recommendations when the product descriptions contain distinct and specific keywords.

Comparison and Ranking

It's important to note that the Word2Vec and FastText algorithms, based on word embeddings, inherently handle dimensionality reduction, while the Bag-of-Words and TF-IDF algorithms required the application of TruncatedSVD for this purpose.

1. **Word2Vec**: This algorithm, based on word embeddings, consistently delivered superior results across all benchmarks. Word2Vec was particularly effective in recommending similar products when the product descriptions contained unique and specific key words. It was able to accurately recommend products from the same category and often from the same brand. The time taken for applying Word2Vec was varying near 110 seconds. Despite long computation time, the quality of the recommendations was consistently high. Word2Vec's ability to capture the semantic meaning of words and their context within the product descriptions proved to be a significant advantage and can be recognised as the best algorithm amost the tested ones.

```
Applying Word2Vec...
Time taken: 113.99 seconds
```

Figure 5.24: Time taken for Word2Vec algorithm

2. **FastText**: This word embeddings-based algorithm, also performed exceptionally well, similar to Word2Vec. It recommended highly relevant products and matched the original product's brand in most cases. The time taken for applying FastText was near double the time of Word2Vec algorithm which is why its ranked second. . Despite taking the longest time among the four algorithms, FastText's performance was impressive and arguably same with Word2Vec. Its unique approach of considering subword information allowed it to effectively handle the complexities of the product descriptions and provide accurate recommendations.

```
Applying FastText...
Time taken: 232.31 seconds
```

Figure 5.25: Time taken for FastText algorithm

3. **TF-IDF**: The TF-IDF algorithm performed reasonably well, often recommending products from the same category as the original. However, its performance was somewhat inconsistent, particularly when the product descriptions contained common words. The time taken for applying TF-IDF Vectorizer was about 35 seconds, dimensionality reduction was varying near 40 seconds. While TF-IDF was faster than the word embeddings algorithms, its recommendations were not as accurate. TF-IDF's approach of considering the frequency of words in the product descriptions and their inverse frequency in the corpus helped it to provide relevant recommendations, but it struggled to match the

performance of the word embeddings algorithms. Undoubtedly, it scored well on most benchmarks and was far superior to simple Bag-of-Words algorith.

```
Applying TF-IDF Vectorizer...
Time taken: 35.7 seconds
Applying LSI for dimensionality reduction...
Time taken: 39.85 seconds
```

FIGURE 5.26: Time taken for TF-IDF and TruncatedSVD combination

4. **Bag-of-Words**: The Bag-of-Words algorithm had by far the worst performance. In some benchmarks, it was able to recommend relevant products, particularly when the product descriptions contained unique and specific keywords. However, it struggled when the descriptions contained common words, often recommending irrelevant products. The time taken for applying Bag of Words was close but longer than IF-IDF algorithm. Despite being the simplest algorithm, Bag-of-Words managed to provide some relevant recommendations in some well suited scenarios. But its approach of treating each word independently and failing to capture the semantic meaning of words made it susceptible to being misled by common words thus providing irrelevant results in most cases. It is undoubtedly the worst algorithm of the 4, should be avoided in real production systems and should only account for a basic threshold in comparing better algorithms in research and development.

```
Applying Bag of Words Vectorizer...
Time taken: 39.76 seconds
Applying LSI for dimensionality reduction...
Time taken: 40.54 seconds
```

FIGURE 5.27: Time taken for Bag-of-Words and TruncatedSVD combination

In conclusion, the Word2Vec and FastText algorithms outperformed the TF-IDF and Bag-of-Words algorithms in these benchmarks. However, it's important to note that the time taken for computation is not a critical factor in this context. Unlike real-time user-based recommendation systems, the computation for similar product recommendations is typically done once and updated periodically (daily or weekly), asynchronously. Therefore, even if an algorithm takes longer, it does not impact the user experience as the results are pre-computed. The primary concern would be the computational resources required, as an algorithm that takes significantly longer could potentially slow down the server or cause it to crash if not managed properly.

5.4.2 Dimensionality Reduction and Similarity Measurement

The next steps are essential for dealing with high-dimensional data and for identifying similar items based on their features. In our benchmarks we did not find any difference in the tested

algorithms, indicated that the majority of the work in our approach is done in the feature extraction phase, which we analyzed earlier.

Dimensionality Reduction

We used Truncated Singular Value Decomposition (TruncatedSVD) for dimensionality reduction. TruncatedSVD is a matrix factorization technique that reduces the dimension of a high-dimensional dataset. Unlike Principal Component Analysis (PCA), TruncatedSVD does not require the input data to be centered and can work with sparse matrices, making it more suitable for text data represented as TF-IDF vectors or Bag-of-Words vectors.

It's important to note that we experimented with several dimensionality reduction techniques, including PCA. However, we found no difference at all in the performance of the recommendation system across these techniques. Given this, we opted for TruncatedSVD due to its computational efficiency in contrast to PCA that took a prohibitive amount of time.

Similarity Measurement

In our benchmark, we used cosine similarity as our similarity metric. This simple metric is particularly effective for high-dimensional, sparse data, as it considers the direction of the vectors rather than their magnitude. We experimented with several similarity metrics, including Euclidean distance and Jaccard similarity. However, we found no significant difference in the performance of the recommendation system across these metrics. Given this, we opted for cosine similarity due to its computational efficiency and its effectiveness with high-dimensional, sparse data.

In conclusion, our choice of TruncatedSVD for dimensionality reduction and cosine similarity for similarity measurement was based on their computational efficiency and suitability for our data. Despite experimenting with several techniques, we found no difference in the performance of the recommendation system across these techniques. Therefore, we opted for the fastest algorithms to ensure the efficiency of our system.

Chapter 6

Experimentation

Experimentation and parameter tuning are integral parts of any algorithmic research. While our benchmark study is firmly grounded and the results align with the findings from the literature review, it's crucial to explore different parameters to ensure the robustness of our algorithms across various datasets. By adjusting these parameters, we can better understand the strengths and weaknesses of our algorithms and optimize them for different scenarios.

One of the key parameters we experimented with was the sparsity of the reviews dataset. In the default state, we retained only the reviews from users who rated 100 or more products. This was done to create a denser dataset of user-item interactions, which typically leads to more accurate recommendations. However, this approach may not be suitable for all datasets, particularly those with fewer user-item interactions. Therefore, we conducted additional experiments with varying degrees of dataset sparsity.

6.1 Experimenting with Dataset Sparsity

In these experiments, we adjusted the minimum number of products that a user must have rated to be included in the dataset. By increasing this threshold, we created a denser dataset with fewer users and more interactions per user. Conversely, by decreasing this threshold, we created a sparser dataset with more users and fewer interactions per user.

These experiments allowed us to observe how the performance of our algorithms changed with the density of the dataset. For instance, we could assess whether certain algorithms were more robust to sparsity or if they benefited significantly from a denser dataset. This is particularly important in the context of recommendation systems, as the density of user-item interactions can vary greatly across different platforms and domains.

The results of our benchmark study provide valuable insights into the performance of the tested algorithms under varying levels of dataset sparsity. Here, we discuss the results obtained for different sparsity levels: 50, 200, 300, and 400 ratings per user.

6.1.1 Results for 200 Ratings per User

Algorithm	Test RMSE	Test MAE	Fit Time	Test Time	Precision	Recall
SVD	0.884	0.660	0.145	0.014	0.858	0.994
SVDpp	0.891	0.665	0.108	0.033	0.860	0.999
BaselineOnly	0.887	0.666	0.022	0.026	0.862	0.995
CoClustering	0.960	0.708	0.870	0.010	0.870	0.970
SlopeOne	0.990	0.718	0.091	0.018	0.879	0.969
NMF	1.017	0.789	0.541	0.013	0.880	0.940
KNNWithMeans	0.974	0.714	1.491	0.101	0.871	0.971
KNNBaseline	0.898	0.647	1.867	0.153	0.866	0.983
KNNBasic	0.948	0.705	1.431	0.114	0.862	0.987
KNNWithZScore	0.978	0.714	1.931	0.143	0.868	0.970

Table 6.1: Performance of Model-Based and Memory-Based Algorithms for 200+ Ratings

When the threshold was reduced to 200 ratings per user, the performance of the algorithms varied due to the new dataset of 15256 reviews. All algorithms and their metrics had a very analogical increase in quality of results, due to the new sparser dataset. This is an expected behavior and no algorithm had an offset of the previous 100 ratings-user dataset analogy of results and ranking. The most impactful change was on Training time of the k-NN memory based algorithms that decreased from 16 seconds to around 1 second. Even though this is an expected change, it has to be mentioned in order to highlight the important influence the size of the dataset has upon the training time of the memory based algorithms, which increases in exponential scale.

Algorithm	Test RMSE	Test MAE	Fit Time	Test Time	Precision	Recall	Algorithm Type
SVD	0.884	0.660	0.145	0.014	0.858	0.994	Model-Based
SVDpp	0.891	0.665	0.108	0.033	0.860	0.999	Model-Based
BaselineOnly	0.887	0.666	0.022	0.026	0.862	0.995	Model-Based
CoClustering	0.960	0.708	0.870	0.010	0.870	0.970	Model-Based
SlopeOne	0.990	0.718	0.091	0.018	0.879	0.969	Model-Based
NMF	1.017	0.789	0.541	0.013	0.880	0.940	Model-Based

Figure 6.1: Model-Based algorithms with 200+ rating-users

Algorithm	Test RMSE	Test MAE	Fit Time	Test Time	Precision	Recall	Algorithm Type
KNNWithMeans	0.974	0.714	1.491	0.101	0.871	0.971	Memory-Based
KNNBaseline	0.898	0.647	1.867	0.153	0.866	0.983	Memory-Based
KNNBasic	0.948	0.705	1.431	0.114	0.862	0.987	Memory-Based
KNNWithZScore	0.978	0.714	1.931	0.143	0.868	0.970	Memory-Based

Figure 6.2: Memory-Based algorithms with 200+ rating-users

6.1.2 Results for 300 Ratings per User

Algorithm	Test RMSE	Test MAE	Fit Time	Test Time	Precision	Recall
SVD	0.819	0.625	0.060	0.006	0.886	0.999
SVDpp	0.819	0.623	0.040	0.009	0.869	0.999
BaselineOnly	0.818	0.625	0.009	0.005	0.891	1.000
CoClustering	0.871	0.655	0.355	0.004	0.889	0.985
SlopeOne	0.874	0.651	0.041	0.008	0.893	0.975
NMF	0.913	0.715	0.291	0.008	0.885	0.952
KNNWithMeans	0.871	0.652	0.392	0.045	0.883	0.986
KNNBaseline	0.827	0.607	0.465	0.065	0.888	0.995
KNNBasic	0.863	0.650	0.378	0.050	0.896	0.994
KNNWithZScore	0.871	0.655	0.534	0.045	0.886	0.984

Table 6.2: Performance of Model-Based and Memory-Based Algorithms for 300+ Ratings

For the dataset with a minimum of 300 ratings per user, the SVD, SVDpp, and BaselineOnly algorithms performed remarkably well, with RMSE values below 0.82 and almost precision and recall scores. The new dataset of 6878 ratings proved to be an important factor on the quality of the results. The KNN-based algorithms also performed well, with KNNBaseline showing the best results among them in the same manner as the previous benchmarks. The most notable mention on this benchmark is that the both RMSE-MAE metrics and Precision-Recall had a great increase and are the best compared to all our benchmarks even the 400 ratings per user which will discuss next. Here is another run:

Algorithm	Test RMSE	Test MAE	Fit Time	Test Time	Precision	Recall	Algorithm Type
SVD	0.831	0.630	0.145	0.020	0.881	1.000	Model-Based
SVDpp	0.835	0.630	0.134	0.022	0.889	0.999	Model-Based
BaselineOnly	0.834	0.631	0.022	0.009	0.879	0.999	Model-Based
CoClustering	0.879	0.662	0.781	0.010	0.888	0.984	Model-Based
SlopeOne	0.896	0.669	0.062	0.009	0.890	0.984	Model-Based
NMF	0.928	0.724	0.409	0.009	0.890	0.969	Model-Based

Figure 6.3: Model-Based algorithms with 300+ rating-users

Algorithm	Test RMSE	Test MAE	Fit Time	Test Time	Precision	Recall	Algorithm Type
KNNWithMeans	0.904	0.672	0.699	0.076	0.891	0.986	Memory-Based
KNNBaseline	0.864	0.626	0.514	0.067	0.882	0.995	Memory-Based
KNNBasic	0.894	0.666	0.530	0.080	0.877	0.993	Memory-Based
KNNWithZScore	0.909	0.674	0.653	0.049	0.893	0.979	Memory-Based

Figure 6.4: Memory-Based algorithms with 300+ rating-users

6.1.3 Results for 400 Ratings per User

When the threshold was increased to 400 ratings per user, the dataset became denser to 2696 rows. Comparing to 300 ratings-user dataset all metrics were worse. The kings of our benchmark SVDpp and SVD performance was decrease and matched the BaselineOnly algorithm

Algorithm	Test RMSE	Test MAE	Fit Time	Test Time	Precision	Recall
SVD	0.923	0.691	0.023	0.002	0.874	1.000
SVDpp	0.931	0.702	0.017	0.002	0.871	1.000
BaselineOnly	0.931	0.701	0.003	0.002	0.862	1.000
CoClustering	0.989	0.737	0.145	0.012	0.874	0.985
SlopeOne	0.969	0.718	0.016	0.003	0.857	0.994
NMF	0.989	0.760	0.088	0.002	0.862	0.989
KNNWithMeans	0.923	0.684	0.080	0.011	0.863	0.998
KNNBaseline	0.900	0.652	0.063	0.010	0.875	0.998
KNNBasic	0.928	0.692	0.068	0.010	0.891	0.998
KNNWithZScore	0.925	0.689	0.132	0.011	0.888	0.992

Table 6.3: Performance of Model-Based and Memory-Based Algorithms for 400+ Ratings

which was scoring less quality results in all our previous benchmarks. A notable highlight is the fact that this is the first time the NMF algorithm did not have a noticeable difference in the end of the list of best algorithms, meaning that all 3 worst algorithms scored almost the same. On the memory based algorithms however, the best recommendations were provided by the KNN Baseline, as in all our other benchmarks but the highlight is that all other 3 algorithms scored almost the same.

Algorithm	Test RMSE	Test MAE	Fit Time	Test Time	Precision	Recall	Algorithm Type
SVD	0.923	0.691	0.023	0.002	0.874	1.000	Model-Based
SVDpp	0.931	0.702	0.017	0.002	0.871	1.000	Model-Based
BaselineOnly	0.931	0.701	0.003	0.002	0.862	1.000	Model-Based
CoClustering	0.989	0.737	0.145	0.012	0.874	0.985	Model-Based
SlopeOne	0.969	0.718	0.016	0.003	0.857	0.994	Model-Based
NMF	0.989	0.760	0.088	0.002	0.862	0.989	Model-Based

Figure 6.5: Model-Based algorithms with 400+ rating-users

Algorithm	Test RMSE	Test MAE	Fit Time	Test Time	Precision	Recall	Algorithm Type
KNNWithMeans	0.923	0.684	0.080	0.011	0.863	0.998	Memory-Based
KNNBaseline	0.900	0.652	0.063	0.010	0.875	0.998	Memory-Based
KNNBasic	0.928	0.692	0.068	0.010	0.891	0.998	Memory-Based
KNNWithZScore	0.925	0.689	0.132	0.011	0.888	0.992	Memory-Based

Figure 6.6: Memory-Based algorithms with 400+ rating-users

6.1.4 Decreasing Density

In our experimentation, we also explored the performance of our algorithms on a less dense dataset, specifically one where users have rated 50 or more products. This resulted in a significantly larger dataset, with 197,030 rows of data.

The model-based algorithms were able to handle this larger dataset but as expected the performance of these algorithms decreased compared to the original dataset (100+ ratings per user). The RMSE values increased, indicating a higher error rate, and the precision and recall scores

decreased, suggesting less accurate and comprehensive recommendations. The ranking of the best to worst algorithm we recognised on the previous benchmarks, still apply here without any apparent spin-off.

Algorithm	Test RMSE	Test MAE	Fit Time	Test Time	Precision	Recall	Algorithm Type
SVD	0.989	0.705	2.096	0.266	0.845	0.972	Model-Based
SVDpp	0.989	0.700	3.774	0.965	0.842	0.973	Model-Based
BaselineOnly	0.988	0.708	0.433	0.201	0.840	0.974	Model-Based
CoClustering	1.098	0.752	7.929	0.142	0.851	0.921	Model-Based
SlopeOne	1.177	0.803	0.889	0.442	0.848	0.927	Model-Based
NMF	1.173	0.881	5.939	0.265	0.866	0.862	Model-Based

Figure 6.7: Model-Based algorithms with 50+ rating-users

When we attempted to apply the memory-based algorithms to this larger dataset, we encountered a memory error. This error is due to the large size of the similarity matrix that these algorithms need to compute, which exceeded the available RAM on our machine. This error

```
numpy.core._exceptions._ArrayMemoryError: Unable to allocate 16.8 GiB for an array with shape (67064, 67064) and data type int32
```

Figure 6.8: Error on Memory-Based algorithms with 50+ rating-users

highlights an important consideration when working with collaborative filtering algorithms, especially memory-based ones. The size of the dataset and the available computational resources can significantly impact the feasibility and performance of these algorithms. In particular, memory-based algorithms, which require the computation of a similarity matrix, can be particularly resource-intensive and may not be suitable for larger datasets or machines with limited RAM.

This also means that the default dataset (100+ ratings per user) that we used in our study may not be suitable for all machines. Depending on the available computational resources, it may be necessary to adjust the dataset density (i.e., the minimum number of ratings per user) to ensure that the algorithms can run successfully.

6.1.5 Summary of Findings

In summary, our experimentation with varying the density of the user-item interactions dataset yielded some insightful findings. We observed that the performance of the algorithms varied depending on the sparsity of the data.

For the model-based algorithms, we found that as the density of the dataset increased (from users who rated 100+ products to those who rated 400+ products), the performance of the algorithms generally improved. This was evident in the decrease in both the Test RMSE and Test MAE values, indicating more accurate predictions. The precision and recall values also increased, suggesting that the algorithms were able to recommend more relevant items to the users.

However, when the density was further increased (to users who rated 50+ products), the performance of the model-based algorithms declined. This was likely due to the increased complexity and noise in the data, which made it more challenging for the algorithms to identify patterns and make accurate predictions. As for the memory-based algorithms, we were unable to run them on the 50+ ratings per user dataset due to memory limitations. This highlights the importance of considering computational resources when choosing an algorithm for a recommendation system.

Overall, our findings suggest that while increasing the density of the dataset can improve the performance of recommendation algorithms, there is a threshold beyond which further increases in density may lead to decreased performance. Furthermore, the choice of algorithm should take into consideration not only its quality performance but also its computational requirements.

6.2 Varying Neighborhood Size in KNN Algorithms

As part of our experimental design in applying collaborative filtering for recommendation systems, we studied the impact of varying the size of the neighborhood (k-value) in K-nearest neighbors (KNN) algorithms, specifically using the '**KNNBaseline**' algorithm which scored the best on our default benchmark. Our exploration included a wide range of k-values, from a minimum of 5 up to a maximum of 150. Due to the nature of the k-NN algorithms, the dataset size was decreased to 300+ ratings-users we benchmarked before, in order to generate results into a fair amount of time. Running the algorithms for various k, with the default of 100+ ratings-users would take an extravagant amount of time, with no purpose, since our previous findings suggest that the algorithms ranking is maintained throughout the different experiments we tried.

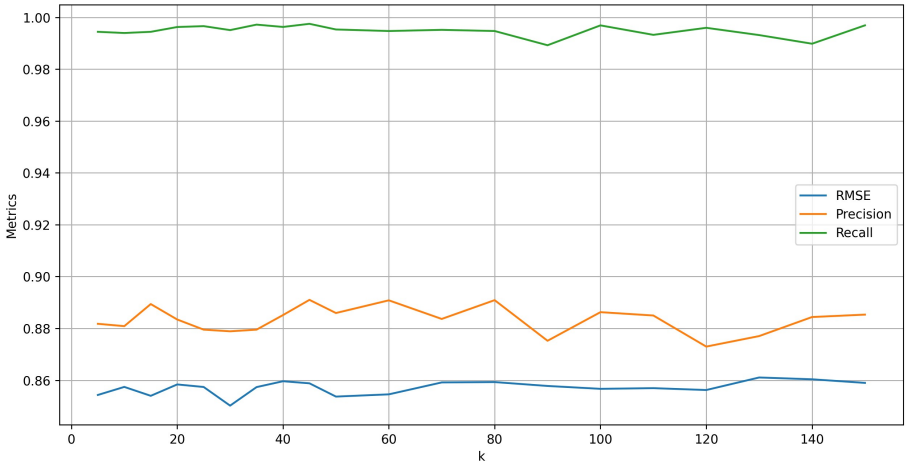

Figure 6.9: Metrics varying with different k in k-NN algorithms

The results from our experiment suggest some noteworthy observations. We found that, despite the variations in the k-value, the quality of recommendations doesn't deteriorate significantly. This is an important finding because it demonstrates the robustness of the 'KNNBaseline' algorithm, and by extension, KNN algorithms in general.

The performance measures we looked at included precision, recall, and RMSE. Each of these metrics provide a different perspective on the quality of the recommendations being made. Interestingly, the observed stability in these metrics with varying k-values is not a common phenomenon across all types of datasets or algorithms. It can be attributed to several factors:

- **Nature of the Dataset**: The specific characteristics of the dataset, such as sparsity or inherent noise, can impact the behavior of the recommendation algorithm. The results indicate that our dataset may be well-suited for KNN-based recommendation approaches.
- **Algorithm Design**: KNN algorithms are inherently flexible, with the ability to adapt to the local structure of the data. The stability of metrics could be a result of this inherent flexibility of KNN algorithms.
- **Quality of Neighborhood**: Even with a large number of neighbors, if the quality (similarity) of neighbors is high, the recommendation quality remains high. This suggests that our dataset may have strong user-user similarity structures.

While the stability of the recommendation quality with varying neighborhood sizes is encouraging, it's important to note that a larger k-value implies more computational resources. Hence, there is a trade-off to consider between computation cost and the quality of recommendations. The results suggest that we can choose a relatively smaller k-value without significantly compromising the recommendation quality.

6.3 Item-Based vs User-Based Collaborative Filtering

In this section, we analyze the performance of the item-based collaborative filtering approach. As we previously discussed user-based collaborative filtering, the results obtained here will be contrasted against those from the user-based analysis to draw further insights.

Algorithm	Test RMSE	Test MAE	Fit Time	Test Time	Precision	Recall
KNNWithMeans	1.013	0.729	0.045	0.083	0.845	0.975
KNNBaseline	0.977	0.686	0.114	0.080	0.864	0.967
KNNBasic	1.050	0.756	0.025	0.084	0.844	0.977
KNNWithZScore	1.013	0.725	0.075	0.085	0.851	0.974

Table 6.4: Performance of Item-Based Collaborative Filtering

The item-based collaborative filtering results indicate a dramatic reduction in Fit Time even for the same 100 review-users subset we tested by default. This reduction in computational time might be particularly significant when dealing with even larger datasets, contributing to the scalability of the item-based approach.

In terms of accuracy, item-based collaborative filtering also performed comparably better to the user-based method. The Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) values are relatively close between the two methods, with the KNNBaseline algorithm showing the lowest RMSE of all the algorithms, and the best Recall and Precision in the same manner as our default user-based benchmark.

In summary, the item based approach seems to have a clear edge on user-based in terms of these statistical metrics we tested, especially considering its computational efficiency. This findings align with the theoretical knowledge that item-based collaborative filtering can provide more stable recommendations, as item properties often change less frequently than user preferences. However, the best choice of algorithm may still depend on the specific dataset and application at hand. Further testing and validation may be necessary when implementing these algorithms in a real-world system since the quantifiable metrics utilized in benchmarks dont always match with the reality.

Chapter 7

Conclusions & Future Work

7.1 Summary of Findings

Throughout this study, we conducted a series of experiments and benchmarks to evaluate numerous Collaborative Filtering (CF) and Content-Based Filtering (CBF) algorithms, employing the harnessing the Amazon product dataset for reviews and a subset of this dataset for product metadata. The primary objective of these algorithms was to provide effective product recommendations based on different contexts.

For CF, the goal was to suggest products that similar users have shown a preference for, utilizing review data as the primary source of user behavior. In a real-world e-commerce context, CF algorithms are particularly useful for revisiting users, where their past behavior, such as reviews, clicks, visit duration, bookmarks, and search history, can be harnessed to provide personalized product recommendations. Most datasets only provided user reviews , and not other kinds of user metadata, in the same context, the one we used in this benchmark which is one of the most popular and widely used dataset for testing product recommendation systems, only provide user reviews. However, only using these to evaluate CF algorithms was very effective, since in order to utilize even more user metadata would require more subjective pre-processing to make them suitable, vectorized and quantified that wouldnt have a noticeable difference in the comparison and ranking of the algorithms performance.

In the evaluation of collaborative filtering systems, the SVD, SVDpp, and BaselineOnly algorithms achieved the best performances in terms of RMSE and MAE, suggesting that these three algorithms were similarly and remarkably effective at predicting user ratings, with a small average error. The difference in computational efficiency between memory-based and model-based algorithms is an essential consideration in real-world applications. As the dataset grows larger, the computational cost of memory-based algorithms tends to increase significantly. In

contrast, **SVDpp** algorithm maintaining computational efficiency, achieved an extraordinary Recall score in all our benchmarks and experiments, thus, it can be crowned as the best algorithm tested in this field.

On the other hand, CBF algorithms aimed to recommend items similar to a specific product. This approach is often employed on individual product pages of e-commerce sites, recommending products similar to the one the user is currently viewing. The intention here is to present potential better matches that the user might prefer. Given the absence of a quantifiable metric for similarity in our dataset, we evaluated the algorithms by displaying similar products across numerous tests. Although in the previous sections we only analyzed four benchmarks in detail, these were representative of the results obtained from over a hundred runs, providing a comprehensive overview of the performance of the CBF algorithms.

Word2Vec and FastText, the two word embedding algorithms, outperformed both Bag-of-Words and TF-IDF in almost all benchmarks. They were able to capture the semantic meaning of words and phrases in the product descriptions, leading to highly accurate and relevant recommendations. These algorithms were also less affected by the specific wording of product descriptions, making them more versatile and robust. In terms of computational efficiency, Bag-of-Words and TF-IDF required additional steps for dimensionality reduction, which added to their overall processing time. Word2Vec and FastText, on the other hand, inherently handle high-dimensional data, making them more efficient in terms of computation time, **Word2Vec**, managed to score the same as FastText but also run less than half the time, so it will be crowned the king among all the tested algorithms in the content based filtering approach.

7.2 Limitations of the Study

This study, while comprehensive in its scope, is not without its limitations.

Dataset Constraints: The Amazon Reviews dataset used in this study, although renowned and widely used, has its own set of limitations. Firstly, the dataset only includes review data and product metadata, which may not fully capture the complexity of user behavior and product characteristics in a real world e-commerce environment. For instance, user behavior such as browsing history, click patterns, and time spent on each product page, which can provide valuable insights for recommendation systems, are not included in the dataset.

Moreover, the format and nature of the data in the dataset are quite distinct and different for the reviews and product metadata. This discrepancy posed a challenge in combining Collaborative Filtering and Content-Based Filtering into a hybrid system. In a real-world e-commerce scenario, user behavior and product characteristics data would be more intertwined and dynamic, which would allow for a more effective implementation of hybrid recommendation systems.

Given the distinct and static nature of the dataset, it was is more appropriate to evaluate Collaborative Filtering and Content-Based Filtering separately. This approach allowed us to focus on the strengths and weaknesses of each method in isolation, providing clear insights into their performance and characteristics.

It's important to note that this limitation does not diminish the value of the findings from this study. Rather, it highlights the need for further research using more complex and dynamic datasets that closely mimic real-world e-commerce environments. The purpose of a research paper and its associated benchmarks is not to deliver a fully operational, highly sophisticated real-world production system that can be seamlessly integrated into any e-commerce platform. By its very nature, such a goal is impossible within the spectrum of a single study. Instead, the primary objective of our research is to provide valuable insights into the selection of suitable algorithms and to serve as a guide for the development of real-world recommendation systems. Our findings offer a roadmap for researchers and engineers in the field, helping them navigate the complexities of building effective, efficient, and scalable recommendation systems for e-commerce applications.

Computational Resources: The computational resources available for this study also posed a limitation. The memory-based collaborative filtering algorithms, in particular, require substantial computational resources for training, especially with large datasets. This limitation was evident in our experiments, where memory-based algorithms exhibited longer fit times compared to model-based algorithms. While we were able to conduct the experiments and obtain meaningful results, the computational constraints may limit the scalability of these algorithms in a real world scenario with much larger datasets and more complex user-item interactions.

Range of tested Algorithms: While we tested a dozen of collaborative filtering and content-based filtering algorithms, the study did not cover all possible recommendation algorithms. Other types of recommendation systems, such as hybrid methods that combine collaborative filtering and content-based filtering, or deep learning recommendation algorithms, were not included in this study. These algorithms could potentially offer improved performance and address some of the limitations of the algorithms tested in this study. Also The evaluation metrics used in this study, while standard and widely accepted, may not fully capture the quality of a recommendation system from a user's perspective. Metrics like precision and recall do not consider the ranking of the recommended items, which is crucial in a real-world recommendation scenario where the top few recommendations are more likely to be clicked by the user. Furthermore, these metrics do not account for aspects like diversity, novelty, and serendipity of recommendations, which are recognized as important factors in user satisfaction with recommendation systems.

Despite these limitations, this study provides valuable insights in the context of e-commerce product recommendations. The findings from this study can serve as a useful starting point

for further research and development in this area, with a focus on addressing these limitations and improving the effectiveness and efficiency of recommendation systems.

7.3 Future Work

The findings of this study spark up several thoughts for future research and development in the field of recommendation systems. While we have made significant steps towards understanding the performance of various algorithms on the Amazon Reviews dataset, there are numerous opportunities to build upon this work and further enhance the effectiveness of recommendation systems in production e-commerce applications.

- **Expanding the Dataset**: One of the key areas for future work involves expanding the dataset used for benchmarking. In this study, we used the Amazon Reviews dataset, which, while comprehensive, does not fully capture the complexity of user behavior and product characteristics in a real-world e-commerce environment. A more representative dataset, perhaps a clone of a real life e-commerce store with a vast array of products, extensive user interactions, and numerous reviews, could provide a more robust platform for testing and refining recommendation algorithms.
- **Hybrid and Deep Learning Approaches** :Another promising direction for future research is the exploration of hybrid recommendation systems, which combine the strengths of both Collaborative Filtering and Content-Based Filtering. These systems could potentially offer more accurate and personalized product recommendations. Furthermore, the integration of deep learning techniques into recommendation systems could also be a fruitful area of investigation. Deep learning has shown great promise in various fields of artificial intelligence and could potentially enhance the predictive accuracy of recommendation systems.
- **Context-Specific Recommendations**: In a real-world e-commerce environment, different types of pages may benefit from different types of recommendations. For instance, the home page for new visitors could display popularity-based recommendations to address the cold start problem, while the home page for returning users could display recommendations based on collaborative filtering. Similarly, category or search pages could display hybrid recommendations based on the similarity of products that the user has searched for, combined with what other similar users have shown interest in. Single product pages could display similar products based on content-based filtering.

  This context-specific approach to recommendations could potentially enhance the user experience and increase conversion rates by providing more relevant and personalized

product suggestions. However, implementing such a system would require a more complex and dynamic dataset, as well as sophisticated algorithms capable of handling different types of data and producing different types of recommendations.

Appendix A

Mathematical Notations

This appendix introduces the general mathematical notation used throughout this thesis.

1. **Vectors and Matrices**
   - A matrix is denoted by a bold uppercase letter, e.g., **M**
   - The element in the i^{th} row and j^{th} column of a matrix **M** is denoted as M_{ij}.

2. **Sets**
   - A set is denoted by an uppercase letter, e.g., U.
   - The number of elements in a set U is denoted as $|U|$.

3. **Summation**
   - The sum of a sequence of numbers a_i from $i = 1$ to n is denoted as $\sum_{i=1}^{n} a_i$.

4. **Averages**
   - The average (or mean) of a sequence of numbers a_i from $i = 1$ to n is denoted as $\bar{a} = \frac{1}{n} \sum_{i=1}^{n} a_i$.

5. **Ratings**
   - The rating given by user u to item i is denoted as r_{ui}.
   - The predicted rating of user u for item i is denoted as $\hat{r}_{ui}$.

6. **Bias Terms**
   - The user bias for user u is denoted as b_u.
   - The item bias for item i is denoted as b_i.
   - The global average rating is denoted as μ.

7. **Deviation**
   - The average deviation between items i and j is denoted as $dev(i, j)$.

8. **Standard Deviation**
   - The standard deviation of the ratings of user u is denoted as σ_u.
   - The standard deviation of the ratings of item i is denoted as σ_i.

This notation is used consistently throughout the thesis to represent various mathematical and statistical concepts.

Bibliography

[1] G. Adomavicius and A. Tuzhilin. 2005. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering* 17, 6 (2005), 734–749. https://doi.org/10.1109/TKDE.2005.99

[2] Gediminas Adomavicius and Jingjing Zhang. 2012. Impact of data characteristics on recommender systems performance. *ACM Transactions on Management Information Systems (TMIS)* 3 (04 2012). https://doi.org/10.1145/2151163.2151166

[3] Yassine Afoudi, Mohamed Lazaar, and Mohammed Al Achhab. 2021. Hybrid recommendation system combined content-based filtering and collaborative prediction using artificial neural network. *Simulation Modelling Practice and Theory* 113 (2021), 102375. https://doi.org/10.1016/j.simpat.2021.102375

[4] Atallah Al-Shatnawi, Bader Al-Fawwaz, and Wafa Alsharafat. 2015. Recognizing the Importance of Brand Awareness on E-commerce Sales while Shopping on Internet: Empirical Analysis of European Countries. *International Journal of Interactive Mobile Technologies (iJIM)* 9 (01 2015), 15. https://doi.org/10.3991/ijim.v9i1.4111

[5] Hemant Ambulgekar, Manjiri Pathak, and Manesh Kokare. 2019. *A Survey on Collaborative Filtering: Tasks, Approaches and Applications: eHaCON 2018, Kolkata, India.* 289–300. https://doi.org/10.1007/978-981-13-1544-224

[6] Panteli Antiopi and Basilis Boutsinas. 2023. Addressing the Cold-Start Problem in Recommender Systems Based on Frequent Patterns. *Algorithms* 16 (03 2023), 182. https://doi.org/10.3390/a16040182

[7] Tessy Badriyah, Sefryan Azvy, Wiratmoko Yuwono, and Iwan Syarif. 2018. Recommendation system for property search using content based filtering method. In *2018 International Conference on Information and Communications Technology (ICOIACT).* 25–29. https://doi.org/10.1109/ICOIACT.2018.8350801

[8] Tessy Badriyah, Erry Tri Wijayanto, Iwan Syarif, and Prima Kristalina. 2017. A hybrid recommendation system for E-commerce based on product description and user profile.

In *2017 Seventh International Conference on Innovative Computing Technology (INTECH)*. 95–100. https://doi.org/10.1109/INTECH.2017.8102435

[9] Ahmet Tuğrul Bayrak. 2022. A Session-Based Recommendation Approach with Word Embeddings. In *2022 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*. 1–5. https://doi.org/10.1109/INISTA55318.2022.9894161

[10] Robert M. Bell and Yehuda Koren. 2007. Lessons from the Netflix Prize Challenge. *SIGKDD Explor. Newsl.* 9, 2 (dec 2007), 75–79. https://doi.org/10.1145/1345448.1345465

[11] James Bennett, Charles Elkan, Bing Liu, Padhraic Smyth, and Domonkos Tikk. 2007. KDD Cup and workshop 2007. *ACM SIGKDD Explorations Newsletter* 9 (12 2007), 51–52. https://doi.org/10.1145/1345448.1345459

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

[13] Ehsan Bojnordi and Parham Moradi. 2012. A novel collaborative filtering model based on combination of correlation method with matrix completion technique. In *The 16th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP 2012)*. 191–194. https://doi.org/10.1109/AISP.2012.6313742

[14] Poonam B.Thorat, R. Goudar, and Sunita Barve. 2015. Survey on Collaborative Filtering, Content-based Filtering and Hybrid Recommendation System. *International Journal of Computer Applications* 110 (01 2015), 31–36. https://doi.org/10.5120/19308-0760

[15] Jürgen Buder and Christina Schwind. 2012. Learning with Personalized Recommender Systems: A Psychological View. *Comput. Hum. Behav.* 28, 1 (jan 2012), 207–216. https://doi.org/10.1016/j.chb.2011.09.002

[16] Robin Burke. 2002. Hybrid Recommender Systems: Survey and Experiments. *User Modeling and User-Adapted Interaction* 12 (11 2002). https://doi.org/10.1023/A:1021240730564

[17] Licínio Carvalho, Fátima Rodrigues, and Pedro Oliveira. 2020. *A Hybrid Recommendation Algorithm to Address the Cold Start Problem.* 260–271. https://doi.org/10.1007/978-3-030-14347-325

[18] Pablo Castells, Neil Hurley, and Saul Vargas. 2015. *Novelty and Diversity in Recommender Systems.* 881–918. https://doi.org/10.1007/978-1-4899-7637-626

[19] Abhijnyan Chandra, Arif Ahmed, Sandeep Kumar, Prateek Chand, Malaya Dutta Borah, and Zakir Hussain. 2022. Content-Based Recommender System for Similar Products in E-Commerce. In *Edge Analytics*, Ripon Patgiri, Sivaji Bandyopadhyay, Malaya Dutta Borah, and Valentina Emilia Balas (Eds.). Springer Singapore, Singapore, 617–628.

[20] Olivier Chapelle, Eren Manavoglu, and Romer Rosales. 2015. Simple and Scalable Response Prediction for Display Advertising. *ACM Trans. Intell. Syst. Technol.* 5, 4, Article 61 (dec 2015), 34 pages. https://doi.org/10.1145/2532128

[21] Chao Chen, Dongsheng Li, Yingying Zhao, Qin Lv, and Li Shang. 2015. WEMAREC: Accurate and Scalable Recommendation through Weighted and Ensemble Matrix Approximation. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Santiago, Chile) *(SIGIR '15)*. Association for Computing Machinery, New York, NY, USA, 303–312. https://doi.org/10.1145/2766462.2767718

[22] Yuanyi Chen. 2021. Research on Resource Personalization Recommendation Algorithm and Model Based on Computer Deep Learning. *Journal of Physics: Conference Series* 1915, 2 (may 2021), 022024. https://doi.org/10.1088/1742-6596/1915/2/022024

[23] Keunho Choi, Donghee Yoo, Gunwoo Kim, and Yongmoo Suh. 2012. A hybrid online-product recommendation system: Combining implicit rating-based collaborative filtering and sequential pattern analysis. *Electronic Commerce Research and Applications* 11, 4 (2012), 309–317. https://doi.org/10.1016/j.elerap.2012.02.004

[24] KENNETH WARD CHURCH. 2017. Word2Vec. *Natural Language Engineering* 23, 1 (2017), 155–162. https://doi.org/10.1017/S1351324916000334

[25] Colum Cronin. 2011. Doing your literature review: traditional and systematic techniques. *Evaluation Research in Education* 24 (09 2011), 219–221. https://doi.org/10.1080/09500790.2011.581509

[26] Mukund Deshpande and George Karypis. 2004. Item-based top- N recommendation algorithms. *ACM Transactions on Information Systems - TOIS* 22 (01 2004), 143–177. https://doi.org/10.1145/963770.963776

[27] Hong-Quan Do, Tuan-Hiep Le, and Byeongnam Yoon. 2020. Dynamic Weighted Hybrid Recommender Systems. In *2020 22nd International Conference on Advanced Communication Technology (ICACT)*. 644–650. https://doi.org/10.23919/ICACT48636.2020.9061465

[28] Simon Dooms, Toon De Pessemier, and Luc Martens. 2013. Offline optimization for user-specific hybrid recommender systems. *Multimedia Tools and Applications* 74 (2013), 3053–3076.

[29] Ramazan Esmeli, Mohamed Bader-El-Den, and Hassana Abdullahi. 2020. Using Word2Vec Recommendation for Improved Purchase Prediction. In *2020 International Joint Conference on Neural Networks (IJCNN)*. 1–8. https://doi.org/10.1109/IJCNN48605.2020.9206871

[30] Nadia Fadhil and Soukaena Hashem. 2019. Collaborative Filtering Recommendation Model Based on k-means Clustering. *Al-Nahrain Journal of Science* 22 (03 2019), 74–79. https://doi.org/10.22401/ANJS.22.1.10

[31] Fethi Fkih. 2021. Similarity Measures for Collaborative Filtering-based Recommender Systems: Review and Experimental Comparison. *Journal of King Saud University - Computer and Information Sciences* 34 (09 2021). https://doi.org/10.1016/j.jksuci.2021.09.014

[32] Sandra Gadanho and Nicolas Lhuillier. 2007. Addressing uncertainty in implicit preferences. 97–104. https://doi.org/10.1145/1297231.1297248

[33] Rahul Gaikwad, Sandeep Udmale, and Vijay Sambhe. 2018. *E-commerce Recommendation System Using Improved Probabilistic Model*. 277–284. https://doi.org/10.1007/978-981-10-3920-128

[34] Thomas George and Srujana Merugu. 2005. A scalable collaborative filtering framework based on co-clustering. 4 pp.–. https://doi.org/10.1109/ICDM.2005.14

[35] Shraddha Gupta. 2020. A Literature Review on Recommendation Systems.

[36] Emma Haddi, Xiaohui Liu, and Yong Shi. 2013. The Role of Text Pre-processing in Sentiment Analysis. *Procedia Computer Science* 17 (12 2013), 26–32. https://doi.org/10.1016/j.procs.2013.05.005

[37] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* 5, 4, Article 19 (dec 2015), 19 pages. https://doi.org/10.1145/2827872

[38] Hartatik Hartatik, Bayu Permana Sejati, Aulia Nur Fitrianto, and Wiwi Widayani. 2021. A Comparison Study of Model Based Collaborative Filtering Using Alternating Least Square and Singular Value Decomposition. In *2021 3rd International Conference on Electronics Representation and Algorithm (ICERA)*. 185–190. https://doi.org/10.1109/ICERA53111.2021.9538709

[39] Jon Herlocker, Joseph Konstan, Loren Terveen, John C.s Lui, and T. Riedl. 2004. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems* 22 (01 2004), 5–53. https://doi.org/10.1145/963770.963772

[40] Thomas Hofmann. 2004. Latent Semantic Models for Collaborative Filtering. *ACM Trans. Inf. Syst.* 22, 1 (jan 2004), 89–115. https://doi.org/10.1145/963770.963774

[41] Shigang Hu, Akshi Kumar, Fadi Al-Turjman, Shivam Gupta, Simran Seth, and Shubham. 2020. Reviewer Credibility and Sentiment Analysis Based User Profile Modelling for Online Product Recommendation. *IEEE Access* 8 (2020), 26172–26189. https://doi.org/10.1109/ACCESS.2020.2971087

[42] Folasade Isinkaye, Yetunde Folajimi, and Bolanle Ojokoh. 2015. Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal* 16 (08 2015). https://doi.org/10.1016/j.eij.2015.06.005

[43] Celestine Iwendi, Ebuka Ibeke, Harshini Eggoni, Sreerajavenkatareddy Velagala, and Gautam Srivastava. 2021. Pointer-Based Item-to-Item Collaborative Filtering Recommendation System Using a Machine Learning Model. *International Journal of Information Technology Decision Making* 21 (09 2021), 1–22. https://doi.org/10.1142/S0219622021500619

[44] Mozhgan Karimi, Dietmar Jannach, and Michael Jugovac. 2018. News recommender systems – Survey and roads ahead. *Information Processing Management* 54, 6 (2018), 1203–1227. https://doi.org/10.1016/j.ipm.2018.04.008

[45] Harleen Kaur and Gourav Bathla. 2019. Techniques of Recommender System. *International Journal of Innovative Technology and Exploring Engineering* (2019).

[46] Ketki Kinkar. 2021. Product Recommendation System: A Systematic Literature Review. *International Journal for Research in Applied Science and Engineering Technology* 9 (07 2021), 3330–3339. https://doi.org/10.22214/ijraset.2021.37024

[47] Yehuda Koren. 2008. Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Las Vegas, Nevada, USA) *(KDD '08)*. Association for Computing Machinery, New York, NY, USA, 426–434. https://doi.org/10.1145/1401890.1401944

[48] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (2009), 30–37. https://doi.org/10.1109/MC.2009.263

[49] Joonseok Lee, Seungyeon Kim, Guy Lebanon, and Yoram Singer. 2013. Local Low-Rank Matrix Approximation. In *Proceedings of the 30th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 28)*, Sanjoy Dasgupta and David McAllester (Eds.). PMLR, Atlanta, Georgia, USA, 82–90. https://proceedings.mlr.press/v28/lee13.html

[50] Kyoung Lee, Yu Hwangbo, Baek Jeong, Ji Yoo, and Kyung Park. 2021. Extrapolative Collaborative Filtering Recommendation System with Word2Vec for Purchased Product for SMEs. *Sustainability* 13 (06 2021), 7156. https://doi.org/10.3390/su13137156

[51] David D. Lewis, Robert E. Schapire, James P. Callan, and Ron Papka. 1996. Training Algorithms for Linear Text Classifiers. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Zurich, Switzerland) *(SIGIR '96)*. Association for Computing Machinery, New York, NY, USA, 298–306. https://doi.org/10.1145/243199.243277

[52] Dongsheng Li, Chao Chen, Qin Lv, Junchi Yan, Li Shang, and Stephen Chu. 2016. Low-Rank Matrix Approximation with Stability. In *Proceedings of The 33rd International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 48)*, Maria Florina Balcan and Kilian Q. Weinberger (Eds.). PMLR, New York, New York, USA, 295–303. https://proceedings.mlr.press/v48/lib16.html

[53] Yu Li, Liu Lu, and Li Xuefeng. 2005. A hybrid collaborative filtering method for multiple-interests and multiple-content recommendation in E-Commerce. *Expert Systems with Applications* 28, 1 (2005), 67–77. https://doi.org/10.1016/j.eswa.2004.08.013

[54] Wolfram Alpha LLC. 2023. Matrix Factorization. https://www.wolframalpha.com/input?i=Matrix+Factorization Accessed: 2023-05-16.

[55] Wolfram Alpha LLC. 2023. Singular value decomposition. https://www.wolframalpha.com/input?i=singular+value+decomposition Accessed: 2023-05-16.

[56] Xin Luo, Mengchu Zhou, Yunni Xia, and Qingsheng Zhu. 2014. An Efficient Non-Negative Matrix-Factorization-Based Approach to Collaborative Filtering for Recommender Systems. *IEEE Transactions on Industrial Informatics* 10, 2 (2014), 1273–1284. https://doi.org/10.1109/TII.2014.2308433

[57] Farhin Mansur, Vibha Patel, and Mihir Patel. 2017. A review on recommender systems. In *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*. 1–6. https://doi.org/10.1109/ICIIECS.2017.8276182

[58] McKinsey & Company. Year. *How retailers can keep up with consumers*. https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers

[59] Priyanka Meel and Agniva Goswami. 2019. Inverse Document Frequency-Weighted Word2Vec Model to Recommend Apparels. In *2019 6th International Conference on Signal Processing and Integrated Networks (SPIN)*. 1–7. https://doi.org/10.1109/SPIN.2019.8711722

[60] Najdt Mustafa, Ashraf Osman Ibrahim, Ali Ahmed, and Afnizanfaizal Abdullah. 2017. Collaborative filtering: Techniques and applications. *2017 International Conference on Communication, Control, Computing and Electronics Engineering (ICCCCEE)* (2017), 1–6.

[61] Cataldo Musto. 2010. Enhanced vector space models for content-based recommender systems. 361–364. https://doi.org/10.1145/1864708.1864791

[62] Vito Ostuni, Sergio Oramas, Tommaso Di Noia, Xavier Serra, and Eugenio Di Sciascio. 2015. A Semantic Hybrid Approach for Sound Recommendation. https://doi.org/10.1145/2740908.2742775

[63] Junwei Pan, Jian Xu, Alfonso Ruiz, Wenliang Zhao, Shengjun Pan, Yu Sun, and Quan Lu. 2018. Field-weighted Factorization Machines for Click-Through Rate Prediction in Display Advertising. *WWW '18: Proceedings of the 2018 World Wide Web Conference*, 1349–1357. https://doi.org/10.1145/3178876.3186040

[64] Se-Joon Park, Chul-Ung Kang, and Yungcheol Byun. 2021. Extreme Gradient Boosting for Recommendation System by Transforming Product Classification into Regression Based on Multi-Dimensional Word2Vec. *Symmetry* 13 (04 2021), 758. https://doi.org/10.3390/sym13050758

[65] Axita Patel. 2018. "Survey and Evolution Study Focusing Comparative Analysis and Future Research Direction in the Field of Recommendation System Specific to Collaborative Filtering Approach". https://doi.org/10.1007/978-981-13-1742-2

[66] S GOPAL PATRO, Brojo Mishra, Sanjaya Panda, Raghvendra Kumar, Hoang Long, David Taniar, and Ishaani Priyadarshini. 2020. A Hybrid Action-Related K-Nearest Neighbour (HAR-KNN) Approach for Recommendation Systems. *IEEE Access* 8 (05 2020), 1–1. https://doi.org/10.1109/ACCESS.2020.2994056

[67] H. Polat and Wenliang Du. 2003. Privacy-preserving collaborative filtering using randomized perturbation techniques. In *Third IEEE International Conference on Data Mining*. 625–628. https://doi.org/10.1109/ICDM.2003.1250993

[68] Huseyin Polat and Wenliang Du. 2005. SVD-Based Collaborative Filtering with Privacy. In *Proceedings of the 2005 ACM Symposium on Applied Computing* (Santa Fe, New Mexico) *(SAC '05)*. Association for Computing Machinery, New York, NY, USA, 791–795. https://doi.org/10.1145/1066677.1066860

[69] Ivens Portugal, Paulo Alencar, and Donald Cowan. 2015. The Use of Machine Learning Algorithms in Recommender Systems: A Systematic Review. *Expert Systems with Applications* 97 (11 2015). https://doi.org/10.1016/j.eswa.2017.12.020

[70] Abinash Pujahari and Dilip Singh Sisodia. 2020. Pair-wise Preference Relation based Probabilistic Matrix Factorization for Collaborative Filtering in Recommender System. *Knowledge-Based Systems* 196 (2020), 105798. https://doi.org/10.1016/j.knosys.2020.105798

[71] Md. Mijanur Rahman, Ismat Shama, Siamur Rahman, and Rahmatullah Nabil. 2022. HYBRID RECOMMENDATION SYSTEM TO SOLVE COLD START PROBLEM. *Journal of Theoretical and Applied Information Technology* 100 (06 2022).

[72] Vahid Roudposhti, Mehrbakhsh Nilashi, Abbas Mardani, Dalia treimikienė, Sarminah Samad, and Othman Ibrahim. 2018. A new model for customer purchase intention in e-commerce recommendation agents. *Journal of International Studies* (2018).

[73] Ruslan Salakhutdinov and Andriy Mnih. 2008. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. *Proceedings of the 25th International Conference on Machine Learning* 25, 880–887. https://doi.org/10.1145/1390156.1390267

[74] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted Boltzmann Machines for Collaborative Filtering. In *Proceedings of the 24th International Conference on Machine Learning* (Corvalis, Oregon, USA) *(ICML '07)*. Association for Computing Machinery, New York, NY, USA, 791–798. https://doi.org/10.1145/1273496.1273596

[75] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based Collaborative Filtering Recommendation Algorithms. *Proceedings of ACM World Wide Web Conference* 1 (08 2001). https://doi.org/10.1145/371920.372071

[76] Andrew Schein, Alexandrin Popescul, Lyle Ungar, and David Pennock. 2002. Methods and Metrics for Cold-Start Recommendations. *SIGIR Forum (ACM Special Interest Group on Information Retrieval)*, 253–260. https://doi.org/10.1145/564376.564421

[77] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. AutoRec: Autoencoders Meet Collaborative Filtering. In *Proceedings of the 24th International Conference on World Wide Web* (Florence, Italy) *(WWW '15 Companion)*. Association for Computing Machinery, New York, NY, USA, 111–112. https://doi.org/10.1145/2740908.2742726

[78] Lipi Shah, Hetal Gaudani, and Prem Balani. 2016. Survey on Recommendation System. *International Journal of Computer Applications* 137 (03 2016), 43–49. https://doi.org/10.5120/ijca2016908821

[79] Zeinab Shahbazi and Yungcheol Byun. 2020. Toward Improving the Prediction Accuracy of Product Recommendation System Using Extreme Gradient Boosting and Encoding Approaches. *Symmetry* 12 (09 2020). https://doi.org/10.3390/sym12091566

[80] J. S. Shyam Mohan, Hanumath Sreeman Vedantham, Venkata Chakradhar Vanam, and Nagendra Panini Challa. 2021. Product Recommendation Systems Based on Customer Reviews Using Machine Learning Techniques. In *Data Intelligence and Cognitive Informatics*, I. Jeena Jacob, Selvanayaki Kolandapalayam Shanmugam, Selwyn Piramuthu, and Przemyslaw Falkowski-Gilski (Eds.). Springer Singapore, Singapore, 267–286.

[81] Khajamoinuddin Syed, William Sleeman, Kevin Ivey, Michael Hagan, Jatinder Palta, Rishabh Kapoor, and Preetam Ghosh. 2020. Integrated Natural Language Processing and Machine Learning Models for Standardizing Radiotherapy Structure Names. *Healthcare* 8 (04 2020). https://doi.org/10.3390/healthcare8020120

[82] Yan Tian and Concetta Stewart. 2007. *History of E-Commerce*. https://doi.org/10.4018/9781599049434.ch001

[83] Duy Thanh Tran and Jun-Ho Huh. 2022. New machine learning model based on the time factor for e-commerce recommendation systems. *The Journal of Supercomputing* 79 (11 2022), 1–46. https://doi.org/10.1007/s11227-022-04909-2

[84] Cong Wang, Yifeng Zheng, Jinghua Jiang, and Kui Ren. 2018. Toward Privacy-Preserving Personalized Recommendation Services. *Engineering* 4, 1 (2018), 21–28. https://doi.org/10.1016/j.eng.2018.02.005 Cybersecurity.

[85] Hongzhi Yin, Bin Cui, Yizhou Sun, Zhiting Hu, and Ling Chen. 2014. LCARS: A Spatial Item Recommender System. *ACM Trans. Inf. Syst.* 32, 3, Article 11 (jul 2014), 37 pages. https://doi.org/10.1145/2629461

[86] Sheng Zhang, Weihong Wang, James Ford, and Fillia Makedon. 2006. Learning from Incomplete Ratings Using Non-negative Matrix Factorization. *Proceedings of the Sixth SIAM*

International Conference on Data Mining 2006. https://doi.org/10.1137/1.9781611972764.58

[87] Zhi-Dan Zhao and Mingsheng Shang. 2010. User-Based Collaborative-Filtering Recommendation Algorithms on Hadoop. *2010 Third International Conference on Knowledge Discovery and Data Mining* (2010), 478–481.

[88] Yin Zheng, Bangsheng Tang, Wenkui Ding, and Hanning Zhou. 2016. A Neural Autoregressive Approach to Collaborative Filtering. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48* (New York, NY, USA) *(ICML'16)*. JMLR.org, 764–773.

[89] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. 2008. Large-Scale Parallel Collaborative Filtering for the Netflix Prize. 337–348. https://doi.org/10.1007/978-3-540-68880-832