NOVA IMS
Information Management School

UTILIZING IMPLICIT FEEDBACK DATA TO BUILD A HYBRID RECOMMENDER SYSTEM

Ehsan Meisami Fard

Dissertation presented as partial requirement for obtaining the Master's degree in Data Science and Advanced Analytics

NOVA Information Management School
Instituto Superior de Estatística e Gestão da Informação
Universidade Nova de Lisboa

NOVA Information Management School
Instituto Superior de Estatística e Gestão de Informação
Universidade NOVA de Lisboa

UTILIZING IMPLICIT FEEDBACK DATA TO BUILD A HYBRID RECOMMENDER SYSTEM

by

Ehsan Meisami Fard

Dissertation presented as partial requirement for obtaining the Master's degree in Data Science and Advanced Analytics

Adviser: Professor Dr. Roberto Enriques

November, 2022

Utilizing implicit feedback data to build a hybrid recommender system

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v6.10.10) [1].

To my parents Morteza and Parvin, whose good example has taught me to work hard and sacrifice for the things I want to achieve. To Shirin, Rasoul and Mehdi, my dear sister and brothers.

Acknowledgements

First, my most profound appreciation to my teammates at Magaloop, especially Sascha Kastert, who introduced me to the FMCG world, gave me helpful advice on this work and generously provided his knowledge and expertise.

I could not have embarked on this journey without Professor Dr Roberto Henriques, who patiently guided, encouraged and advised me throughout my studies and whose lectures at NOVA IMS influenced and inspired me to explore the field of recommendation engines.

Last but not least, I thank my dear friend and former colleague, Konstantin Biel, who believed in me and introduced me to the field of analytics for the very first time.

"I wish you bad luck." (John Roberts)

Abstract

In e-commerce applications, buyers are overwhelmed by the number of products due to the high depth of assortments. They may be interested in receiving recommendations to assist with their purchasing decisions. However, many recommendation engines perform poorly in the absence of community data and contextual data. This thesis examines a hybrid matrix factorisation model, LightFM, representing users and items as linear combinations of their content features' latent factors. The model embedding item features displays superior user and item cold-start performance. The results demonstrate the importance of selectively embedding contextual data in the presence of cold-start.

Keywords: Recommendation systems, Cold-start, Matrix factorisation, Implicit feedback

Contents

List of Figures

List of Tables

1 Introduction

1.1 Paradox of Choice

It is a common understanding within the current modern society and economy that the more choices available, the better and the human desire for choice is infinite. The idea is that more options should increase the chance of a successful search. In many cases, this association is the primary driver of organizations competing to offer their customers the most diversity of products and services. However, there is an increasing amount of evidence and research that suggest that people can have difficulties managing complex decisions. Research has shown that facing many alternatives could result in postponed decisions, searching for new alternatives, or opt not to choose [2].

Furthermore, recent research in cognitive psychology has disclosed the impact of choice overload and the paradox of choice. If choices are important and success is personally critical, it will lead to impaired decision-making and degraded satisfaction [3]. For instance, the research found that passersby are more likely to buy jams on display and more satisfied as customers when there are six jams to choose from than twenty-four [4].

These decision-making issues caused by choice overload gain significant importance in retail and e-commerce industries due to the increased product depth that these organizations provide for their customers, which is a consequence of the low costs of inventory like online retailing. It grants online retailers an advantage: to offer their customers an enormous product assortment, explained by [5] as the availability of various shapes of products offered by marketers to be owned or consumed by consumers. Therefore, the movement toward e-commerce has enabled these sites to provide their customers with a broad range of products. Competing as an online retailer in today's highly competitive retail industry has become necessary. Moreover, geography does not constrain customers' expectations and desires concerning product depth and quality. Amanah and Harahap investigated the impact of product assortment on online purchases decision among university students in Indonesia. They found that product assortment was a vital and influential factor in students' online purchase

decision-making [6]. However, in expanding to this new level of increased alternatives, businesses escalate the information customers must process before selecting items that meet their needs. Many researchers suggest that the vast number of substitutes leads consumers the experience information overload. It is a negative affective effect caused by the surplus of information and the consumer's processing capacity [7]. Moreover, from the customers' perspective, selecting from among a large number of options tend to shoot the decision-making difficulty and thus leads consumers to rely more on easy-to-obtain justification in purchasing an item [8]. This choice overload problem is an issue that online e-commerce platforms have been attempting to address since the nineties [9] by aiding customers to discover exciting and related products in the retailer's assortment.

The choice overload has been partly addressed by creating capable search engines which are increasingly optimized during the last decades. Nevertheless, search engines are only helpful when the customers are aware of the specific name of the product they are looking for and are specifically aware of their wants and needs. There are instances in which customers are unaware of their needs, nor have they thought of their need for a specific item. As a result, E-commerce sites use Recommender Systems to suggest products to their customers. As search engines actually require user input in order to work, how Recommender Engines help users to find the appropriate item for their needs work differently. Generally, the products can be recommended based on their popularity and sales volume, the customer's demographics, or an analysis of the past buying behaviour of the customer as a prediction for future buying behaviour [10]. Recommender Engines use different methodologies such as similarrity metrics to produce item suggestions that might interest users and utilize the user's behaviour and meta data [11].

1.2 Recommender Systems

Generally, a recommender engine or a recommender system is a computation that recommends user-specific items [12]. In this case, the term item represents tangible or intangible products, and a user is a person who obtains the recommendation based on the assumed preferences. Since the early nineties, recommendation technologies have improved tremendously. Karlgren introduced Recommender Systems in his paper: An Algebra for Recommendations as an intelligent bookshelf [13]. The intelligent system was intended to enable readers to obtain a collection of books tailored to them. Despite the advanced idea, Karlgren's paper was rejected by reviewers of the INTERACT committee in 1990 because such a system would interfere with the privacy and integrity of the users. As of now, 30 years after the publication of the first paper on this topic, it is hard to imagine our everyday life without them. The usage of such intelligent systems is across all online sites from social media to e-commerce and streaming platforms such as YouTube, Amazon and Netflix [14–16]. Amazon utilizes at

least seven different recommendation systems simultaneously [17]. In 2006, Amazon CEO Jeff Bezos stated, "If I have 3 million customers on the Web, I should have 3 million stores on the Web", suggesting that every single customer on Amazon should have a unique experience and Amazon should adjust itself towards their personalized needs and preferences [18]. In the same year, Amazon stated that 35% of its sales originate from recommendations made to the customer. However, Amazon is one of many sites to utilize this powerful weapon to maximize customer satisfaction and revenue. YouTube has a Recommender System for personalized video recommendations. Spotify uses a Recommender System to predict the potential songs the user might favour. Lastly, social media sites such as Facebook and LinkedIn use it for recommending social, and professional connections [12, 19].

Ricci et al. mention five merits of such an intelligent system [12]. A recommender system could increase the number of sales and, thereby, the revenue. It also improves the diversity of the items sold since suggestions made by the system could embrace unwanted items that are not easily found. Additionally, a recommendation system is beneficial for e-commerce sites, and customers are generally pleases with the presence of a recommender system. Consequently, this delight could lead to customer loyalty towards the retailer. Finally, the retailers can gain insight into the needs and preferences of the customer. In addition to Ricci et al. advantages, recommender systems could assist excellent products in exposing themselves to the potential consumer, which leads to disproportionately reduced search costs for niche products. According to Anderson, these systems tend to deflect the demand from blockbusters away toward niches movies that better meet consumer preferences [20]. Due to the many merits of these systems, retailers are creating specified user profiles for better conceiving customer behaviour and habits to provide their customers with more great user-specific recommendations. Numerous marketing studies suggested that penalization enhances revenue and that customer satisfaction [21–23]. Moreover, as of now, customers progressively expect retailers to have an accurate and user-tailored recommendation [24].

1.2.1 Small Online Retailers and Recommender Systems

Kaminskas et al. found that usually, it is challenging to implement the currently available recommendation systems to smaller-scale online retailers due to the small amount of data they possess [25]. In contrast to more prominent retailers, such as Amazon, they have fewer user-item interactions such as purchases, added to basket or explicit rating data, which constrains how an accurate recommender system can be constructed [25]. There are especially a few recommender systems that need more sizable amounts of data and thereby need help to recommend appropriate items to consumers. The absence of data and user-item interaction is also known as data sparsity. A sparsity problem arises due to user interactions with a minor portion of items in the particular domain, such as when users generally rate or purchase only a limited number

of items. In addition to sparsity, the cold start problem is another major obstacle to overcome for specific recommender techniques. According to Guo, cold start "refers to the difficulty in bootstrapping the RSs for new users, or new items" [26]. The lack of rating and user-item interaction is generally due to users' need for awareness or incentives to rate items. Although many techniques have been proposed to prevent these issues, an established solution has yet to be presented [26].

1.3 Objectives and Practical Relevance

The primary purpose of the thesis is to implement a recommender system for an e-commerce application capable of providing suggestions of products to a user based on similarities concerning the behaviour of other users. The similarity is computed by considering user interactions (i.e. the number of views and transactions) on a particular product. Then, exploiting this similarity creates a list of recommended products. Moreover, we want to investigate the quality of the recommendation by introducing hybridizing the model and including item data and user data (metadata). Due to the high inflow of new users (new customers) and items (new supplier products), we want to test the model in user and item cold-start problems. To attain this objective, the subsequent research questions were developed:

1. **Research Q1:** Could a mixture of implicit data on users and contextual data on items increase the performance beyond the performance achieved with a pure collaborative filtering model?

2. **Research Q2:** Does a hybridized model with a mixture of implicit data on users and contextual on items aid the recommender system to increase the accuracy in case of user- and item cold-start problem?

1.4 Structure of Thesis

This thesis is constructed as the following. First, the following Chapter will explore the theoretical background of some of the fundamental techniques in the field of recommendation systems which allows us to get a general technical understanding of it. Subsequently, Chapter 3 introduces how the data is fetched, retrieved and used to build a recommender system. Hence, the outcome of the experiment is presented in Chapter 4. Chapter 5 discusses the outcome and responds to the research questions made, and finally, the future work discusses potential further improvements that can be made yet restricted to Magaloops circumstances. Lastly, Chapter 6 concludes all previous chapters.

2

Background

This chapter introduces the basic concepts of different Recommender Systems. It discusses the dominant challenges and issues for individual Recommender Systems, which form the foundation argument for designing the new hybrid method discussed in this work.

2.1 Definition of Recommender System

The Recommender problem consists of the inputs of sets U and I, where U represents the set of all users, and I represents the set of all items. Each of the users in U specifies ratings from a set S of possibilities for the items I present in the system (e.g., S = [1,5] or S = like, dislike). The resulting rating matrix R is composed of $R = U \cdot I$ which shows that entries in R indicate the rating of user U to the item I [27]. To describe the subset of users interacting with an item I, we use the notation U_i while I_u represents the subset of items rated by a user U. Moreover, recommender systems are often classified into the following groups:

- **Content-based recommendations:** user receives items suggestions similar to the ones the user has interacted with or liked in the past.
- **Collaborative recommendations:** user receives items suggestions that the user with similar tastes preferred in the past.
- **Hybrid methods:** Usually a combination of at least two recommendation techniques, such as content-based and collaborative filtering.

2.1.1 Explicit and Implicit Feedback

Generally, the data used for a specific model are based on explicit or implicit data where an explicit data is created by a user, while an implicit data is created from user-specific actions. Alternatively stated, explicit feedback illustrates the preference of a user head-on, while implicit feedback displays the preference of a user indirectly

	Explicit feedback	Implicit Feedback
Accuracy	High	Low
Abundance	Low	High
Expressivity of a user preference	Positive and negative	Positive
Measurement reference	Absolute	Relative

Table 2.1: Explicit and Implicit feedback summarized

[28]. However, despite tremendous investments in projects that help the company extract as much explicit customer feedback, in an imperfect world, explicit feedback is only sometimes obtainable. According to Jannach et al., "the main problems with explicit ratings are that such ratings require additional efforts from the users of the recommender system and users might not be willing to provide such ratings as long as the value cannot be easily seen [29]. Thus, the number of available ratings could be too small, resulting in poor recommendation quality." For example, a Netflix user might be reluctant to give out a personal rating of the movie right after watching it or even later on. As a consequence, an alternative solution to a user's willingness to rate products and services is together user preference intelligence through implicit feedback [30]. For instance, a user watching Sci-Fi movies several times could show their inclination for that specific genre. Furthermore, in the domain of the smaller-scale e-retailers, explicit feedback is rarely available [25] while implicit feedback is almost always available [31]. Implicit ratings are usually generated by the webshop into which the recommendation engine is integrated. For instance, one could interpret the purchase of an item as a positive rating, while a return of an item could indicate that the user did not have a pleasant experience with the item purchased. Moreover, data could also be generated by monitoring the user's browsing behaviour. Suppose the user lands a page with detailed item description and stays on this page for a long duration, for example. In that case, a recommender could define the time spent on a specific item description as a preferable inclination towards that item. On the contrary, despite the rich supply of implicit ratings, one needs to be more confident whether the customer behaviour is accurately translated. For instance, a user might purchase an item for a friend, which could not necessarily be interpreted as a positive sign toward the item. In addition, implicit ratings do not capture the user's experience with the item; it only captures the user's interest towards the item prior to consuming the item. As a result, a high number of implicit ratings are necessary to neutralize its ambiguity. The explicit and implicit feedback features are summarized in Table 2.1.

2.2 Collaborative Filtering

The name of collaborative filtering originates from the notion that "people collaborate to help one another perform filtering by recording their reactions to documents they read" or any other item feedback [32]. Most collaborative filtering methods leverage item-to-item correlations or user-to-user correlations for the recommendation process. At the same time, a few models utilise both types of interrelationship [33]. In the field of Collaborative Filtering, two sort of techniques are usually utilized, which are introduced as memory-based and model-based mechanism. On the one hand, memory-based models are also known as neighbourhood-based collaborative filtering techniques. According to Aggarwal, "these were among the earliest collaborative filtering algorithms, in which the ratings of user-item combinations are predicted based on their neighbourhoods," consisting of User-based Filtering and Item-based Filtering. On the other hand, model-based methods apply data mining as well as machine learning methods such as decision trees, rule-based models, Bayesian methods and latent factor models. Model-based methods have numerous advantages over neighbourhood-based methods. Firstly, the size of the model-based models is in fact significantly smaller than the original matrix ratings. Secondly, the model-based methods are much faster and more efficient in the preprocessing process of constructing the trained model relative to neighbourhood-based models. Lastly, one of the main issues with sparse matrices is the small set of ratings that could lead to overfitting. In this case, model-based methods could address this problem by using regularization. Regularization decreases the inclination of the mechanism to overfit by integrating a bias into the mechanism [33].

2.2.1 User-based Fitlering

User-based Collaborative Filtering aims to prioritize the search for similar users, also known as neighbours. Similar to Item-based Filtering, prioritization saves a substantial number of calculations as the whole user-items matrix is not considered. As a result, a series of items are suggested which are commonly consumed by homogeneous users [34]. Some conventional approaches are the K-Means and the K-Nearest-Neighbor algorithms, which identify similar users. While the former creates a finite number of user groups, the latter defines a finite number of homogeneous users to a specific one. Different methodologies are utilized to perform user-based filtering, such as Pearson's correlation coefficient. However, methods such as Spearman's rank correlation coefficient and adjusted cosine similarity have been established in the field of recommendation systems to calculate the similarity between users. Nevertheless, several empirical research illustrates the superiority of the Pearson coefficient relative to other measures, which captures the similarity between the rating vectors of two users, a and b [35]. As a result, the Pearson correlation coefficient is defined as:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{2.1}$$

2.2.2 Item-based Fitlering

The power of Collaborative Filtering lies in taking each rating of each user at disposal into consideration. Although User-based Collaborative Filtering Systems have had success in numerous domains, some severe difficulties still need to be addressed regarding e-commerce sites with millions of users and items. As a result, as the number of neighbours among a significant user population increases, the more time-consuming the search for these neighbours. The necessity to calculate many potential neighbours makes it exceptionally difficult to produce recommendations in actual instantaneously, which is a vital performance dimension given the current demand of swift page loading in with current user expectations. According to Sarwar et al., scalability is a significant challenge of User-based collaborative filtering systems despite their success in the past [36]. E-commerce sites with a large number of users and items often implement an item-based recommendation, which could speed up the recommendations shown to the user through offline preprocessing and thus allows for the computation of recommendations in real-time, even for an extensive rating matrix. Moreover, the authors explain that the intuition behind Item-based Filtering is that "a user would be interested in purchasing items that are similar to the items user liked earlier and would avoid items that are similar to the items the user did not like earlier". The objective of Item-based Collaborative Filtering is to identify appropriate items based on other items' consumption information, which prevents the problem by prioritizing the relationship among items first rather than the relationships between users. The primary objective of item-based algorithms is to make recommendations utilizing the similarity among items [36].

A similarity measure must be defined to discover similarity among items in the database. According to Jannach et al., "in item-based recommendation approaches, cosine similarity is established as the standard metric, as it has been shown to produce the most accurate results [29]. The metric measures the similarity between two n-dimensional vectors based on the angle between them". This similarity measure is generally utilized in text mining to compare two text documents [33]. The proximity between item a and b – regarded as the corresponding rating vectors a and b – is formulated:

$$similarity = cos(\alpha) = \frac{a \cdot b}{||a||_2||b||_2} = \frac{\sum_{i=1}^{n}(a_i b_i)}{\sqrt{\sum_{i=1}^{n} a_i^2}\sqrt{\sum_{i=1}^{n} b_i^2}} \tag{2.2}$$

The $\cdot$ symbol is the dot product of vectors. $|a|$ is the Euclidean length of the vector [33]. The output value range is between 0 and 1 while 1 indicates a substantial similarity between item a and b. Besides cosine-based similarity, there are other techniques, such as correlation-based similarity and adjusted cosine similarity which could also be utilized to compute item similarity. Sarwar et al. compared adjusted cosine, basic cosine and correlation in an experiment and found that adjusted cosine similarity has a clear advantage due to its lower mean absolute error or MAE that was used as the evaluation metric [36].

2.2.3 Principal Component Analysis

Goldberg et al. proposed a unique approach to dimensionality reduction, Principal Component Anaylsis, which intends to extract the predominant information of the data that concludes for the majority of the variations [32]. The amount of variance is obtained by the first component is more considerable than the amount of variance on the second component which enables the dimensionality reduction of the data by neglecting the components with a marginal contribution to the variance. Consequently, the users are grouped into clusters of neighbours.

2.2.4 Matrix Factorization

In the Netflix Prize competition, which was held in 2009, sophisticated model-based filtering algorithms such as matrix factorization methods displayed a superior predictive accuracy. Matrix factorization methods generate a set of hidden factors known as latent factors from users and items. For example, in the movie domain, this could translate into different features of a movie, such as a genre or the actors and the producers involved. As a result, a recommendation is made whenever a high correspondence exists between the item and user factors [37]. Moreover, the authors illustrate one of the merits of choosing such a recommender system: "it allows incorporation of additional information" in the absence of explicit feedback. Thereby, the system can utilize implicit feedback, reflecting the user's behaviour implicitly. To reduce the problem of sparsity in the datasets, matrix factorization is one of the optimal approaches, primarily utilized for its ability to process large databases and provide scalable approaches [37]. Furthermore, to recommend N items with f latent factors to M customers, the memory cost of matrix factorization recommendation is $f * M + f * N$, but the similarity-based recommender method has the memory cost of $M2/2$. Let A be the matrix of users M and items N, which contains all the ratings that the users have allocated to the individual items. The objective of matrix factorization is to decompose the original item-user matrix A into two new matrices that is able to make predictions which by multiplying the two decomposed matrices. Consequently, this produces a sparse matrix where each row and column represents users and items, respectively.

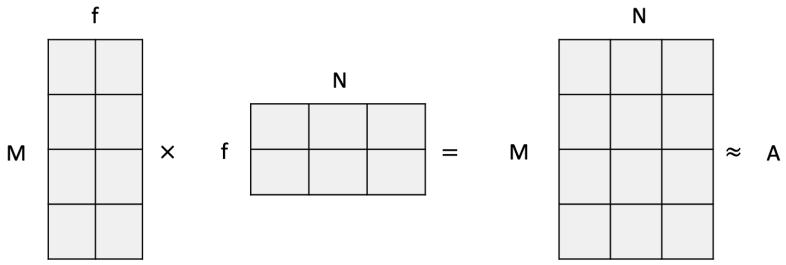

Figure 2.1: Concept of Matrix Factorization

As shown in 2.1, the $M \cdot N$ rating matrix A I decomposed into the user latent factor matrix X and the item latent factor matrix Y. The attributes of items such as movie category or music genre are embedded in factor vectors. As an example, if a customer has a high factor score, it hints at customer preference for that attribute (e.g. music genre). Matrix factorization aims to find the mapping between users and items in factor vectors. To determine the factors x_u and y_i, the value in matrix A is decomposed into $x_u^T y_i$ with the objective of minimizing the following cost function:

$$min_{x,y} \sum (r_{u,i} - x_u^T y_i)^2 + \lambda(||x_u||^2 + ||y_i||^2) \tag{2.3}$$

2.2.5 Alternating Least Square

Hu et al. altered the cost function of matrix factorization for implicit data since the value of implicit feedback cannot be directly mapped to users' preferences by changing the initial rating r_{ui} in the cost function (2.3) with the binary value p_{ui} and the confidence weighting c_{ui} as follows [38]:

$$min_{x,y} \sum c_{ui}(p_{u,i} - x_u^T y_i)^2 + \lambda(||x_u||^2 + ||y_i||^2) \tag{2.4}$$

where $c_{ui} = (1 + \alpha r_{ui})$ and α is the rate of increase in confidence r_{ui}; p_{ui} is derived by binarizing r_{ui} which indicates the user's inclination on the rating item. The values of p_{ui} are:

$$X = \begin{cases} 0, & r_{ui} > 0 \\ 1, & r_{ui} = 0 \end{cases} \tag{2.5}$$

2.2.6 Challenges of Collaborative Filtering

Despite many innovations and novel techniques that minimize the weaknesses of Collaborative Filtering, there are a few inevitable challenges. A significant difficulty when using collaborative filtering techniques is the sparsity of the matrices that include

the ratings. Alternatively stated, the amount of ratings acquired is generally quite insignificant relative to the extent of ratings required to make a suggestion [9]. For example, the users on an e-commerce platform usually interact with a small portion of the items available from all of the products; therefore, most item ratings are unknown. Thereby, the resulting sparsity of the user-item matrices lessens the performance and accuracy of a recommender system [39]. Furthermore, the number of users and items directly affects the required computing power. CPUs and other hardware resources could be crucial to prevent service collapses. As discussed before, the larger the number of users within the dataset, the longer it takes to make a recommendation due to limited computing power. The current solution for such an issue is to perform a preprocessing offline and thereby update the model in an interval. Lastly, from the perspective of the user, collaborative filtering operates similarly to a black box because the resulting recommendations are hard to explain. Consequently, it could lead to user confusion and frustration. Herlocker et al. explain four central merits in transparency when showing the resulting recommendations [35]: (1) explanation of suggestions, (2) increased user engagement, (3) understanding of the user regarding the limitations of the engine (4) easier user adoption and apprival. However, despite the many benefits of transparency, it is challenging to explain sophisticated mathematical models to a regular user.

2.3 Content-based Filtering

The collaborative filtering methodologies examined in the last section use correlations in user rating patterns to recommend items. Such a method disregards item attributes for computing prediction [33]. For example, if a user is interested in a specific music genre, then there is a high probability that the user might like other songs from the same genre. In these instances, the input of other users may not be necessary to make valuable user-specific suggestion. Instead, its ratings might be sufficient, especially when there is a lack of rating for a specific item due to its novelty, also known as the item sparsity. The sparsity problem frequently manifests in e-retail, where customers purchase only a handful of items despite a large number of items available. Solutions concerning this issue will be discussed later in this thesis. Content-based systems, in contrast to collaborative systems, utilize the target user's ratings and preferences by the user as well as the corresponding attributes to make new recommendations which could be advantageous based on the data that is available [33]. The general principle of content-based recommender systems is to classify the standard features of items that have received a rating from user U and then recommend items with similar characteristics to user U [27]. In other words, content-based recommender systems attempt to match users U to items I similar to their preferred items previously based on attributes of the objects liked by the user. The most popular Content-based recommender techniques are Artificial Neural Networks (ANNs), Bayesian Classifiers, Clustering, Decision Trees, and TF-IDF for information retrieval [9]. In the general

constellation, content-based systems require two sources of data:

1. A description of item-attributes
2. A user profile produced from user ratings about different items, which is either defined as implicit or explicit, whereas implicit feedback is associated with total user actions (e.g. click, purchase, cancellation) and explicit feedback could be formulated as a simple rating (e.g. 1-5).

User profiles are generated through a process known as "profile learning" [40]. Essentially, it creates a representative profile of the customer by consolidating profiles of items with which the user has interacted. Content-based systems are utilized when a significant amount of item-specific attribution data is available. As a majority of these data are in the form of text, content-based systems are well suited to give recommendations in text-rich and unstructured domains, which can be stored in a relational database along with relational attributes such as genre, and price [33]. A simple method to structure item descriptions and attributes is to use Term Frequency-Inverse Document Frequency (TF-IDF) analyses [40]. Blei et al. introduced a more advanced technique, Latent Dirichlet Allocation (LDA); meanwhile, Musto et al. utilized Neural Network-based techniques such as Word2Vec [41] [42]. Since content-based models function based on textual and description features, their recommended items are highly interpretable insights. For instance, a movie recommendation system could demonstrate the reason behind its suggestion to the user, such as movie genre, actors, director, producer and more. Meanwhile, collaborative filtering-based recommender systems cannot demonstrate the reasoning behind their suggestions due to the nature of the utilized data. Lastly, it is essential to implement feature selection to be selective which words are worth keeping which excludes the noisy words from the dataset that cause overfitting of user-profiles [33].

2.3.1 Nearst Neighbour Classification

One of the most popular and classification techniques is the nearest neighbour classifier which can be applied relatively effortlessly. Such a technique is based on a similarity function for instance the cosine function which is employed quite frequently and is most common. Cosine similarity is an estimate of similarity that can be applied to compare documents. In this case, the cosine measure is defined as follows:

$$sim(x, y) = cos\theta = \frac{x \cdot y}{||x||y||} \tag{2.6}$$

where $||x||$ is the Euclidean norm of vector $x = (x_1, x_2, x_3, \dots)$ and $|y|$ is the Euclidean norm of vector y. When the value of cosine is zero, it indicates that the two vectors are orthogonal to each other and have no correspondence. On the contrary, the closer the value to one, the the greater the correspondence between the two vectors.

Such a function is regularly utilized in the text domain due to its capacity to adapt to different lengths documents. In contrast, structured data are handled through Euclidean distance and Manhattan distance. For each item, its k-nearest neighbours utilize the cosine similarity and the average rating for k neighbours of each item is calculated. The items are positioned based on the predicted rating, and the items position at the top of the list are suggested to the user. However, to calculate the k nearest neighbours, the nearest neighbour of each item in the dataset requires to be determined. Consequently, the amount of time needed proportional to the magnitude of the dataset. One way to minimize the computational complexity of this approach is to utilize clustering to decrease the number of training items. Usually, a centroid-based clustering technique such as the k-means is utilised to construct each group of clusters [33].

2.3.2 Challenges for Content-based Filtering

It is important to notice that community data is not used as input for the content-based recommendation algorithms. Depending on the scenario, such an approach is both an advantage and a disadvantage. When facing a cold-start situation which is described as lack of information or ratings regarding the user or time, such an approach is effective as long as abundant data about the user's preference exists. However, discarding other users from our model induces over-specialization of users' preference portfolios. As a result, the suggestion made to the user is limited to similar items that the user already rated. Additionally, this issue can be further aggravated because the user finds it hard to go beyond their preferences which origins from the accumulation of data regarding its own preference [9]. Moreover, there is a limit to the number and types of features associated with items. Consequently, more than the attributes available for an item may be required to discriminate items that a user likes from those he dislikes [40]. According to Adomavicius and Tuzhilin, "if two different items are represented by the same set of features, they are indistinguishable" [9]. Furthermore, the authors also point out the new user problem of the content-based system. They argue that "the user has to rate a sufficient number of items before a content-based recommender system can really understand the user's preferences and present the user with reliable recommendations. Therefore, a new user, having very few ratings, would not be able to get accurate recommendations". Lastly, content-based methods need well-structured data from an unstructured dataset and the importance of the preprocessing steps needed to convert unstructured data into structured data, which a model can benefit from.

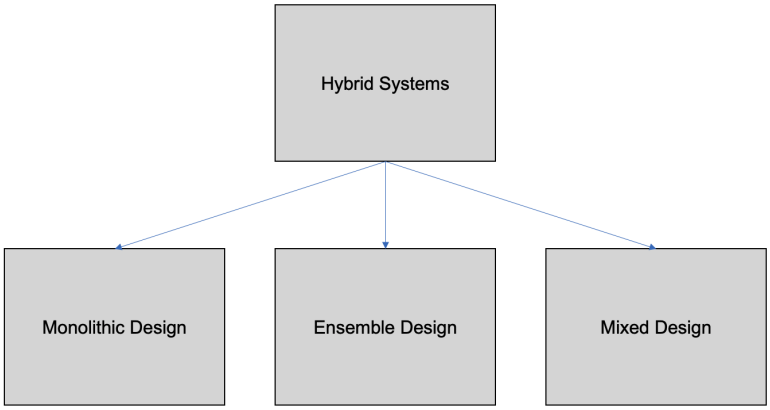

Figure 2.2: Three different types of hybrid recommender systems

2.4 Hybridization

As discussed before, collaborative and content-based methods utilize a different types of feedback to make recommendations. As such, these methods also possess their unique strengths and weaknesses in isolation, and each technique requires a specific type of data to make recommendations. In general, it would make sense to take advantage of all available feedback to obtain better recommendations. As a result, hybrid recommender systems have been researched and designed to overcome some of the aforementioned shortcomings and issues. Although numerous e-commerce applications use hybrid recommender systems, more research is required hybrid algorithms and situations that one can expect a benefit from hybrid models.

The research suggests that hybrid methods have outperformed collaborative filtering models in isolation [43] [44] [33]. Three primary types of hybrid recommender systems are shown in figure 2.2. That are the monolithic design, the ensemble design, and lastly, the mixed design.

2.4.1 Monolithic Design

A monolithic recommendation system uses mixed components of different recommenders to improve the overall performance of the suggestions. Essentially, it includes parts of different types of recommendation algorithms and combines components from different recommendation engines to improve the overall performance of the recommendations. For instance, the resulting output of a content-based approach that finds similar items could be used as input for collaborative filtering incorporating community data.

2.4.2 Ensemble Design

An ensemble is a group of things or people acting or taking together as a whole. Therefore, in this case, it defines a combination of predictions from different recommenders into one recommendation. For instance, one could mix the recommendations made from a collaborative as well as content-based recommender one into a individual results. Figure 3 illustrates an example of how an ensemble hybrid model might function. In this design, there are two primary models, namely, the switched and weighted ensembles.
The switching hybrid mechanism chooses the most suitable recommender given the context of the request. In the absence of community data for a specific customer, the ensemble model might use the content-based recommender instead of the collaborative recommender. In other terms, the system switches between recommendation systems depending on that specific situation. According to Burke switching hybrid mechanisms introduce additional complexity into the preprocessing process since the switching criteria must be determined [45]. Furthermore, switching models are often utilized to manage the cold-start problem [33]. As an example, in a case study on how user feedback can be used for personalization in e-Commerce scenarios, a combination of knowledge-based and collaborative systems has been experimented with. While the knowledge-based system suggests relatively better recommendations in the presence of sparsity and cold-start, the collaborative mechanism produces better recommendations when sufficient community data is collected. Therefore, knowledge-based systems are generally utilized to manage the cold-start problem.
The weighted hybrid mechanism combines scores of multiple recommender systems into a single score by computing the weighted aggregates of the scores from individual systems.

2.4.3 Mixed Design

Lastly, there are hybrid models based on mixed design. Burke argues that "it may be possible to use a 'mixed hybrid', where recommendations from more than one technique are presented together" [45]. Put recommendations from numerous engines are presented to the user simultaneously. According to Aggarwal, mixed recommenders cannot be categorized as monolithic nor ensemble-based techniques as they do not combine the predicted ratings of the same item from different recommenders [33]. As a result, mixed recommender systems are classified into a unique category of their own. The author states that "the main distinguishing characteristic of such systems is the combination of presentation rather than the combination of predicted scores". In terms of unique properties, Burke argues that "this technique has the desirable 'niche-finding' property in that it can bring in new items that a strict focus on content would eliminate" [45].

3 Methods

This chapter describes and introduces the data used for this work. In particular, it illustrates how the data for the Recommender System has been fetched and retrieved. Subsequently, the next step is to build an appropriate recommender system which fits the needs based on the generated data and goals set in first chapter. Moreover, a brief statistical analysis is demonstrated, which better represents the dataset.

Figure 3.1: Google Analytics and BigQuery

3.1 Data

3.1.1 Google Analytics and BigQuery

Web analytics involves gathering, measuring, analyzing, and reporting web usage data to understand users' behaviour. Many organizations utilize analytics to optimize websites to enhance customer satisfaction and loyalty [46]. In this case, Google Analytics is used to track user behaviour. Google Analytics is a web analytics service offered by Google to collect, measure, and report user behaviour.

Google Analytics offers information on the geographic origin of users and the type of device with which the user operates. Furthermore information can be derived by breaking down the URL. For instance, the Urchin Tracking Module (UTM) is specifically designed to provide the most accurate measurements of unique website visitors and attributes the users' visit to a specific traffic source. Despite many details derived from Google Analytics, the goal is to track customer app usage to gather information about users' interaction with product items. This information can be fetched and stored in a database with all Magaloop users and items using Google BigQuery [47].

BigQuery is a developers tool that runs in the Cloud, SQL-like queries against large datasets and provides real-time insights about the data. In BigQuery, scheduling standard SQL-written queries to run regularly is possible. A table with numerous fields is imported each day of export. During the day, intraday data are imported every 60 minutes from 04:55 to 20:55 UTC (Universal Time Coordinated) when the daily import is completed. Table 3.2 demonstrates several important columns we import from Google Analytics.

Field name	Data type	Description
User		
`user_id`	STRING	The user ID set via the setUserId API
Geo		
`geo.region`	STRING	The region from which events were reported, based on IP address
`geo.city`	STRING	The city from which events were reported, based on IP address
Event		
`Event_date`	STRING	The date on which the event was logged (YYYYMMDD format)
`Event_timestamp`	INTEGER	The time (in microseconds, UTC) at which the event was logged
`event_name`	STRING	The name of the event
`Event_params`	RECORD	A repeated record of the parameters associated with this event
`Event_params.key`	STRING	The name of the user property
`Event_params.value`	RECORD	A record for the user property value

Table 3.1: Format and schema of the Google Analytics 4 property data and the Google Analytics for Firebase data that is exported to BigQuery

To build our data model, we require the following information:

- **User Id**: Identify the unique user among Magaloop customers.
- **Date**: Associated with when an event has occurred.
- **Event**: Identifies the type of user action (e.g. Clicks, Views).
- **Event Frequency**: How often an action has occurred during a specific time.
- **Purchases**: The purchase of an item from a customer.

- **Purchase Quantity**: The quantity of an item purchased within an order.
- **Item Id**: Associated with the specific item the customer interacts with.
- **Product Title**: The product name is linked to the Id of the item.

Although the purchase data are not retrievable from the Google Analytics data, we can use the purchase history data in another table to get the user-item purchases and the quantities. We can concatenate these two generated outputs into a single set of results by making purchases as part of the events. Therefore, instead of events (clicks, views) and purchases (order), we combine both into one single column *event*. Accordingly, the field name *quantity* will count the number of orders for a *purchase event* and count the number of Clicks and Views for *Google Analytics events*.

User id	*Date*	*Item id*	*Item description*	*Event name*	*Quantity*
1	2022-06-07	13	Coca Cola 12x0,5l Pet Einweg	`purchase`	4
2	2022-06-07	37	Tragetasche Blau extra stark 1x100Stück	`orderapp_scan_perform`	10
3	2022-05-13	72	BRLO Happy Pils 24x0,33l Glas	`orderapp_add-to-cart_scan`	8
4	2022-04-02	52	Fanta Orange 12x0,5l Pet Einweg	`stock-item_add-to-cart`	5
5	2022-03-28	55	Original 5.0 Pils 24x0,5l Dose Einweg	`orderapp_page-view_pdp`	3
...	...	...	...	...	...

Table 3.2: Example of a CSV Export

The resulting output returns a row for each user with at least one interaction or purchase within a time interval of a day. Each row is grouped by *User id*, *Date*, *Item id*, *Item description* and lastly, *Event name* while the *Quantity* is aggregated. In the case of *Google Analytics*, we count the frequency of an event occurrence, and in the case of *Purchase Data*, the *Quantity* is summed.

3.2 Calculating Ratings

3.2.1 Setting Weights

We can define a rating function $R_{i,u}$ that outputs a number that demonstrates how much user u is willing to buy an item i. In this case, we are interested in understanding how close the user u is to purchasing item i. Therefore, we consider the following rating of item i for user u:

$$R_{i,u} = (w_1 \times \#event_1) + (w_2 \times \#event_2) + \cdots + (w_n \times \#event_n) \tag{3.1}$$

Where

- $R_{i,u}$ is the rating.
- $\#event_n$ is occurrence frequency of $event_n$ (or just *Quantity*)

- $w_1 \cdots w_n$ are weights based on the strength of the event signal.

By setting specific weights for each event based on their significance, we can produce numeric ratings which replace the missing explicit ratings. Before calculating the weights, we first define and list all the events. Furthermore, each event will be explained in detail:

- $Event_1$ = `purchase`: Item purchase.
- $Event_2$ = `orderapp_scan_perform`: When a customer scans the item to find that specific item in the Magaloop App.
- $Event_3$ = `orderapp_add-to-cart_scan`: When a customer scans an item and immediately adds the item into the basket
- $Event_4$ = `stock-item_add-to-cart`: When a customer adds an item into basket
- $Event_5$ = `orderapp_addpage-view_addpdp`: When a customer has a look at the offer page, where a product is offered at a discount.

To set specific weights for each event, we start from a range of weights = $[1, 100]$ where 100 has the highest indication that a customer favours an item and 1 the lowest indication that an item is preferred. From all the events, a transactional event, an actual purchase of an item, is the most significant indicator of a customer's preference for an item. Moreover, scanning an item with subsequently adding it to the basket is also a significant indicator as the customer needs to physically go to the item, grab it, find the bar code, scan it and put it into a basket. Furthermore, a user also has the option to add items to the basket, similar to other e-commerce applications, which is also an indicator that a user is interested in purchasing the item. Additionally, Google Analytics lets us track whether a user has removed an item from their baskets. As a result, we can calculate the net amount in the customer's basket and reject the assumption that an add-to-basket event includes an unintentional click. Similar to scans with add-to-basket, a customer can scan the product and not put it into the basket. Lastly, there are offer pages where customers can find products at a discount price. Although glancing at product discounts is not a poor indication of interest and, therefore, should be not be highly weighted. Table 3.3 shows the assumptions that can be translated into weights.

3.2.2 Constraints and Time-decay

The current rating increases the more the customer interacts with a certain item during one day. However, at a certain point, more interaction does not necessarily translate to more customer preference or information about that user-item interaction. Therefore, it could be useful to use *cut-offs* to set a maximal relevance for the interactions. The formula returns the number of times an event occurs unless its value is higher than *maximal relevant*. Such addition to our rating model decreases the exposure to interaction outliers

Event	Interpretation	Weight
Purchase	Top rating	100
Performs scan and adds an item into basket	Very positive	80
Adds item into cart	Very positive	75
Performs scan	Positive	65
View item discounts	Indecisive	50

Table 3.3: Weights on events

and, therefore, should decrease the distance between the minimum- and maximum ratings and the rating mean. In our model, we used three interactions as the *maximal relevant* (mr) value for all event types. As a result, the equation 3.1 is replaced by the following equation:

$$R_{i,u} = max((w_1 \times \#event_1), mr) + max((w_2 \times \#event_2), mr) \\ + \cdots + max((w_n \times \#event_n), mr) \tag{3.2}$$

For each event on Google Analytics or purchase on the application, the timestamp of each action is captured and stored. Such a feature can be utilized to include recency into our current model and thereby consider recent ratings more significant than older ratings. The *window-based* and *decay-based* methods are the solution to emphasize the significant of recency [33, 48]. The *window-based* method is a binary decay function which simply ignores user information that is past a certain amount of time. Given a window size parameter W, the function $f_W(a) = 1$ for $a < W$ and $f_W(a) = 0$ for $a \geq W$ includes items whose age are less than W [48]. However, in [48], authors propose an exponential form for the time-decay function $f(a) = e^{-\lambda a}$ for $\lambda > 0$ for recommender systems which are widely used in many applications. The function of exponential decay is as follows:

$$f(t) = e^{-\lambda t} \tag{3.3}$$

where

- T_0 is the half-life parameter in days
- $\lambda = \ln(2)/T_0$ is the decay rate
- t is the age of the event in days

The decay rate λ is a parameter defined by the user, which controls the significance of the role of time when calculating ratings. The larger the value of λ, the less significant, the older ratings. In a case where time dramatically affects the recommendation success, such as new articles, the half-life T_0 should be much smaller. However, in our case, a

purchase made a month ago is still relevant for the customer. As a result, 30 days as a half-time parameter is chosen. Last but not least, each event quantity is computed on the user, date as well as the item level and followed by ratings. Consequently, the equation 3.2 is replaced by the following equation:

$$R_{i,u,t} = [max((w_1 \times \#event_1), mr) + max((w_2 \times \#event_2), mr) + \cdots + max((w_n \times \#event_n), mr)] \times timedecay_t \tag{3.4}$$

3.3 Pre-Processing and Dataframe Description

The rating calculation with the inclusion of the *maximum relevance cut-off* as well as *time-decay* function is made in Bigquery to speed up the process by letting BigQuery take over some of the pre-processing steps.

Subsequently, the generated ratings can be imported into a Python file with `pandas-gbq` module[1]. The `pandas-gbq` module is a wrapper for Google's Bigquery analytics web service for getting data from BigQuery tables using SQL queries. The results loaded are then parsed into a `pandas.Dataframe`.

User id	*Date*	*Item id*	*Item description*	*rating*
1	2022-06-07	13	Coca Cola 12x0,5l Pet Einweg	895
2	2022-06-07	37	Tragetasche Blau extra stark 1x100Stück	830
3	2022-05-13	72	BRLO Happy Pils 24x0,33l Glas	620
4	2022-04-02	52	Fanta Orange 12x0,5l Pet Einweg	550
5	2022-03-28	55	Original 5.0 Pils 24x0,5l Dose Einweg	530
...	...	...	...	...

Table 3.4: Example of a Bigquery Import into Python

In the next phase, the data pre-processing step is performed, which begins with data cleaning and rating normalization. Normalization or rescaling is a common technique used in prediction and forecasting models. Among the many normalization techniques, the common methods are *Min-Max* normalization, *Z-score* normalization, *Log scaling* and *Decimal scaling* normalization. In this situation, we choose the *Min-Max* due to its popularity and simplicity. The outcomes are depicted in 3.2 and **??** for *Min-Max* and *Log scaling*, respectively.
The general equation for *Min-Max* normalization of $[0, 1]$ is given as:

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{3.5}$$

[1] https://pandas-gbq.readthedocs.io/en/latest/

To scale the numbers from $[0, 10]$, we multiple x' by 10. Figure 3.2 shows the right tailed distribution of the normalized ratings is depicted in figure 3.2.

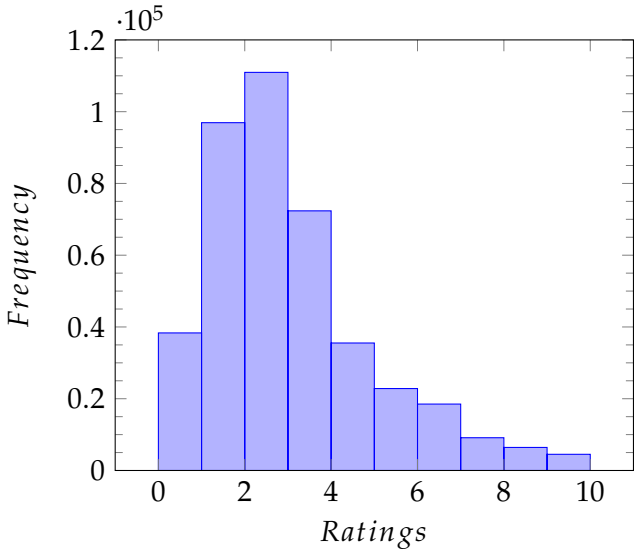

Figure 3.2: Min-Max Normalization: Rating distribution

The final dataframe subsequent to rating normalization consists of four columns, namely, *user id*, *item id*, *item description* and lastly *rating*. The dataframe has 466170 rows and does not have a missing value. In total, there are 4604 and 43396 distinct users and item, respectively. The maximum amount of interactions a user has with one single item is 2003 and the minimum amount is 1 which is minimum required interaction a user needs to have with any item to be in the dataset.

3.3.1 User Features and Item Features

The metadata for user features allows the recommender system to generate more information regarding the user by using content-based data. In the absence of sufficient user interactions (>5 interactions), such information could be vital to make appropriate suggestions despite the need for more information regarding the user's preferences. Magaloop does not collect important data regarding the demographics of its users. The only data available are *region* and *customer type*. The region is the state where the user has his or her shop located. If the retailer has a shop in Berlin, the abbreviation *ber* will be used to describe its location. Moreover, there are two types of the user utilizing the Magaloop platform. On the one hand, Magaloop has retailers, such as late-night corner shops, called *trader*. On the other hand, there are also other customers, such as, restaurants which are called *hybrid*.
The item features consist of five distinct metadata for each item, such as *category 1*, *category 2*, *supplier name*, *brand*, *manufacturer*. The first category, *category 1*, describes the

item from a high level while the second category, *category 2* describes the item from a lower level. For instance, a drink can be a beer or a non-alcoholic drink or an energy drink. While an energy drink is a non-alcoholic beverage, the energy drink category makes up a large portion of the assortment, and that justifies its individual dedicated category.
Moreover, the designated supplier name for the respective *item id* is described as well as the brand name of the item and the manufacturer. In 3.5, a snippet of the item feature dataframe is illustrated.

item id	*category 1*	*category 2*	*category 3*	*supplier name*	*brand*	*manufacturer*
1	Getränke	Bier	Bier	BRLO	BRLO	BRLO
2	Tabakwaren	E-Zigarette	Sonstiges	Inter Tabak	IQOS	Philip Morris GmbH
3	Getränke	Spirituosen und Barmixgetränke	Vodka	Inter Tabak	Landwirth's	Berentzen-Gruppe AG
4	Getränke	Alkoholfreie Getränke	Erfrischungsgetränke	B.E.S.T.	Bionade	Hassia Mineralquellen
5	Süßwaren	Sonstige Snacks	Sonstiges	izi 36	Bifi	BiFi Snacks LSI
...	...	...	...	...	...	...

Table 3.5: Dataframe Item Features

3.4 Model

3.4.1 LightFM

There are a few factors that conspire to make recommendations challenging in the case of e-commerce, specially Magaloop. Firstly, our data includes a significant number of products which translates to very sparse data. Second, as Magaloop is still a very young start-up which has its focuses on expansion, market share increases geographically, which translates to a high rate of customer and supplier acquisition. Therefore, we have a high number of new users as well as new items in our data, which further underlines the sparsity in our data. Nevertheless, despite all of these challenges, we would like to present our user's compelling recommendations despite the limited amount of information available. Given the challenges facing, we consider a hybrid content-collaborative model, called LightFM due to its similarity to factorization machines [49], which are a new model class that combines the advantages of Support Vector Machines (SVM) with factorization models [50]. In [49], Kula outlines the advantages of the model and presents the empirical results on two datasets, showing that:

1. LightFM achieves at least the same amount of accuracy as a pure content-based model.

2. It outperforms pure content-based models by including community data (collaborative) or customer information (user features).

3. In absence of a cold-start problem, LightFM performs at least as well as the Matrix Factorisation Model. In addition, the author has released a Python implementation of LightFM[2] and made the source code available on Github[3].

LightFM incorporates matrix factorization, where users and items are represented as latent vectors (embeddings). However, just as in a collaborative model, these are entirely explained by functions (linear combinations) of embedding the content features that outlines each item or user. Each user u is defined by its features f_u. Similarly, each item i is defined by its features f_i. The latent representation of user u is given by the sum of its latent vectors of features:

$$q_u = \sum e_j^U \tag{3.6}$$

Which is the same for item i:

$$q_i = \sum e_j^I \tag{3.7}$$

Additionally, the bias term for user u and item i is given by the sum of the biases of the features:

$$b_u = \sum b_j^U \tag{3.8}$$

$$b_i = \sum b_j^I, \tag{3.9}$$

As a result, the prediction for user u and item i is the dot product of q_u and q_i adjusted by their feature biases b_u and b_i:

$$\hat{r}_{ui} = f(q_u \cdot q_i + b_u \cdot b_i) \tag{3.10}$$

LightFM allows us to use different ranking methods (loss functions) such as BRP (Bayesian Personalized Ranking), WARP (Weighted Approximate-Rank Pairwise Loss), and k-OS WARP. BRP is a pairwise ranking approach which samples the unfavourable rating for every favourable rating. For instance, for each user interaction, it samples an item with which the user has no interaction. The assumption here is that the user favours the item with which he or she has an interaction rather than the one without any interaction. [51] show theoretically and empirically that the BRP optimization method is vital for personalized ranking. Furthermore, two variances of stochastic gradient descent can be utilized in LightFM; namely, Adagrad [52] and Adadelta [53]. To utilize the LightFM model, we are required to build the user-item, item-feature, and, lastly, user-feature matrix.

[2]https://github.com/lyst/lightfm/
[3]https://github.com/lyst/lightfm-paper/

4

Experiments and Results

This section describes the results and the evaluation of the model. First, the experimental setup is described, and thereafter the results are delivered.

4.1 Experiments and Goals

LightFM is tested under distinct circumstances and the goal of each examination is explained.

- **Experiment 1** (dataset split experiment):
  The dataset is randomly split into training and test dataset, which is a common practice in machine learning settings. More specifically, we would like to understand how LightFM performs with and without different metadata attributes (item and user metadata). In this case, we begin by using no item or user metadata and continue to add more metadata attributes to the model incrementally:
  - LightFM (1): LightFM without any metadata as our baseline model.
  - LightFM (2): Baseline model item features.
  - LightFM (3): Baseline model user features.
  - LightFM (4): Baseline model with both item features and user features.
- **Experiment 2** (cold start experiment):
  Following the training set and test set in Experiment 1 (experiment with dataset randomly split). We do not make any changes to the test set while keeping at most i item interaction per user in the training set, $i = [1, 5, 10, 20, 38, 100, 200, 500, 2003]$. Recall that the median, mean and max of this number is 38, 99.9 and 2003, respectively, from the examination of our dataset. Correspondingly, we keep the test set unchanged while keeping at most u user interaction per item in the training set, $u = [1, 5, 10, 20, 30, 40, 50, 75, 100, 150, 200, 500, 653]$.

4.2 Evaluation Metrics

As for the evaluation metric, there are totally four metrics in LightFM modules containing evaluation functions appropriate for estimating the performance of a fitted LightFM model. The following elaborates on each evaluation metric in detail.

- `lightfm.evaluation.auc_score`: estimates the probability that a randomly chosen positive example has a higher score than a randomly chosen negative example. A perfect score is 1.0.

- `lightfm.evaluation.precision_at_k`: estimate the fraction of known positives in the first k positions of the ranked list of results. A perfect score is 1.0.

- `lightfm.evaluation.recall_at_k`: the number of positive items in the first k positions of the ranked list of results divided by the number of positive items in the test period. A perfect score is 1.0.

- `lightfm.evaluation.reciprocal_rank`: measures the reciprocal rank metric for a model, which is calculated as 1 / the rank of the highest ranked positive example. A perfect score is 1.0.

Although all evaluation metrics are accurate indicators of a worthy recommender system, for most recommendation scenarios, users are less likely to quit the platform just because they observe an item they are not interested in and are more likely to keep browsing until they find something interesting. Therefore, a recommender system in this specific scenario should focus more on recommending an exciting item to the user instead of paying too much attention to avoid making a few harmful recommendations.

4.3 Hyperparameter Selection

/ For each model selected from BRP, WARP and k-OS WARP loss functions, we select hyperparameters by using a grid-search on a validation set, with the number of k ranging from *k1* to *k2* and a learning schedule chosen between Adagrad and Adadelta. Finally, the iterations *epochs* is set to 100 for all blends.

One of the first insightful experiments concerning hyperparameter tuning compares the accuracy between the WARP (Weighted Approximate-Rank Pairwise) and BPR (Bayesian Personalised Ranking) losses. According to LightFM documentation, the WARP loss for implicit feedback illustrates a superior performance than popularized

BRP loss by a significant margin. To compare the loss function available, both models are performed with equivalent hyperparameters, and their accuracy across epochs is tested.

Epochs	*Train AUC*	*Val. AUC*	*Train Prec.@10*	*Val. Prec.@10*	*Train Rec.@10*	*Val. Rec.@10*	*Train Recip.*	*Val. Recip.*
1	0.946	0.951	0.217	0.061	0.048	0.047	0.359	0.162
5	0.969	0.971	0.251	0.070	0.058	0.057	0.385	0.182
10	0.976	0.976	0.272	0.076	0.061	0.062	0.415	0.190
15	0.980	0.979	0.284	0.079	0.069	0.066	0.446	0.203
20	0.982	0.981	0.299	0.082	0.073	0.070	0.456	0.208
25	0.983	0.982	0.302	0.081	0.075	0.068	0.472	0.211
30	0.984	0.982	0.310	0.085	0.077	0.073	0.476	0.213
35	0.985	0.983	0.319	0.089	0.079	0.077	0.477	0.218
40	0.986	0.984	0.321	0.087	0.081	0.076	0.490	0.218
45	0.986	0.984	0.321	0.088	0.080	0.076	0.492	0.221
50	0.987	0.985	0.332	0.090	0.083	0.080	0.504	0.219
55	0.987	0.985	0.331	0.091	0.086	0.080	0.504	0.229
60	0.988	0.986	0.327	0.089	0.084	0.077	0.498	0.229
65	0.988	0.986	0.335	0.091	0.087	0.081	0.511	0.232
70	0.988	0.986	0.333	0.090	0.088	0.082	0.505	0.225

Table 4.1: Evaluation metrics over epochs

4.4 Experimental Results and Analysis

4.4.1 Results of experiment 1

To recall, *experiment 1* examines the performance of LightFM with and without metadata attributes (item and user metadata). 4.2 illustrates the performance of the baseline model (Pure CF) relative to the other models with metadata.

- We can see that the item model outperforms other models in all evaluation metrics except in Recall@10, where the base model turns out to be the best model.
- The user model shows inferior results relative to other models in every evaluation metric, which suggests that proposing more metadata does not automatically enhance the accuracy of the suggestion. However, the introduction of item features shows a significant performance improvement relative to our base model. As the user features consist of *Region* and *Customer type*, we can conclude that

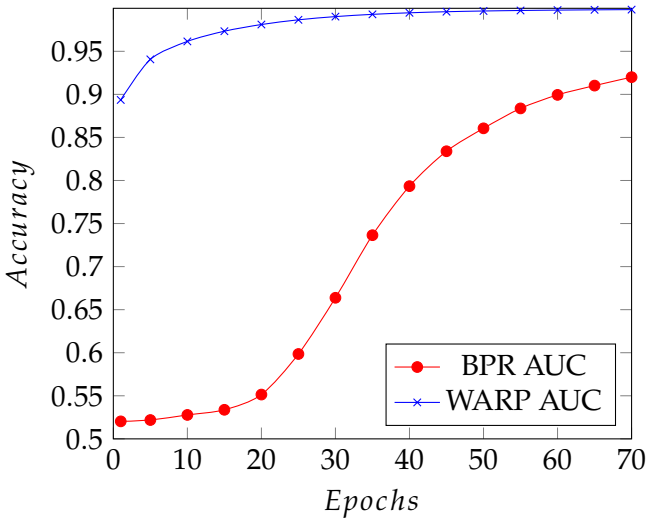

Figure 4.1: Accuracy of BRP and WARP function losses

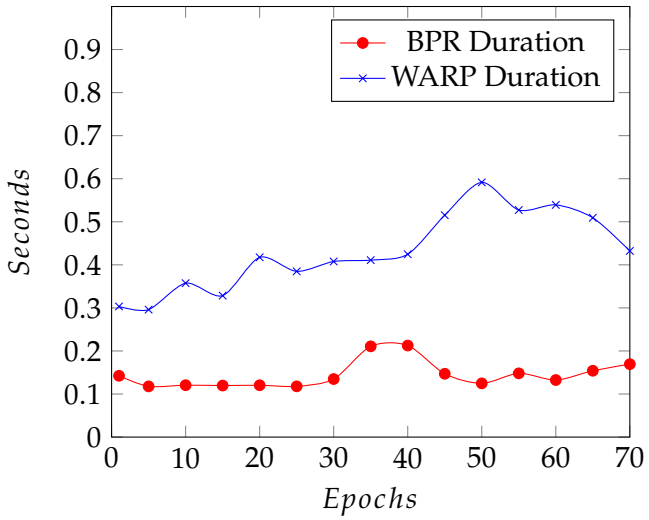

Figure 4.2: Duration for BRP and WARP in seconds

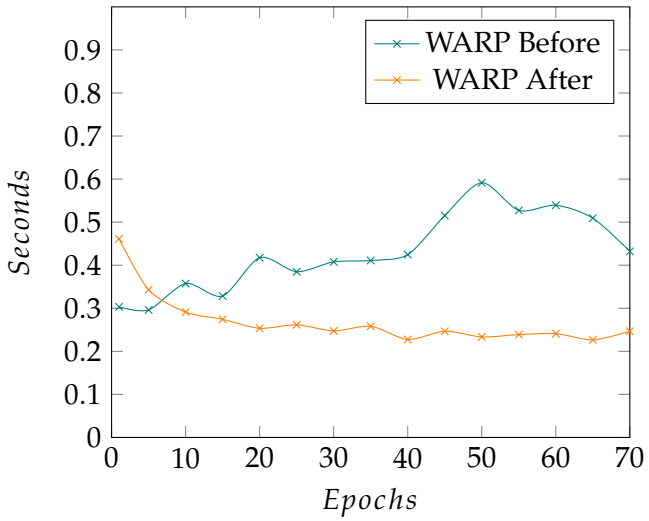

Figure 4.3: Duration for WARP after setting parameter

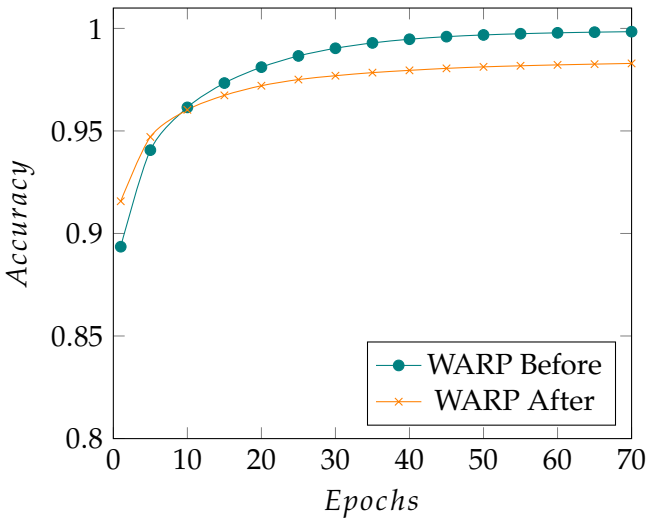

Figure 4.4: Accuracy for WARP after setting parameter

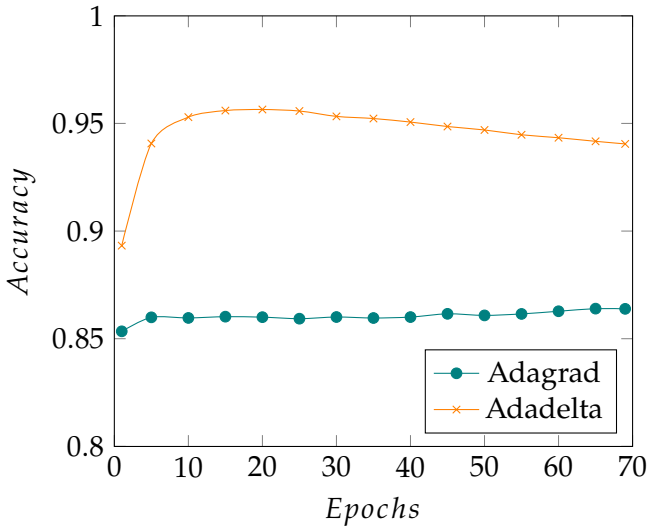

Figure 4.5: Accuracy Learning Rates: Adagrad and Adadelta

these features worsen the recommendations considerably and are not important indicators of user preference.

- As a consequence, the user features also prevent the hybrid model from better performance. We can conclude this statement by the fact that the base model performs better than the hybrid model in every single evaluation metric.

From the outcome, we are able to conclude that adding contextual data improves the recommendations made by the model. On the other hand, the wrong type of context could also easily prevent the hybrid model from performing inferior to the simple base model. Furthermore, this experiment demonstrates that by hybridizing our base model with item features, the value of AUC increases by 0.13%, the value of Precision@10 increases by 5.9%, the value of Reciprocal ranking increases by 1.3% yet the value of Recall@10 decreases by 3.1%. In figure 4.6, 4.7, 4.8 and 4.9 the test and validation results of every model is depicted as a bar chart.

4.4.2 Results of experiment 2: User cold-start

In our second experiment, we examine the outcome of the cold-start experiment in the case of a new user as well as in the case of a new item. Firstly, given that the most user interaction with an item is in 2003, we test a new user from 1 to 2003 interactions and observe the 'robustness' of the model in these situations. Secondly, given that the most item interaction with a user is 653, we test a new item from 1 to 653 interactions and observe the 'robustness' of the model in the presence of an item cold-start problem. Additionally, we observed that the item model performs much better than the base model in all evaluation metrics except Recall@10. Therefore, the item model will

Method	*Evaluation Metrics*	*Train*	*Validation*
Hybrid	AUC	0.981535	0.981202
Item model	AUC	0.986945	**0.982649**
Base model	AUC	0.984429	0.981283
User features	AUC	0.977270	0.970414
Hybrid model	Precision@10	0.320318	0.083189
Item model	Precision@10	0.380892	**0.097101**
Base model	Precision@10	0.386425	0.091684
User model	Precision@10	0.277218	0.071414
Hybrid model	Recall@10	0.076828	0.071767
Item model	Recall@10	0.113357	0.093350
Base model	Recall@10	0.112305	**0.096354**
User model	Recall@10	0.068693	0.059268
Hybrid model	Reciprocal ranking	0.477369	0.213976
Item model	Reciprocal ranking	0.577147	**0.240445**
Base model	Reciprocal ranking	0.572265	0.237265
User model	Reciprocal ranking	0.444769	0.189037

Table 4.2: All model compared

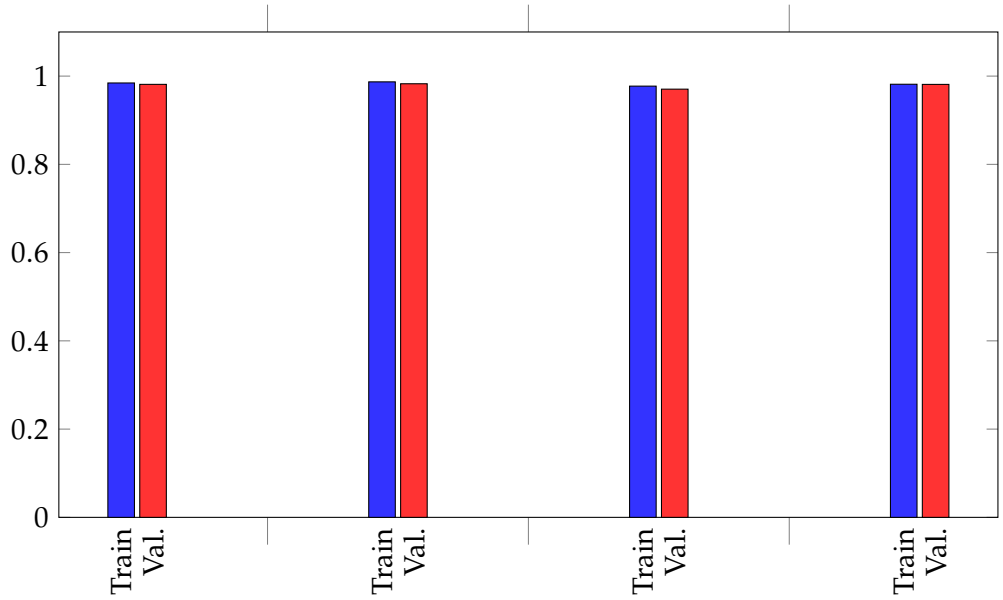

Figure 4.6: Accuracy of all models compared

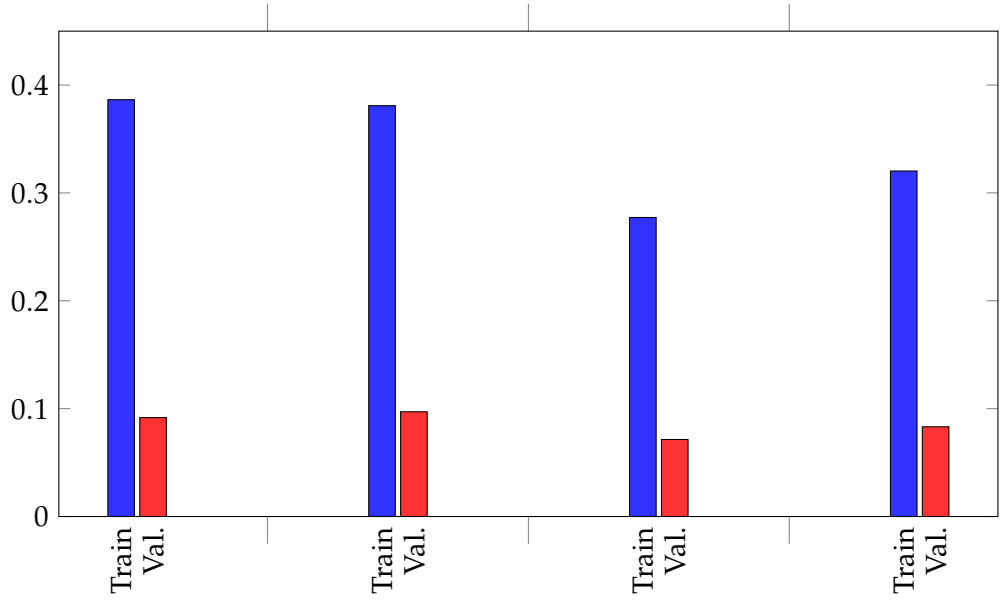

Figure 4.7: Precision@10 of all models compared

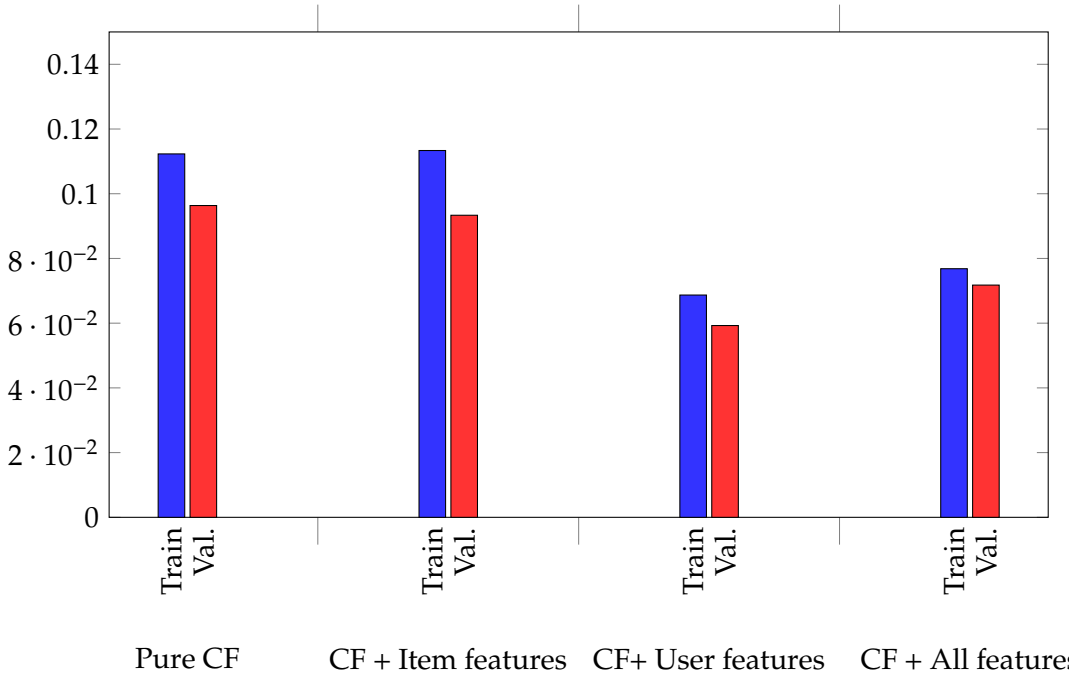

Figure 4.8: Recall@10 of all models compared

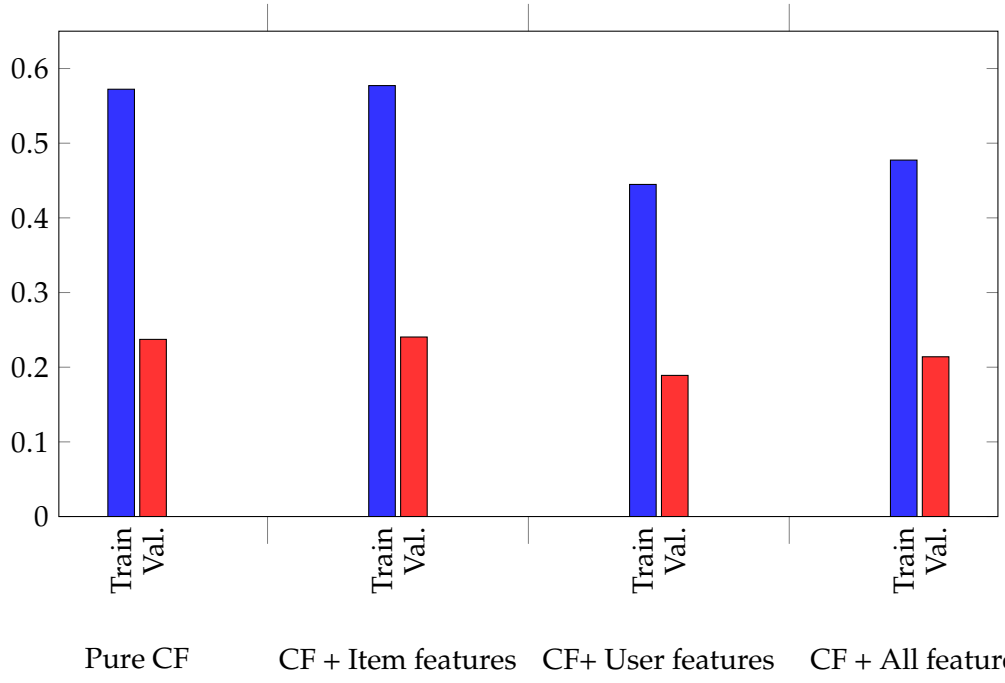

Figure 4.9: Reciprocal ranking of all models compared

be tested against the item and user cold-start problem and compared its 'robustness' against the base model.
In figure 4.10, we observe the AUC performance of the base model and the item model. The results show that the AUC of both models scales up relatively quickly after only a handful of interactions which implies that the LightFM model can make superior suggestions subsequent to a few interactions. Interestingly, the base model starts (1 interaction) of with a 0.62 AUC compared to 0.66 for the item model. The 6% increase shows that the item model handles the item cold-start problem better than the base model.
Similar to the AUC performance, in figure 4.11, we observe a swift scale-up for Precision@10 for the base model as well as the item model. In case of one interaction and five interactions, we discover a Precision@10 of 0.016 and 0.087, respectively. An important fact to notice is the diminishing score after the 100th interaction.
In regards to Recall@10 in figure 4.12, both models face a diminishing Recall after a few interactions. However, the item model prevents a drastic loss in Recall where the base model drops significantly relative to the item model. Interestingly, the base model outperforms the item model as the maximum number of user interactions reaches the end.
Lastly, in figure 4.13, in terms of Reciprocal Ranking, the base model and item model perform similarly, with the item model outperforming the base model by a small margin.

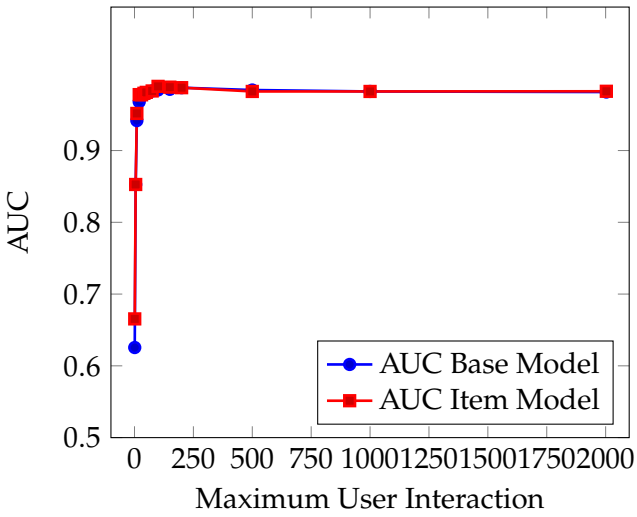

Figure 4.10: User Cold Start - AUC

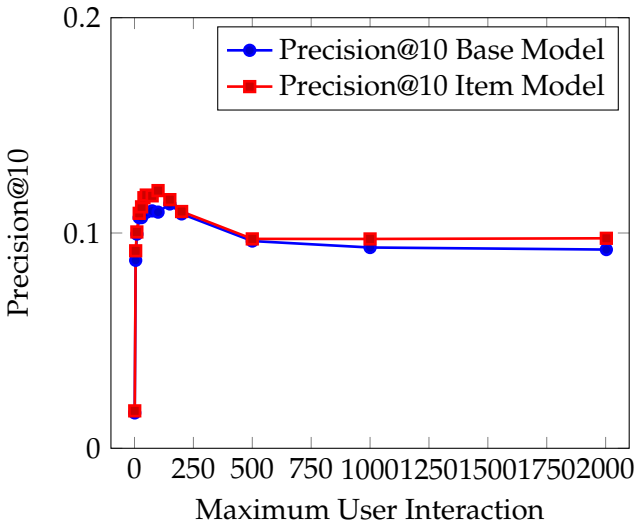

Figure 4.11: User Cold Start - Precision@10

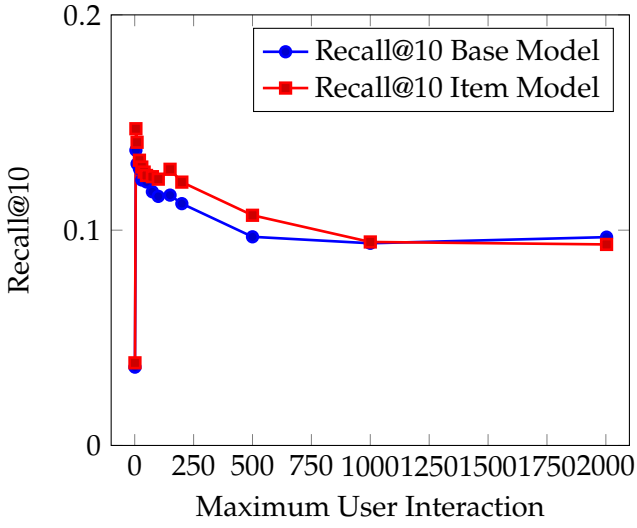

Figure 4.12: User Cold Start - Recall@10

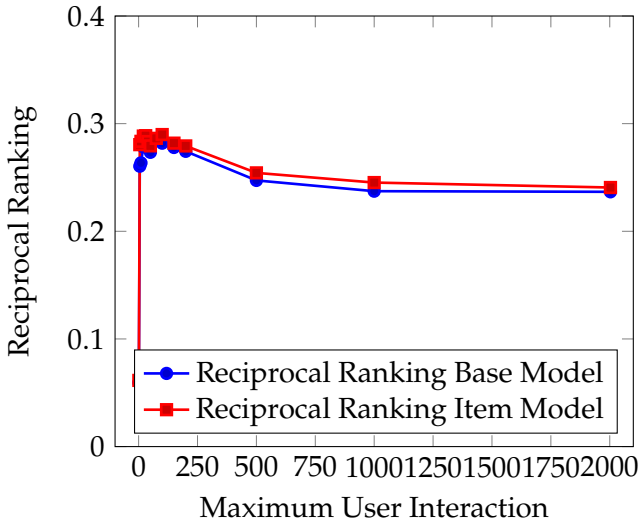

Figure 4.13: User Cold Start - Reciprocal Ranking

4.4.3 Results of experiment 2: Item cold-start

In the case of a user cold-start problem, the maximum interactions per item are limited to test the performance of the LightFM model in a case of a new item.
In this case, the most user per item in the training set were set to [1, 5, 10, 20, 30, 40, 50, 75, 100, 150, 200, 500, 653]. In figure 4.14, the AUC performance of the base model is similar to the item model. Yet, the item model clearly outperforms the base model with only 5 interactions per item, where the baseline model has an AUC of 0.57 and 0.95 for the first and fifth interactions. In contrast, the item model has an AUC of 0.63 and 0.97 for the same number of interactions. The AUC of both models increasingly becomes indistinguishable as the number of maximum item interactions increases. Moreover, the item model shows further superiority in the case of the limited number of item interactions (cold-start) which implies that the incorporation of item features and, thereby, utilizing metadata shows an improvement of the model in the presence of a cold-start problem.

4.5 Discussion

Despite the relative low accuracy achieved for Recall@10 and Precision@10,, the results for test dataset which ranges from 0.0717 up to 0.0971 seem promising compared with well-accepted and popular industry benchmark such as MovieLens 100K thoughtfully organized by a recommender system research lab at the University of Minnesota. For instance, Kula reported a precision@5 of 0.04 on the MovieLens 100k [49]. In another case, Heitman and Hayes reported precision at 5 between 0.03 and 0.09 on the MovieLens 100k dataset. In the section 4.3, the importance of choosing the right values

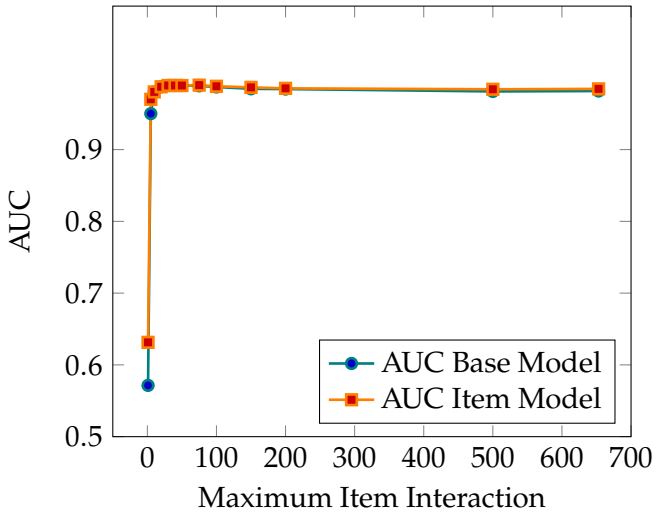

Figure 4.14: Item Cold Start Problem - AUC

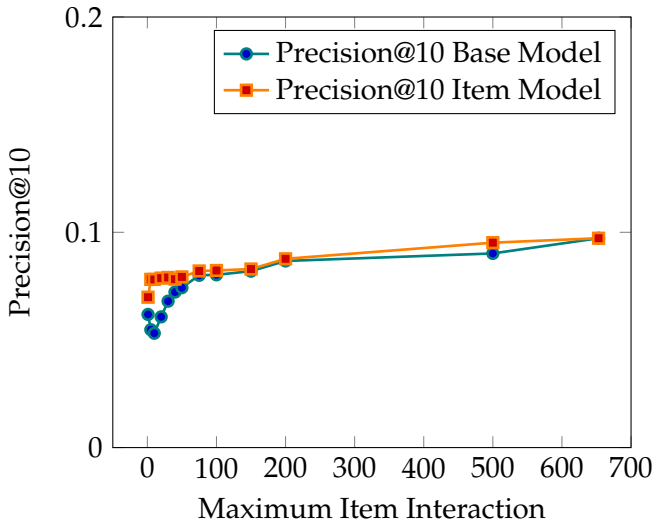

Figure 4.15: Item Cold Start Problem - Precision@10

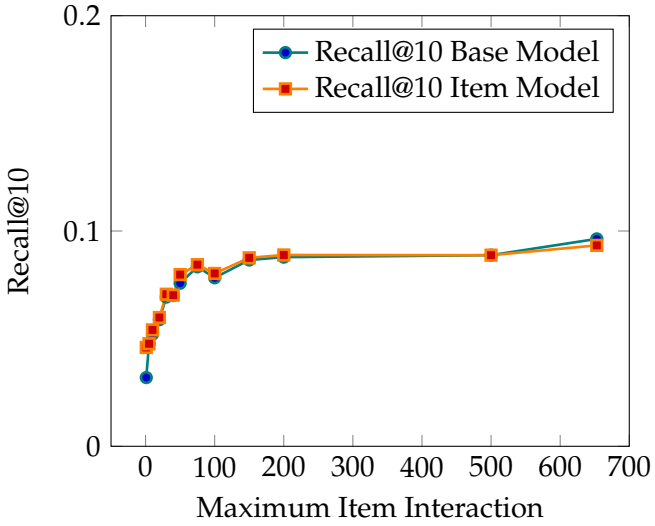

Figure 4.16: Item Cold Start Problem - Recall@10

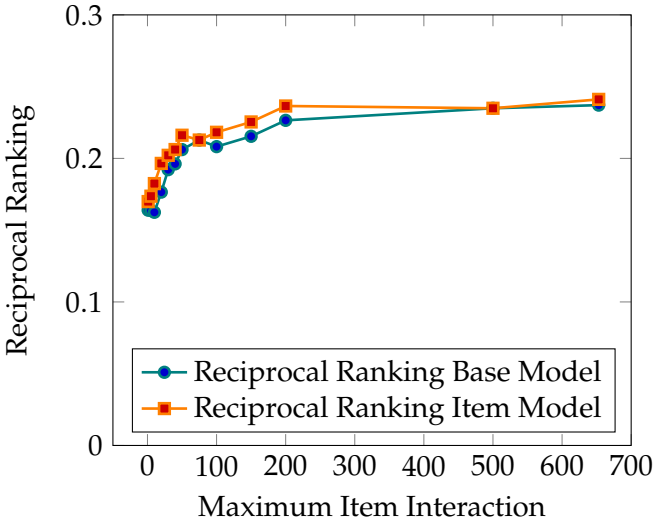

Figure 4.17: Item Cold Start Problem - Reciprocal Ranking

for constants and extent of regularization was shown. In presence of substandard values for hyperparamters, the models are expected to performance worse in regards to computation duration as well as accuracy. Whenever the user features, namely, *Region* and *Customer type*, were introduced to the model, as was the case for user model and hybrid model, the model achieved a worse accuracy than without user features which is contrary to the results from the original paper of Kula where the model accuracy benefited from introduction of user and item metadata [49]. The decrease in performance by introducing item features into the model could indicate to the low predictive power of the features selected. In our case, firstly, the *region* of the user seem to not be significant in terms of predicting users inclination and preferences and secondly, the *customer type* of the customer also does not reflect the preference of the customer. As a results, one could argue that the increased benefits of introducing contextual data depends on the case at hand and more importantly on the features themselves. In this setting, including item features increased the performance while including user features lowered the performance which suggests that the quality of description of an item or user is vital to make superior suggestions than the base model. In existence of a cold-start problem, introducing relevant metadata, such as item features in our case, enhances the model compared than without metadata. Despite the fact that by introducing the user and item metadata in conjunction the hybrid model performs worst than the base model, it might better for Magaloop to still choose to go with a hybrid model as it customer base still includes fresh acquired users coming to the platform. The small better performance could lead to higher new users retention and might be worth of the trade-off.

Alternatively, Magaloop could use customer profiling based on the days since registration of the user to utilize different recommendation model for different customers. An existing customer could benefit from the pure collaborative filtering due to existence of a warm setting and sufficient data while a new user could benefit from a hybrid model as it has shown to perform better in cold-start settings than the pure collaborative filtering model.

5

Conclusions and Future Work

This chapter concludes the thesis by going back to the research questions asked in the very first chapter and discussing possible future work that Magaloop can initiate to further improve the model built.

The main intention of this thesis was to research yet also discover the an encouraging technique to build a recommendation engine for Magaloop users. In context, Magaloop is a young organisation that finds itself still in a growth phase where new users are acquired and are onboarded to the platform. Therefore, recommender system is required to be able to handle a large set of freshly acquired user. The data generated by Magaloop consists of item purchases, i.e., user-item interactions which are distinguished into separate set of events as implicit feedback, i.e., putting an item to your basket or viewing a product in detail, and additional metadata on the users and the item. To achieve this objective, the LightFM model was chosen due to its ability to handle large dataset, addition of user and item meta data and promising accuracy in presence of cold-start problem. Therefore the following research questions were formulated:

1. **Research Q1:** Could a mixture of implicit data on users and contextual data on items increase the performance beyond the performance achieved with a pure collaborative filtering model?

2. **Research Q2:** Does a hybridized model with a mixture of implicit data on users and contextual data on items aid the recommender system to increase the accuracy in case of user- and item cold-start problem?

To explore these research questions, a literature review was written to discover the fundamental techniques that could be a potential solution for Magaloop. Based on the literature review as well as in a more practical approach, matrix factorization models have been widely utilised because how effectively they can handle a sparse user-item interaction data. Moreover, matrix factorization also is able to incorporate latent features of items and users to make recommendations which has the potential to be beneficial in a cold-start setting where the user-item interaction matrix is sparse.

In this experiment, four distinct version of matrix factorization models were explored: LightFM as pure collaborative filtering (base model), LightFM with item features, LightFM with user features and lastly, hybrid LightFM with a combination of user and item metadata. The results show that, in this case, the item features allow the model to make better and more accurate recommendations while the user features function as the opposite. In fact, the base model shows far superior accuracy than the hybrid model, which includes only user features.

5.1 Future Work

As the data given is recent actual data, an online accuracy evaluation can be an additional evaluation metric beside the offline accuracy evaluations given by the LightFM model. However, an online evaluation could have been possible through an earlier model deployment.
Moreover, although the actual transactional purchase behaviour of the user possesses the majority of the weight in determining the numeric rating, an extensive study could be made to discover the importance of the other included implicit data generated from Google Analytics. In addition, the existence of a time-decay could also be examined as well as different time-decay functions which determine how fast the significance of a rating fades as time passes.
Lastly, the importance of each item feature and user feature could be examined in individual cases. Instead of using item features and useful features as a whole, an incremental addition could indicate which particular feature has the most positive or negative impact on the accuracy and overall performance of the model.

Bibliography

[1] J. M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/master/template.pdf (cit. on p. iii).

[2] E. Shafir, I. Simonson, and A. Tversky. "Reason-based choice". In: *Cognition* 49.1 (1993), pp. 11–36. ISSN: 0010-0277. DOI: https://doi.org/10.1016/0010-0277(93)90034-S. URL: https://www.sciencedirect.com/science/article/pii/001002779390034S (cit. on p. 1).

[3] B. Schwartz. *The Paradox of Choice: Why More is Less*. Harper Perennial, 2005 (cit. on p. 1).

[4] S. S. Iyengar and M. R. Lepper. "When Choice is Demotivating: Can One desire Too much of a Good Thing?" In: *Journal of Personality and Social Psychology* 79.6 (2000), pp. 995–1006 (cit. on p. 1).

[5] P. Kotler and K. L. Keller. *Marketing Management*. Pearson, 2016 (cit. on p. 1).

[6] D. Amanah and D. A. Harahap. "Online Purchasing Decision of College Students in Indonesia". In: *International Journal of Latest Engineering Research and Applications* 3 (2018), pp. 05–15 (cit. on p. 2).

[7] J. Jacoby, D. E. Speller, and C. K. Berning. "Brand Choice Behavior as a Function of Information Load: Replication and Extension". In: *Journal of Consumer Research* 1.1 (1974), pp. 33–42 (cit. on p. 2).

[8] A. Sela, J. Berger, and W. Liu. "Variety, Vice, and Virtue: How Assortment Size Influences Option Choice". In: *Journal of Consumer Research* 35.6 (2009), pp. 941–951 (cit. on p. 2).

[9] G. Adomavicius and A. Tuzhilin. "Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions". In: *IEEE Transactions on Knowledge and Data Engineering* 17.6 (2005), pp. 734–749. DOI: 10.1109/TKDE.2005.99 (cit. on pp. 2, 11, 13).

[10] J. B. Schafer, J. A. Konstan, and J. Riedl. "Recommender systems in e-commerce". In: *Association for Computing Machinery* (1999), pp. 158–166. DOI: 10.1145/336992.337035 (cit. on p. 2).

[11] A. Ansari, S. Essegaier, and R. Kohli. "Internet recommendation systems". In: *Journal of Marketing Research* 37.3 (2000), pp. 363–375 (cit. on p. 2).

[12] F. Ricci, L. Rokach, and B. Shapira. *Recommender Systems Handbook, chapter Recommender Systems: Introduction and Challenges*. Springer New York, NY, 2015, pp. 1–34 (cit. on pp. 2, 3).

[13] J. Karlgren. "An Algebra for Recommendations". In: *Department of Computer and Systems Sciences The Royal Institute of Technology and Stockholm University Electrum* 179 (1990) (cit. on p. 2).

[14] G. Linden, B. Smith, and J. York. "Amazon.com recommendations: item-to-item collaborative filtering". In: *IEEE Internet Computing* 7.1 (2003), pp. 76–80. DOI: 10.1109/MIC.2003.1167344 (cit. on p. 2).

[15] J. Davidson et al. "The YouTube Video Recommendation System". In: *Proceedings of the Fourth ACM Conference on Recommender Systems*. RecSys '10. Barcelona, Spain: Association for Computing Machinery, 2010, 293–296. ISBN: 9781605589060. DOI: 10.1145/1864708.1864770. URL: https://doi.org/10.1145/1864708.1864770 (cit. on p. 2).

[16] C. A. Gomez-Uribe and N. Hunt. "The Netflix Recommender System: Algorithms, Business Value, and Innovation". In: *ACM Trans. Manage. Inf. Syst.* 6.4 (2016). ISSN: 2158-656X. DOI: 10.1145/2843948. URL: https://doi.org/10.1145/2843948 (cit. on p. 2).

[17] E. Brynjolfsson, Y. J. Hu, and M. D. Smith. "Brynjolfsson, Erik and Hu, Yu Jeffrey and Smith, Michael D., Consumer Surplus in the Digital Economy: Estimating the Value of Increased Product Variety at Online Booksellers". In: *Association for Computing Machinery* 49.11 (2003), 1580–1596. DOI: http://dx.doi.org/10.2139/ssrn.400940 (cit. on p. 3).

[18] N. Barbieri, G. Manco, and E. Ritacco. "Probabilistic Approaches to Recommendations". In: *Synthesis Lectures on Data Mining and Knowledge Discovery* 5.2 (2014), pp. 1–197. DOI: https://doi.org/10.1007/978-3-031-01906-7 (cit. on p. 3).

[19] C. C. Johnson. *Logistic Matrix Factorization for Implicit Feedback Data*. 2014 (cit. on p. 3).

[20] C. Anderson. *The Long Tail: Why the Future of Business Is Selling Less of More*. Hyperion, 2006, pp. 1–256 (cit. on p. 3).

[21] P.-T. Chen and H.-P. Hsieh. "Personalized mobile advertising: Its key attributes, trends, and social impact". In: *Technological Forecasting and Social Change* 79.3 (2012), pp. 543–557. ISSN: 0040-1625. DOI: https://doi.org/10.1016/j.techfore.2011.08.011. URL: https://www.sciencedirect.com/science/article/pii/S0040162511001788 (cit. on p. 3).

[22] P. Smutkupt, D. Krairit, and V. Esichaikul. "Mobile marketing: Implications for marketing strategies". In: *International Journal of Mobile Marketing* 5 (2010-01), pp. 126–139 (cit. on p. 3).

[23] J. Zhang and M. Wedel. "The Effectiveness of Customized Promotions in Online and Offline Stores". In: *Journal of Marketing Research* 46.2 (2009), pp. 190–206. DOI: 10.1509/jmkr.46.2.190. eprint: https://doi.org/10.1509/jmkr.46.2.190. URL: https://doi.org/10.1509/jmkr.46.2.190 (cit. on p. 3).

[24] M. Reiß and M. Koser. "From Mass Customization to Mass Personalization A New Competitive Strategy in E-Business". In: *Trendberichte zum Controlling: Festschrift für Heinz Lothar Grob*. Ed. by F. Bensberg, J. v. Brocke, and M. B. Schultz. Heidelberg: Physica-Verlag HD, 2004, pp. 285–310. ISBN: 978-3-7908-2708-8. DOI: 10.1007/978-3-7908-2708-8_15. URL: https://doi.org/10.1007/978-3-7908-2708-8_15 (cit. on p. 3).

[25] M. Kaminskas et al. "Product Recommendation for Small-Scale Retailers". In: *E-Commerce and Web Technologies*. Ed. by H. Stuckenschmidt and D. Jannach. Cham: Springer International Publishing, 2015, pp. 17–29. ISBN: 978-3-319-27729-5 (cit. on pp. 3, 6).

[26] G. Guo. "Resolving Data Sparsity and Cold Start in Recommender Systems". In: *User Modeling, Adaptation, and Personalization*. Ed. by J. Masthoff et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 361–364. ISBN: 978-3-642-31454-4 (cit. on p. 4).

[27] C. Desrosiers and G. Karypis. "A Comprehensive Survey of Neighborhood-based Recommendation Methods". In: *Recommender Systems Handbook*. Ed. by F. Ricci et al. Boston, MA: Springer US, 2011, pp. 107–144. ISBN: 978-0-387-85820-3. DOI: 10.1007/978-0-387-85820-3_4. URL: https://doi.org/10.1007/978-0-387-85820-3_4 (cit. on pp. 5, 11).

[28] Y. Hu, Y. Koren, and C. Volinsky. "Collaborative Filtering for Implicit Feedback Datasets". In: *2008 Eighth IEEE International Conference on Data Mining*. 2008, pp. 263–272. DOI: 10.1109/ICDM.2008.22 (cit. on p. 6).

[29] D. Jannach et al. *Recommender Systems: An Introduction*. Cambridge University Press, 2010. DOI: 10.1017/CBO9780511763113 (cit. on pp. 6, 8).

[30] D. W. Oard and J. Kim. "Implicit Feedback for Recommender Systems". In: 1998 (cit. on p. 6).

[31] D. Kelly and J. Teevan. "Implicit Feedback for Inferring User Preference: A Bibliography". In: *SIGIR Forum* 37.2 (2003), 18–28. ISSN: 0163-5840. DOI: 10.1145/959258.959260. URL: https://doi.org/10.1145/959258.959260 (cit. on p. 6).

[32] D. Goldberg et al. "Using Collaborative Filtering to Weave an Information Tapestry". In: *Commun. ACM* 35.12 (1992), 61–70. ISSN: 0001-0782. DOI: 10.1145/138859.138867. URL: https://doi.org/10.1145/138859.138867 (cit. on pp. 7, 9).

[33] C. C. Aggarwal. *Recommender systems*. en. 1st ed. Basel, Switzerland: Springer International Publishing, 2016-03 (cit. on pp. 7–9, 11–15, 21).

[34] M. Chevalier et al. "Information Retrieval and Folksonomies together for Recommender Systems". In: *E-Commerce and Web Technologies*. Ed. by C. Huemer and T. Setzer. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 172–183. ISBN: 978-3-642-23014-1 (cit. on p. 7).

[35] J. L. Herlocker, J. A. Konstan, and J. Riedl. "Explaining Collaborative Filtering Recommendations". In: *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*. CSCW '00. Philadelphia, Pennsylvania, USA: Association for Computing Machinery, 2000, 241–250. ISBN: 1581132220. DOI: 10.1145/358916.358995. URL: https://doi.org/10.1145/358916.358995 (cit. on pp. 7, 11).

[36] B. Sarwar et al. "Item-Based Collaborative Filtering Recommendation Algorithms". In: *Proceedings of the 10th International Conference on World Wide Web*. WWW '01. Hong Kong, Hong Kong: Association for Computing Machinery, 2001, 285–295. ISBN: 1581133480. DOI: 10.1145/371920.372071. URL: https://doi.org/10.1145/371920.372071 (cit. on pp. 8, 9).

[37] Y. Koren, R. Bell, and C. Volinsky. "Matrix Factorization Techniques for Recommender Systems". In: *Computer* 42.8 (2009), pp. 30–37. DOI: 10.1109/MC.2009.263 (cit. on p. 9).

[38] Y. Hu, Y. Koren, and C. Volinsky. "Collaborative Filtering for Implicit Feedback Datasets". In: *2008 Eighth IEEE International Conference on Data Mining*. 2008, pp. 263–272. DOI: 10.1109/ICDM.2008.22 (cit. on p. 10).

[39] X. Luo et al. "An Efficient Non-Negative Matrix-Factorization-Based Approach to Collaborative Filtering for Recommender Systems". In: *IEEE Transactions on Industrial Informatics* 10.2 (2014), pp. 1273–1284. DOI: 10.1109/TII.2014.2308433 (cit. on p. 11).

[40] P. Lops, M. de Gemmis, and G. Semeraro. "Content-based Recommender Systems: State of the Art and Trends". In: *Recommender Systems Handbook*. Ed. by F. Ricci et al. Boston, MA: Springer US, 2011, pp. 73–105. ISBN: 978-0-387-85820-3. DOI: 10.1007/978-0-387-85820-3_3. URL: https://doi.org/10.1007/978-0-387-85820-3_3 (cit. on pp. 12, 13).

[41] D. M. Blei, A. Y. Ng, and M. I. Jordan. "Latent Dirichlet Allocation". In: *J. Mach. Learn. Res.* 3.null (2003), 993–1022. ISSN: 1532-4435 (cit. on p. 12).

[42] C. Musto et al. "Word Embedding Techniques for Content-based Recommender Systems: An Empirical Evaluation". In: *RecSys Posters*. 2015 (cit. on p. 12).

[43] B. Kanagal et al. "Supercharging Recommender Systems Using Taxonomies for Learning User Purchase Behavior". In: *Proc. VLDB Endow.* 5.10 (2012), 956–967. ISSN: 2150-8097. DOI: 10.14778/2336664.2336669. URL: https://doi.org/10.14778/2336664.2336669 (cit. on p. 14).

[44] R. Zhang et al. "Collaborative Filtering for Recommender Systems". In: *2014 Second International Conference on Advanced Cloud and Big Data* (2014), pp. 301–308 (cit. on p. 14).

[45] R. Burke. "Hybrid Recommender Systems: Survey and Experiments". In: *User Modeling and User-Adapted Interaction* 12 (2002), pp. 331–370 (cit. on p. 15).

[46] L. Hasan, A. Morris, and S. Probets. "Using Google Analytics to Evaluate the Usability of E-Commerce Sites". In: *Human Centered Design*. Ed. by M. Kurosu. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 697–706. ISBN: 978-3-642-02806-9 (cit. on p. 17).

[47] S. Fernandes and J. Bernardino. "What is BigQuery?" In: *Proceedings of the 19th International Database Engineering amp; Applications Symposium*. IDEAS '15. Yokohama, Japan: Association for Computing Machinery, 2015, 202–203. ISBN: 9781450334143. DOI: 10.1145/2790755.2790797. URL: https://doi.org/10.1145/2790755.2790797 (cit. on p. 18).

[48] Y. Ding and X. Li. "Time Weight Collaborative Filtering". In: *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*. CIKM '05. Bremen, Germany: Association for Computing Machinery, 2005, 485–492. ISBN: 1595931406. DOI: 10.1145/1099554.1099689. URL: https://doi.org/10.1145/1099554.1099689 (cit. on p. 21).

[49] M. Kula. "Metadata Embeddings for User and Item Cold-start Recommendations". In: *Proceedings of the 2nd Workshop on New Trends on Content-Based Recommender Systems co-located with 9th ACM Conference on Recommender Systems (RecSys 2015), Vienna, Austria, September 16-20, 2015*. Ed. by T. Bogers and M. Koolen. Vol. 1448. CEUR Workshop Proceedings. CEUR-WS.org, 2015, pp. 14–21. URL: http://ceur-ws.org/Vol-1448/paper4.pdf (cit. on pp. 24, 38, 41).

[50] S. Rendle. "Factorization Machines". In: *2010 IEEE International Conference on Data Mining*. 2010, pp. 995–1000. DOI: 10.1109/ICDM.2010.127 (cit. on p. 24).

[51] S. Rendle et al. "BPR: Bayesian Personalized Ranking from Implicit Feedback". In: *CoRR* abs/1205.2618 (2012). arXiv: 1205.2618. URL: http://arxiv.org/abs/1205.2618 (cit. on p. 25).

[52] J. Duchi, E. Hazan, and Y. Singer. "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization". In: *Journal of Machine Learning Research* 12.61 (2011), pp. 2121–2159. URL: http://jmlr.org/papers/v12/duchi11a.html (cit. on p. 25).

[53] M. D. Zeiler. *ADADELTA: An Adaptive Learning Rate Method*. 2012. DOI: https://doi.org/10.48550/arxiv.1212.5701. URL: https://arxiv.org/abs/1212.5701 (cit. on p. 25).