Product Recommendation System

Schoinas Ioannis

SID: 3306160008

Supervisor: Assist. Professor Dr. Christos Tjortjis

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Mobile and Web Computing

Abstract

This dissertation is the final part required for the completion of the MSc in Mobile and Web Computing at the International Hellenic University. The main purpose is the development of a product recommendation system based solely on implicit feedback. The proposed algorithm incorporates technologies that include collaborative filtering with matrix factorization and association rule mining.

The proposed methodology implements a hybrid recommendation algorithm in such a way that is able to provide recommendations in multiple ways as well as use them to increase its accuracy. Moreover it includes implementation of methods for addressing data sparsity, an important issue for recommendation systems.

In addition, it is implemented a relatively new approach to increase the accuracy of matrix factorization algorithms via initialization of factor vectors, which as far as we know is tested for the first time an implicit model-based collaborative filtering approach.

The evaluation of the methodology shows that the implemented methods are promising and their implementation in real world scenarios could offer personalization and its benefits to customers and shop owners.

I would also like to thank my supervisor Dr. Christos Tjortjis for the guidance and valuable ideas that was necessary during the elaboration of this dissertation.

Contents

1 Introduction

It is evident that the pace that technology advances have been increased over the last decades. Scientific discoveries and technological growth introduced to people a huge variety of options and possibilities. One of the most important advantages that technology offers is the direct and easy access to information. Nowadays access to vast networks of information is easy and people can be informed about almost anything they desire.

Even though ease of access provided people with the ability to acquire the needed information, they are now facing a new obstacle: this of easily finding what they need. On one hand, information abundance covers the majority of needs but on the other hinders accessibility to information truly valuable to the user. The term that describes this phenomenon is “Information Overload”. Often users are presented with seemingly similar information to their inquiry but irrelevant to their actual needs, rendering this way the discovery of the desired knowledge a difficult task.

Continuous expanding of information overload necessitated the development of systems that aim to alleviate such problems. Such systems were introduced in order to filter or retrieve the desired information. Recommendation systems is an example. Recommenders aim to filter out all the unnecessary and irrelevant information and present those that fit the user’s needs. This way the user is relieved of the burden of discovering what he needs making this way information truly accessible.

1.1 Definitions

The information overload problem has been created due to the increased volume and availability of information. It describes the difficulty that today’s users face in discovering a specific piece of

information. For the purpose of tackling the problem numerous algorithms from different approaches have been developed.

According to [1] different approaches fall into those categories. Hyper-textual links is a form of organizing information and utilizing hyper-text links to guide user to the desired documents or items. Categorization aims to assign any document or item to categories and provide users with access to these. Retrieval and Filtering Systems generally aim to understand the relevance degree of the data against the user inquiries and preferences.

A retrieval system has the purpose of selecting relevant information from dataset of fixed size. Moreover they are designed to serve user inquiries and satisfy a specific need of the user each time. On the other hand a filtering system aims to filter out irrelevant information, as the name implies, usually from a stream of data. Another contrast to retrieval systems is that they use profiling techniques to infer user interests based on present and past user behaviour.

Recommender systems (RSs) fall in the category of information filtering approaches. The term describes the software tools and techniques that are utilized in order to recommend items of interest and value to the user [2].

It was observed that people take into consideration the opinion of their social environment in order to decide upon buying items. For example people are influenced and often rely on reviews regarding the products they intend to buy from an online store or follow suggestions from their friends on what books to read and movies to watch. First recommendation algorithms tried to model this behaviour and used communities for creating their recommendations.

Initial systems were known as Collaborating Filtering [3]. First communities of users were formed based on their interests. Suggestions to a user were formed based from those items that the other users of his community preferred.

As items can be considered a variety of things; from books, websites or products to social profile pages and even job positions. For example a news blog can use RSs to recommend articles based on the readers preference of topics. In the same fashion a social network could recommend friends with interests in similar topics to a user.

Different RSs can be very diverse depending on the domain, approach in creating the recommendations, information that used to extract them and other factors. Different domains offer different possibilities, limitations or data that the system could exploit and thus play a crucial role in the design of such a system. It is obvious that when we consider a book recommender system we could use the author or the title of a book as well as possible user ratings to extract knowledge from. On the other hand author and ratings might not be present when considering a recommendation system for an online store.

Another aspect that should be considered during the designing of such a system is the source of the information and by what means these are acquired. Input data are of two types either explicit or implicit. Explicit input data are the ones provided from the user and implicit ones are those that come from monitoring user behaviour such as past purchases.

Also different case studies have different functional requirements. A recommender system that aims to provide real time suggestions depends heavily on the execution time of the algorithm. This factor excludes by definition approaches with high cost in execution time.

Due to the variety of domains, the source and the information availability while also any other possible limitations and restrictions that might exist, not all algorithms are equally useful for each and every case. There is not one algorithm able to provide the best recommendations compared for any case.

1.2 Recommendation Systems and Benefits

Recommendation Systems proved to be a valuable tool for the users as well as the systems that implemented them [33,34]. On one hand they effectively deal with the information overload problem and in addition they offer personalization to a web site or service which appeals to the end user enhancing his experience.

1.2.1 Well-known Implementations

The importance of Recommendation Systems can easily be seen by their various existing implementations in well-known websites.

Amazon.com

[4] Amazon.com created its recommendation system in 1998 implementing an item-based collaborative filtering algorithm. Based on users' past behavior, context and ratings it manages to offer a unique customer experience.

The system is embedded and utilized in multiple ways throughout the whole shopping experience. It starts even from the beginning when a number of products is recommended based on previously seen items. It recommends products that were bought together with the product being watched. Furthermore it recommends items that might be related to the product in order to discover unknown interests. Additionally it has a section where recommendations extracted based on user purchase history.

The company met a big growth while at the same time a big part of its sales comes from the recommendations.

YouTube

As described in [5] YouTube also has developed its own recommendation engine. YouTube's RS is a top-N recommender and it aims to provide personalized recommendations based on user recent behavior. Moreover the second goal of the system is to promote through recommendations the wide range of available content that offers. Some of the main challenges that the system had to face was the lack of meta-data associated with the videos while also the short user interactions that made the discovery of user intent a difficult task. Moreover constant refreshment of the suggestions is required due to the short life cycle of the videos from upload to becoming viral. The RS gathers data from a variety of sources. It uses all the content information of the videos and at the same time considers user input either it explicit or implicit.

Lastly YouTube considers the way the recommendations are being presented to its users as highly important. It offers explanations as to why they are recommended and advances personalization even more by allowing its users to control where and how many recommendations should appear.

Netflix

Another well-known RS is Netflix. Netflix does not create recommendations based on a single algorithm, but rather utilizes a number of different ones regarding the use case they were designed for [6]. Taking into account the fact that the longer time a user spends in searching for a show the more likely it is for him to stop using the service. Company focused its system in being able to provide suggestions that will draw the attention of the user in the top of the list of shows. On top of that it should be made clear to the user why each show is being recommended.

At this point it should be noted that in its effort to improve the RS the company organized a competition. Netflix Prize 2009 offered 1 million dollars to the team with the higher prediction accuracy algorithm. As a result many algorithms are being used in production even today [7]

As already noted the system is composed by a number of algorithms. Personalized Video Ranker (PVR) is used to offer personalization and is responsible to define the order in which videos of a specific genre appear. Top-N Video Ranker is responsible to find the best suggestions regarding the user's preference regardless of genre. Essentially Netflix also uses different algorithms for ordering shows in the rows of Trending rows and Continues watching. Moreover Video-Video similarity is used to suggest show in a "Because You Watched" section.

1.2.2 Benefits

It is evident that RS do not only benefit the users of a system; a website or a company can have a great deal of profit by implementing an efficient recommender system.

Revenue

Recommender systems not only aim to provide interesting items to its users, but also aim to increase the sales via e-commerce. By modelling each user's needs and interests it is able to guide or promote those items that are more likely to be consumed by the specific user. Moreover they can be implemented in such a way to aim to increase cross-sales. Suggestions could be made based on the items that the user intents to buy and find possible complementary items. It is not a mystery why RSs are strongly associated with higher conversion rates.

Promote unpopular products

RS knowing the interests of a user and is able to provide diverse suggestions. This way the provider can advertise those products they wish to promote to the appropriate users.

User Satisfaction

RSs engage users with their recommendations since they draw the attention of the user to more items without needing him to search. With interesting and relevant recommendations the user will have an enjoyable experience. It is more likely for the user to prefer the service again in the future.

Loyalty

RS is able to offer enhanced personalization to the user which is improved the more the user visits the system. A relation between the website and the user is formed since the more tailored to the needs to the customer a system becomes the more satisfied the user will be. This leads to increase of loyalty, because continuous satisfied users value the experience they get and keep returning to the sites they enjoy to interact with.

Provide Reports

Furthermore RS can be used as valuable tool to extract reports and monitor user behaviour. This can give a leverage to the service provider since it can see the impact of various promotional campaigns or other actions. In addition knowing the interest of customers and users is a valuable

knowledge that can be used in other ways to provide profit for the company. On point example of that is Netflix which by taking into account subscribers preferences is able to maintain an efficient catalogue size. Before buying new content, the company calculates the actual worth of the content against its subscribers' interests.

1.3 The Problem

Recommender systems are proved to be a valuable feature which enhances user experience as well as an important tool in the hands of the service provider. E-commerce sites are a domain that could greatly benefit from the advantages a Product Recommendation System has to offer. These websites though must take into consideration all the factors that affect user experience in order to increase revenue. Users often use e-commerce systems because they can do their shopping with comfort and ease. It is evident that the design of an ecommerce website should be done in such a way that it does not harass users with constant questions about their preferences and force them to provide feedback.

Additionally the system needs to make the whole process of shopping easy and fast, otherwise it risks losing sales and loyalty. It obvious that these implementations suffer from the lack of explicit input and feedback from the users. Even though recommender systems have been extensively researched, most of the times research focused on use cases where the user provided input explicitly via a ranking system. There is still more room to design and enhance recommendation algorithms that are based solely on implicit information gathered by monitoring user behavior. This way ecommerce systems can offer personalized recommendations without deteriorating the experience of its users. Such implicit information, even though they are easily gathered from monitoring users browsing behavior and exploiting his purchase history, often concern a small number of items rather than the whole set offered by the provider. Thus, an additional challenge that should be addressed is data sparsity that these implementations suffer from.

1.4 Scope of the Dissertation

The purpose of this study is to examine implementations of various recommendation algorithms. It aims at developing a Product Recommendation Engine suitable for e-commerce websites, thus it will be based solely on implicit input. As input we use a combination of implicit sources. This combination consists of browsing user behavior, add to cart actions as well as their purchase history. The implemented algorithm is a Matrix Factorization algorithm which incorporates confidence and it based on the work of [8]. Additionally, the data sparsity problem is being addressed by utilizing association rule mining.

1.5 Dissertation Outline

In first chapter we present an overview of the recommender systems and their purpose. Furthermore we showcase the impact those systems had the latest years and how well-known companies integrated them to their systems. Also we present some key benefits they can offer to their service providers. Lastly we define the problem we will address, present the scope of the dissertation and provide a description for every chapter of the dissertation

In the second chapter we present the basic concepts behind recommender systems and describe various input sources, followed by the main categories of recommendation algorithms with an analysis of each approach to its key concepts and the idea that it aims to model. Furthermore, we describe various evaluation ways used to measure the overall efficiency of such systems.

In the third chapter we conduct a literature review of recommender implementations. We make a comparison to our approach and in which way some of these ideas could beneficially impact a system like ours. Most of the literature review concerns implicit feedback systems, but also algorithms that exploit explicit input as well as a combination of those.

In the fourth chapter we provide description and details of the design for our proposed system. The idea we aim to model is discussed and finally we also present flow charts of the system. In

the fifth chapter we demonstrate the implementation of our system in Python. We describe each function designed and its purpose. Additionally we give brief overview of the libraries used.

In the sixth chapter we present the evaluation of our system as well as conclusions resulted from our research. Finally we propose some ideas for future work in order to improve the efficiency of the specific system and discuss promising concepts and paths that could be modeled in new designs of a recommendation algorithm.

2 Core Concepts and techniques

2.1 Feedback

Recommender systems are in essence algorithms that process data in order to predict user interest regarding items. The data that these systems process are categorized in two categories based on the way they are acquired: they can be explicit or implicit.

2.1.1 Explicit feedback

Explicit feedback is the input that the users provide to the system by a mechanism in order to express their opinion regarding items. Examples of such mechanisms are rating scales like star rating systems and like/dislike buttons. Rating scales consist of range of numbers each of which denotes different interest levels that a user has towards an item. Ranges vary depending on the implementation of the rating system. In a usual 5 rating system the user denotes with 1 star those items they did not like or have no interest in and with 5 those of great importance.

2.1.2 Implicit feedback

Implicit feedback on the other hand is generated by the systems without requiring the user to express his opinion regarding items and is heavily dependent on the domain the system is designed for. It involves monitoring user behavior while using the system. The most common sources of implicit information is purchase history (e-commerce systems), views (products, videos, articles), view duration which is used to differentiate actual interest to accidental clicks or shares in case of social network systems. A combination of such measurements can be used to infer user interests.

2.1.3 Feedback differences similarities

As described in [9] explicit and implicit feedback have some key differences. Explicit feedback is more scarce compared to implicit feedback which can be overwhelming since users tend to spend much time on rating items. Explicit feedback is biased since every user uses the ranking system in a different way. A key difference is that explicit feedback contains information regarding negative feedback. Implicit feedback consists of monitored user behavior towards an item, the absence of actions for a specific item cannot be considered as negative, since the user might not have discovered this item.

2.2 Content-Based Filtering

The main idea behind Content based Information Filtering algorithms is based in comparing the item features against the user's preference on these features [10]. Thus calculating the level of interest a user has on a specific item. User's preferences on item features is called a user profile, modeled by analyzing past user behavior on item features and is an important step for content based filtering [28]. Such systems require processes for creating feature representation of the items. Furthermore, a mechanism for capturing and constructing accurate user profiles is essential. Finally the last phase of CBF algorithms is to compare items features and user profiles and create personalized suggestions for each user.

2.2.1 Item Features

Due to the diversity of items the process of extracting item features depends heavily on the specific domain. For example, some of the features that could be used for a book recommender system are the title, description and author while in an e-commerce product does not have authors, but the product category could be utilized instead.

Moreover, items hold information in a different way. A product's price and color is an example of structured data. Though information of items that exists in a structured manner is not always the case. We can consider book descriptions where there are not any limitations or predefined values as unstructured data. In addition, it is possible for a domain to hold information about its items in both ways. An example would be an e-commerce system which holds attributes of its products like price and color as well as a description of it. In cases where the extraction of item features from unstructured data is required, techniques from information retrieval systems and natural language processing algorithms are usually utilized [2].

As described in [10], one of the usual approaches is for the system to transform unstructured to structured information. An example would be to count the number of occurrence of words in text. An alternative is stemming, i.e. the process that creates terms to categorize words with the same root and meaning. The system can then calculate the importance level of a term in an item. Importance is measured by calculating the *tf*idf* (term-frequency times inverse document frequency) weight, where a document is the feature of the item we want to process, i.e. product or book description. From [10] the weight $w_{t,d}$ of the a term t in a document d is a function of the frequency of t in the document tf_t, d , the number of documents with the term df_t and the total number of documents is N.

$$w(t,d) = \frac{tf_{t,d} \log\left(\frac{N}{df_t}\right)}{\sqrt{\sum_i (tf_{t_i,d})^2 \log\left(\frac{N}{df_{t_i}}\right)^2}}$$

The idea behind Term Frequency-Inverse Document Frequency is that terms with high occurrence in a document but rare in the rest are more relevant and important [2].

2.2.2 User profiles

Content-based Filtering systems aim to create profiles of user interests. These profiles can be constructed from various sources. They could consist of explicitly stated preferences using predefined terms or categories. In addition they can be constructed with the use of implicit information from past interactions like queries, views or time spent browsing.

In cases of explicitly stated preferences, the system offers a way for the user to express his interests. This can be done in a variety of ways, such as by presenting him with a set of products and asking him to rate or choose those of interest to him or by allowing him to build his own profile by stating topics of interests.

This systems have some limitations while at the same time they are not suitable for every case. One of its main disadvantages is that usually users do not like to provide or spend time rating items. Especially in e-commerce systems which they aim at minimizing the effort required by the user to complete a transaction, such implementations hinter overall performance of the system.

2.2.3 Classification Algorithms

Another way that user profiling can be used is to feed a classification learning algorithm [10]. Such algorithms are used extensively in content based algorithms since they learn a model of user's interests. The classifier given a model and a new item can predict if the specific item is of interest to the user.

Some approaches use probabilistic models on past user interaction. Naïve Bayes used for text classification is an accurate and popular example of such algorithms. The idea of the algorithm is to calculate the probability of a document to belong to a specific class. The probability for any document to belong in this class, the probability the specific document to belong to this class and the probability to see this document are used. The Bayes theorem is applied to these probabilities:

$$P(c|d) = \frac{P(c)P(d|c)}{P(d)}$$

The document belongs to the class with the highest probability.

Another established algorithm used in content-based filtering, and more specifically in text categorization domains is Relevance feedback and Rocchio's Algorithm [2]. The main idea is to let users specify if the items returned by the system are interesting to them. The feedback can then be used to improve the retrieval of items. Relevance feedback has been the main focus of many researchers that aimed to develop information retrieval algorithms.

Rocchio's algorithm is a relevance feedback algorithm. The algorithm modifies the initial query by weighting relevant and irrelevant documents. The approach as described in [10] creates two document prototypes. A relevant document prototype which is created from the vector sum of all relevant documents and an irrelevant document prototype which is created in the same way as the relevant one, a vector sum of irrelevant documents. Q is the query, α, β and γ are parameters used to control the influence of the initial query and the vector sums are the relevant and irrelevant prototypes.

$$Q_{i+1} = \alpha Q_i + \beta \sum_{rel} \frac{D_i}{|D_i|} - \gamma \sum_{nonrel} \frac{D_i}{|D_i|}$$

The aim of the formula is with each iteration to transform the query vector closer to relevant clusters and away from irrelevant ones.

Researchers have used a modification of the Rocchio algorithm for user profiling in unstructured text. In Rocchio-based classification, a vector sum is created for each class by summarizing the documents that belong to this class. These vectors can be used against the vector of an unlabeled document in order to classify it.

Another learning method used in content-based filtering is the Nearest Neighbor algorithm [2]. This type of algorithms is very simple and straightforward. In order to classify a new item, they

compare it with all the known items and is classified based on the labels of its nearest neighbors. The comparison is done by similarity function, such as the cosine similarity. Even though this type of algorithms are proved to be accurate their overall efficiency drops due to execution time. Due to the fact that the item is compared to all previously stored known items during classification the classification time required is long. Though there are algorithms that implement it for modeling user interests.

2.3 Collaborative Filtering

The main focus of collaborative filtering is to model a very common thing that humans do during their decision making process. It aims to model the habit of sharing opinions about items with other people, thus influence and be influenced. People often discuss their opinions on products they bought or a restaurant they have been to. During the decision making process people take into account these opinions.

The collaborative filtering approach does not rely on the content of the item in order to produce recommendations. It relies on the ratings or past behavior of the user as well as other users [27] [2]. It is based on the idea that a user is likely to prefer an item also preferred by similar users .

This approach overcomes some of the drawbacks of content-based filtering. One of the key advantages is that it does not deal with item representation, thus it has reduced complexity as it does not deals with feature extraction and it is suitable for systems where the content of items is not available. Moreover, it is more likely to produce diverse recommendations to a user as it does not depend on a content defined model of the user's interests.

Collaborative filtering, even though proved to be effective, heavily depends on the domain that it is applied. The specific use case should fit some prerequisites in order to be suitable for collaborative filtering. In [11] some factor that affect collaborative algorithms are presented.

Data can influence directly the efficiency of the algorithm. First there should be many ratings per item. These algorithms have been observed to increase in accuracy when there is abundance in

ratings, while they hinder in their absence. The number of users should be greater than the number of items. If the number of items is bigger the system would be able to create accurate recommendations only for a relative small amount of items. Lastly, the user is required to rate multiple products, otherwise there are not any information about related items.

Collaborative filtering algorithms can be divided in categories based on their design. The most known distinction is item-based and user-based filtering, which will be presented in the following paragraphs. Though according to [12] they can also be categorized based on the way of their implementation. Collaborative algorithms that require the whole set of ratings, users and items loaded in memory are denoted as memory-based algorithms, while the rest are model-based. Model based algorithms depend on summarizing rating patterns offline and periodically refreshing this summary. In real-world applications though, where the number of users and items can be enormous, memory-based algorithms require a lot of resources and thus are deemed unproductive. Moreover, another distinction among algorithms can be done if they utilize probabilistic models for their recommendations or not [11]. Probabilistic and non-probabilistic algorithms.

2.3.1 User-Based Collaborative Filtering

Common collaborative algorithms used user similarity for their predictions. Similar users are usually described as neighbors. In user-based algorithms this association is utilized in order to predict preference of a user towards a specific item. This is achieved by processing the ratings of all the user's neighbors. A very simple form of this approach could take as prediction the average rating score of all the neighbors to this item.

Denoting as *u* the user, as *i* the item, as *n* the neighbor, as N the total number of neighbors and as *r* the rating of the neighbor then then p_{ui} is the predicted rating of user *u* towards *i* and is equal to

$$p_{ui} = \frac{\sum_n r_{ni}}{N}$$

The above equation is naïve because it does not take into consideration that all neighbours are not equally similar with the user. In order to consider this fact the equation is transformed to consider user similarity.

$$p_{ui} = \frac{\sum_n uSim(u,n) * r_{ni}}{\sum_n uSim(u,n)}$$

Furthermore, in order to enhance accuracy we should consider that a user makes different use of the rating system. Simply put, some users tend to give higher rating while others lower, but in the essence they express the same thing. In order to address this the equation should be further adjusted with users' mean scores.

$$p_{ui} = \overline{r}_n + \frac{\sum_n uSim(u,n) * (r_{ni} - \overline{r}_n)}{\sum_n uSim(u,n)}$$

Practitioners that aim to implement user-based algorithms need to take into consideration some obstacles they might face. What is often the case is that some users might not have many similarities with other. In this case recommendations will not be accurate. Moreover, it might also happen for neighbour users' ratings to match exactly; this could render the rating of other users ineffective. In addition, in order to create neighbourhoods each user should be compared to every other user. In real-world applications where there is a plethora of users this would be very expensive. In order to alleviate such problems some techniques have be used. A possible approach would be to sample the users. Lastly, another approach would be to utilize clustering algorithms to locate user's neighbours.

2.3.2 Item-Based Collaborative Filtering

In a similar manner to user-based algorithms, item-based collaborative filtering algorithms base their recommendations considering similarities, alas in this case we compare item similarity instead of user [31]. To make it clearer in order to predict a user's rating for a specific item we

take into account its similar items and the ratings the user have provided for them.
Denoted as p_{ui} the predicted rating of user u for item i, as j a rated item by user and as r_{uj} the rating of the user to item j.

$$p_{ui} = \frac{\sum_j iSim(i,j)\, r_{uj}}{\sum_j iSim(i,j)}$$

The above equation is very similar to user-based algorithms. The key difference is that in this one we have item similarity instead of user similarity. Another difference to the user-based equation is that in this case we do not include the average rating of the user. This happens because all the ratings are from the same user.

Variations of item similarity exist, though [11] notes that the most popular and possibly most accurate similarity metric is the adjusted-cosine similarity. Adjusted cosine similarity takes into account the ratings of all the users. In the following equation we denote as u those users that have rated both i and j.

$$iSim(i,j) = \frac{\sum_u (r_{ui} - \overline{r}_u)(r_{uj} - \overline{r}_u)}{\sqrt{\sum_u (r_{ui} - \overline{r}_u)^2}\ \sqrt{\sum_u (r_{uj} - \overline{r}_u)^2}}$$

Regarding the implementation of this algorithm and in order to increase its algorithmic efficiency it has been proposed to define a minimum amount of required co-ratings for a similar item to be considered in the algorithm. Similar to user-based algorithm this approach also hinders when items have too few co-ratings.

2.4 Contrasting Collaborative and Content-Based filtering

Content-based and Collaborative filtering are essentially different. On one hand content-based is based on the idea that a user is likely to prefer items to those he previously liked. In this case the main problem is to efficiently extract item characteristics and form user profiles [36,2]. On the other hand collaborative filtering does not deal with content and extracts its recommendations based on a user's ratings compared to similar user's ratings. In this case large volume of ratings and preferences is required to be effective. Though, not having to deal with content lifts a big burden from the process.

Since collaborative filtering does not rely on modeling a user's profile, it usually presents more diverse recommendations which are suitable for discovering unknown interests. In this matter content-based is restricted to following the defined user preference model. Thus, it is possible not to recommend interesting items to the user because they differ slightly from his preferences.

2.5 Matrix Factorization

Even though we present matrix factorization in a different section, it is a sub-category of collaborative filtering algorithms. The approaches described earlier for collaborative filtering are usually referred as neighborhood approaches, due to the fact they are based on user or item neighbors for recommendations. Matrix factorization belongs to the latent-factor models' category of collaborative filtering algorithms. In these algorithms are characterized by vectors of factors [13]. Over the last years this approach became very popular due to their accuracy and efficiency.

Matrix factorization models can be used in either explicit or implicit datasets. Users and items can form a matrix with every row representing a user and each column an item. In cases of explicitly gathered information the values denote the rating of the user for each item. In implicit

feedback datasets though values measure the preference inferred from monitoring user behavior like view duration or purchase history.

The main goal of this approach is to decompose the user-item matrix to two matrices. Each of these matrices consists of user or items and same number of factors. Factors are the characteristics each item might have like the color and fabric the item is a clothing or genre if we the item describes a movie. User-factor matrix describes the preference of each user towards to items possessing each characteristic. On the other hand, Item-factor matrix describes how relevant each item is to each characteristic.

We can then visualize users and items as vectors and factors as the dimensions these vectors exist. The dot product of a specific user vector with a specific item vector approximates the interest level of the user towards this items taking into consideration all of its characteristics. Thus we can form recommendations on the result of the dot product.

Denoting users as u, items as i, the user vector as x_u, the item vector as y_i. Then the predicted recommendation r_{ui} is formed from

$$r_{ui} = y_i^T x_u$$

We can easily understand that the success and effectiveness of a matrix factorization approach is to correctly calculate user and item vectors. There are several approaches regarding matrix factorization. A well-established model used in information retrieval algorithms is the Singular Value Decomposition SVD.

In order to learn the user and item factors some the system regularizes the squared error of the known ratings following the function [13].

$$min \sum_{ui} (r_{ui} - x_u y_i^T) + \lambda(||x_u||^2 + ||y_i||^2))$$

λ is the constant that controls the regularization and is taken using cross-validation method.

A very interesting approach regarding implicit datasets was presented in [8]. The main idea was to predict the level of preference the user would have towards and item rather than the score. The preference of a user for an item is modelled by assigning 1 to item the user interacted with and 0 to each he has not.

$$p_{ui} = \begin{cases} 1, & r_{ui} > 0 \\ 0, & r_{ui} = 0 \end{cases}$$

Moreover the preference is associated with confidence. Confidence was modelled as

$$c_{ui} = 1 + ar_{ui}$$

Confidence aims in scaling the preference of the user towards an item in cases that the user viewed the item many times.

Similarly to the previous explicit feedback approach the vectors obtained by minimizing a cost function.

$$min \sum_{ui} c_{ui}(p_{ui} - x_u y_i^T) + \lambda(||x_u||^2 + ||y_i||^2))$$

Moreover, the model is further developed in [14]. In this approach the algorithm computes the preference probability of a user item pair. Thus the cost function furtherly modified into while at this case aimed in maximizing it.

$$max \sum_{ui} c_{ui}(x_u y_i^T) - (1 + c_{ui}) \log(1 + \exp(x_u y_i^T)) - \frac{\lambda}{2}\left|\left|x_u\right|\right|^2 - \frac{\lambda}{2}||y_i||^2$$

Different approaches regarding minimization of the cost function were also developed. Two significant are the Stochastic Gradient Descent SGD approach and the Alternating Least Squares ALS.

In Stochastic Gradient Descent the algorithm predicts r_{ui} for each given rating and calculates the prediction error by:

$$e_{ui} = r_{ui} - y_i^T x_u$$

And then modifies the parameters with:

$$y_i \leftarrow y_i + \gamma(e_{ui} x_u - \lambda y_i)$$

$$x_u \leftarrow x_u + \gamma(e_{ui} y_i - \lambda x_u)$$

In Alternating Least Squares [13] the idea is that by fixing one of the two unknowns of y_i and x_u the optimization problem is quadratic and there is an optimal solution. Following this idea the ALS alternates by fixing on of the vectors.

The key difference of these approaches concerns efficiency and their scalability. SGD approach requires looping over each training case so it is not suitable for dense matrices. Explicit feedback datasets are usually sparse which makes SGD an easy and appropriate approach. On the other implicit feedback dataset that contain abundant information produce denser matrices where SGD might not scale appropriately and thus ALS is more efficient in such cases. On top of the above ALS computes vectors independently and offers parallelization of the algorithm which greatly benefits computation on large datasets and efficiency. Matrix factorization is also proposed by the literature to be suitable for social network environments [34].

2.6 Hybrid Algorithms

Hybrid algorithms consist of various combinations of the previous approaches [29]. Hybrid approaches usually are more efficient and more accurate because they combine techniques to alleviate limitations. Though they are not suitable for simpler cases because of the increased complexity level they present during the design and implementation phase.

2.7 Evaluation

Success of a recommender system is based in a number of properties. Context of the specific domain where the recommender will be deployed heavily affects the overall performance of the system. Different use cases of recommender systems aim to satisfy different needs. The properties that affect the systems success have to be identified for evaluating the system.

Most recommendation systems implement a prediction algorithm, thus prediction accuracy metrics are commonly used for system evaluation. Root Mean Squared Error (RMSE) is commonly used to measure accuracy. Root Mean Squared Error is calculating the squared difference between the predicted ratings rp and the true ratings rt for a test set S_t of user item pairs.

$$RMSE = \sqrt{\frac{1}{|S_t|} \sum_{(u,i) \in S_t} (r_{pui} - r_{tui})^2}$$

The Mean Absolute Error is an alternative to RMSE and is calculating the absolute difference of predicted and real ratings on a test set.

$$MAE = \sqrt{\frac{1}{|S_t|} \sum_{(u,i) \in S_t} |r_{pui} - r_{tui}|}$$

The squared difference results in lower ranking for systems that present larger errors compared to systems with more but smaller errors.

In unbalanced test sets, where some items are used or rated more frequent than the others, the error in a frequent item might hinder system evaluation. Average RMSE and average MAE are used in these cases. The RMSE or MAE of each item is calculated separately and their average forms the system evaluation.

In other cases the recommender system aims to predict user behaviour and create recommendations predicting which items the user will use. In these cases explicit rating feedback

is not available and RMSE and MAE are not suitable. Instead there are four possible outcomes, True-Positive, False-Positive, False-Negative and True-Negative. True-Positive is the case where the item was recommended and did used by the user, False-Positive is the case where the item was recommended but not used, False-Negative is the case where the item did not recommended and it used and True-Negative is the case where the item was not recommended and did not used by the user.

In offline evaluation it is possible to take wrong measurements of False-Positives because the test set is extracted without presenting recommendations to the user. Thus there is no knowledge for a user not using an item because it is not preferred or because it is unknown to him.

From these measurements the following metrics can be computed for the system.
Precision P which measures the proportion of accurate positives over the total number of positives, given by.

$$P = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

Recall R which measures the proportion of accurate positives over the total number of actual positives. The total number of actual positives is equal to true positives and false negatives.

$$R = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

False Positive Rate FPR which measures the inaccurate positives over the total number of actual negatives. The total number of actual negatives is equal to false positives and true negatives.

$$FPR = \frac{False\ Positives}{False\ Positives + True\ Negatives}$$

For recommendations systems with a specific number of recommendations presented to the user it is efficient to measure how many of the recommended items where actually relevant. Thus precision is more suitable.

In systems without a predefined limit on the number of recommended items precision-recall and Receiver Operating Characteristic ROC curves are more suitable. For Precision-recall curve the

precision and recall of the system is compared. In ROC curve the comparison is made between the true positive and false positive rate. Precision-recall considers the items that recommended and actually used while ROC considers the items that falsely have been recommended.

In addition to rating prediction accuracy and usage prediction metrics there are cases where the recommendation system does not exclude irrelevant items or predicts ratings for items. Some recommenders aim to order the items from the most relevant to the least relevant in order to produce recommendations, typically referred as ranking. Evaluation of ranking systems is achieved in two ways either comparing the resulted order of ranked items to the actual order or measuring if the recommended order is useful to the user.

Recommender systems do not exclusively aim in predictive accuracy. Coverage is typically described as the amount of items for which the recommender is able to produce recommendations. Catalog coverage is equal to the percentage of items that can be recommended compared to the total number of products available.

In addition to catalog coverage, some systems do not produce recommendations for users with insufficient information. A similar metric to item coverage is the coverage regarding the number of users the system is able to provide recommendations.

Cold start is a main problem especially for collaborative filtering algorithms where information about a new item or a new user are not available and thus the system is unable to produce recommendations. It is desirable for the recommenders to be able to cover new items and users to their recommendations. Moreover can also be measured the accuracy on these items. Recent papers suggest addressing the cold start problem with the aid of social networks [30, 32, 20]

Another property affecting recommender systems performance is its scalability. As the number of items and users grow the systems requires more processing power and larger memory. In such cases many systems trade off accuracy and precision to be able to continue performing. An important measurement regarding scalability is execution time in various dataset sizes.

3 Literature Review

Here we present several different e-commerce recommender system approaches and their advantages or disadvantages. We see recommendation systems utilizing data mining techniques, clustering analysis (in which the system focuses on customer's comments and reviews), Dynamic Table based approach which takes into account customer's life cycle for his recommendations, mining user-contributed photos and by using visual and UGC for user interest mining, Fuzzy logic, portal. One of the main conclusions is that the mentioned systems aim to enhance conversion rate (the buy part of recommendations) by providing more personalized recommendations. Further on the literature regarding E-commerce recommender's gives and general overview focusing more on how recommender systems affect business. Also gave some insight regarding issues that might arise at the implementation of such systems in real life examples such as security. Next paper the reviewed literature regarding product recommendation qualifies some criteria.

Furthermore description while also advantages and disadvantages of each technique is presented. Mentioned approaches are the classic Content Based filtering, Collaborative filtering, Hybrid (and ways it is applied), Social network based. Then a view of the e-vendors is mentioned which suggest that the cold start and long tail problems, even though addressed, still have room left for further research. Also it is stated that in the last year research regarding lowering computational complexity and increasing accuracy has drawn much attention. Researchers have demonstrated approaches that outperform state-of-the-art approaches in accuracy and scalability and that lowering complexity improves performance of RS. Findings and drawbacks are being presented and concludes with proposing some areas that need further research. What stood out is that not many systems take into account changes in user preferences and suggests research in Dynamic user behavior and ratings behavior.

3.1 Recommender Systems

3.1.1 Collaborative Filtering for Implicit Feedback Datasets

In [8] the research addresses cases where explicit feedback is unavailable and presents an algorithm producing recommendations based strictly in implicitly gathered information. The algorithm is categorized as Collaborative Filtering and more specifically as a Matrix Factorization implementation. Furthermore the algorithm provides explanation regarding the recommendations that produces.

The dataset used consists of viewing history of customers from 300,000 top boxes of a digital television service. Each piece of information shows how many times a user watched a show during a four week period.

In explicit feedback matrix factorization algorithms the matrix consists of user-item and their ratings. In this case the user-item pairs regard estimations for a user to prefer a specific item. This estimation is referred as preference. Preferences is set to 1 for each show the user have watched. Notating users with u, items with i, preference with p and the observed times a user watched a show as r the preferences is given by:

$$p_{ui} = \begin{cases} 1 & r_{ui} > 0 \\ 0 & r_{ui} = 0 \end{cases}$$

Aiming to include cases where factors that led the user in watching the show other than personal preference, the equation is enhanced with a confidence variable. Increasing number of times a user watched a show lead in higher level of confidence that the user actually likes the show. Confidence is constructed using a predefined constant that sets the increase rate of each view. Notating confidence with c and the predefined constant with a, the confidence is given by:

$$c_{ui} = 1 + a\, r_{ui}$$

The current implementation for confidence level is subject of experimentation as there are other possible ways that could fit better in other scenarios. Similar to traditional matrix factorization techniques the aim is to create user vector for each user and item vector for each item and extract recommendations using their inner product.

In this case the cost function that is minimized to extract factors is the following:

$$min_{x,y} = \sum_{u,i} c_{ui}(p_{ui} - x_u^T y_i)^2 + \lambda \left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2 \right)$$

In addition to the different cost function compared to explicit matrix factorization, a different approach for the optimization process is also implemented. The approach used is referred as alternating least squares optimization. The idea is in each iteration to keep one vector static and re-compute the other lowering the value of the cost function while in the next iteration to alternate the static and re-computed vectors. After computing the user and item vectors, x_u and y_i respectively, the predicted user preference for an specific is given by their dot product.

$$\hat{p}_{ui} = x_u^T y_i$$

Another useful feature in this implementation is the ability of the system to provide explanations as to why each item was recommended. This is often a desired feature for recommender systems because helps the user understand the reasoning behind the recommendations presented to him.

For the evaluation of the system precision metrics deemed inappropriate due to the inability to know user feedback over recommendations. The methodology used for evaluation of the system is referred as mean percentile ranking. Each recommendation listed is ordered and assigned a percentile ranking. Ranking with 0% the ones that the systems predicts to be more preferable by the user. Notating with *rank* the percentile ranking of an item in the test period the mean percentile ranking is given by:

$$\overline{rank} = \frac{\sum_{u,i} r_{ui}^t \, rank_{ui}}{\sum_{u,i} r_{ui}^t}$$

To consider the result accurate lower percentile mean is required. The algorithm is compared to two other models, one that produces recommendation based on popularity and an item-item collaborative filtering algorithm. The evaluation showed that this methodology produced the best results among the three.

The proposed implementation performs well and sets the ground rules for dealing with implicit feedback recommenders. The core idea in the implementation presented in this dissertation is based on this algorithm. The matrix is being constructed using a similar equation to map user behavior to user item preference while similar to this alternating least squares and same cost function is used for creating the user and item factor vectors. In addition, researcher based on this approach applied various techniques and methods in order to further improve the model such as [13,36]

3.1.2 Time-Dependant Recommendation based on Implicit Feedback

In [15] we see an implementation of a context-aware system that aims to take into account time as well as depending strictly in implicit feedback for producing recommendations. The main assumption is that user preferences mutate over time but they also tend to repeat. For the system to include time for producing recommendations, a micro-profiling technique is implemented. The algorithm is tested using information which consist of two years' worth of listening habits from 338 random users of the last.fm.

The main concept is to divide a user's profile into micro profiles. This is done in an effort to better describe user taste during a specific time of the day. Similarly suggests that micro profiles can be used to describe longer time intervals such as days, months or years. A key challenge to this implementation is to effectively distinguish representative time intervals from user behavior. Another challenge presented is that time interval might differ amongst users. Though these challenges are considered out of scope of this research. The goal of this research is to test if micro profiling brings better results in a collaborative filtering algorithm.

Furthermore, this paper proceeds to discover the best way to partition the customer information. The results showed that an effective way of segmenting monitored customer behaviour can increase the accuracy of a collaborative filtering algorithm. Specifically better results were observed for hour and day segmentation.

This approach regarding recommendations dictate that context and time are important for the accuracy of the system. Taking into consideration all the available information might hinder the accuracy of the system, especially when recommenders aim to predict short-term preferences.

3.1.3 An Intelligent Product Recommender System in an e-Store

In [16] the proposed system aims in creating recommendations without using customer ratings. It is described a complete system which incorporates collaborative and non-collaborative algorithms on its various stages. The system consists of three parts. The first part is responsible for creating customer clusters. The second is responsible for creating maintaining and update the customer profiles. And the third part is responsible for the presentation of the recommendations to the customer

Clustering the customers is achieved using their preferences. Preferences are provided by the profiling agent. Customers with similar preferences are grouped together and then collaborative filtering techniques produce a ranked list of suggestions that are being fed to the profiling algorithm.

In the second part of the system the customers behaviour is monitored and each preferred items is being added to the customer's personal preference record. In addition this agent of the system is responsible for ranking the items suggested by the clustering part of the system. Lastly is responsible for extracting customer's preference criteria from the observed behaviour.

The generic algorithm implemented in the proposed system considers item popularity on the users with matching preference criteria. Another interesting feature implemented in this approach is mutation. Mutation randomly transfers a set of irrelevant products to the recommendation list.

In cases the user prefers the mutated recommended items reveals that the system produces false negatives and enables profiling algorithm to recalculate user preferences taking into consideration the added information. It is a way of dynamic self-learning algorithm based on user feedback.

In this paper it is described a hybrid algorithm of collaborative filtering with user profiling which is mostly implemented in content-based filtering algorithms. There are key features that improve the overall performance of a recommender system like the self-learning mutation technique. Though the accuracy and performance of the systems is not certain because the evaluation was performed on randomly created data and on only five customers. Real world applications scale a lot more than that. The fact that the system is able to produce accurate predictions on a small number of users does not guarantee that the average performance will remain taking into calculation some thousands of customer's behaviour. Further evaluation of the system is required to monitor the performance on large datasets.

3.1.4 A product recommendation system in e-commerce

In [17] the paper introduces a hybrid algorithm implementing collaborative filtering and association rules for producing recommendations. It also regards implicit feedback information and presents an approach of transforming transaction information to an implicit rating system. Furthermore it presents a way in which sequence recognition can improve the systems performance.

The collaborative filtering utilizes weighted cosine similarity on user ratings. Cosine similarity CS between user 1 and user 2 is given by the type:

$$CS_{r1,r2} = \frac{dot(r1, r2)}{\|r1\| \, \|r2\|}$$

As r is notated the ratings of the customer's. Furthermore the paper suggests improving the method by including purchased item frequency of the rated items. This methodology is referred by the authors as implicit rating. The system gets information from user's ratings and

transactions and transforms transactions in a vector space model. It calculates the item frequency of a user item pair and the inverse item frequency of that item in order to create the implicit rating of the user regarding a specific item. Item frequency is the frequency of an item purchased by a user. The calculation of item frequency is achieved in two ways. Notating as n the frequency of item i purchased by the user u the first method calculates IF with:

$$IF_{ui} = \frac{n_{u,i}}{\sum_I n(u,I)}$$

In contrast the second method calculates IF with:

$$IF_{ui} = \frac{n_{u,i}}{max_{u,I}}$$

Notating item frequency with IF, user with u, item with i, inverse item frequency with IIF, U the total number of users and U_i as the total number of customer purchased a specific product, then the Inverse Item frequency is given by:

$$IIF_i = log\frac{1+U}{U_i}$$

And the implicit rating ir of a user for an item ig given by:

$$ir_{ui} = IF_{ui}\, IIF_i$$

Furthermore provides a way calculating item frequency for new users where observations are not available. The item frequency for new users is given by:

$$IF_{ui} = log\frac{U_i}{1+U}$$

The system is designed to produce additional recommendations using purchase history for extracting association rules. Measuring the support and confidence found in the purchase history

of users the system is able if a user is likely to buy a product based solely on his previous purchases.

Lastly, the system considers sequence recognition. It records the order of purchased items and if a recommended item is not sold after the specific sequence of previously bought items it is not included in the recommendations.

Experimental results demonstrate the superiority of explicit rating precision in contrast to implicit. Though the combination of explicit and implicit provide slightly better results than standalone explicit rating regardless the method used for including item frequency. In addition recall is increased by implementing the association rule recommendations in the system.

Even though the improvement of accuracy using a combination of implicit and explicit rating is not big it still is an improvement. Moreover the paper presents a system that produces multiple recommendations, some items are recommended via collaborative filtering and others from association rules. Lastly points out a way in which sequence analysis in product purchases can benefit recommendation systems.

3.1.5 A case study in a recommender system based on purchase data.

In [18] we have a recommender system used by salesperson in a physical store as a tool for recommending products to the customers rather than the usual implementation of an online recommender. The algorithms used in this paper consider purchase history and association rules to produce its recommendations. Moreover the results point out the importance of context aware recommender systems.

The information used for the training of the proposed model use a user item matrix. Each row of the matrix includes purchase information of a customer for each item over a specific time period. For evaluating the hypothesis that context can increase prediction accuracy, three algorithms have been implemented and evaluated. The first algorithm implements and item-based collaborative algorithm, the second implements a matrix factorization algorithm using singular

value decomposition and non-negative matrix factorization and the third one creates a bigram matrix based on association rules mining.

Each algorithm takes as input the purchase history of customers either with a time limitation or not and returns a list of recommended items. The time limitation that was used was a two week period. So in the case were there is no time limitation the algorithms need to extract recommendations using all the available information from this day and before while when using two weeks history a portion of information worth of two weeks' time is used.

Evaluating the three algorithms on the two different time settings showed that the most accurate algorithm when using full purchase history is matrix factorization algorithms in contrast to the cases of two weeks limit imposed on data association rules present the best results. Comparing performance of the same algorithm in the different time limitation shows that all four algorithms have been affected positively. Though it is noted that in the case of SVD the difference is considered insignificant since it is no more than 0.02%. Association rules presented an increase in accuracy of 3% to 4%. Item based collaborative filtering also benefits significantly with ranges from 3.5% to 4%. Lastly non negative matrix factorization benefit ranged from 1% to approximately 2%. It is observed that matrix factorization techniques do not benefit greatly from time awareness in the input data.

The paper then proceeds in presenting a business oriented approach taking into account knowledge of the specific domain. The data consider purchase history of a store with home improvement products. The customers purchasing history of the store show a significant pattern in their behavior. It is observed that at times customers buy only standard not expensive things while at some short period they spend more buying specific products. This is explained by the company that at these cases the customers have started working on a specific renovation project and they need a collection of products to finish. The aim of the proposed approach is to effectively detect when a customer is in the middle of a project and be able to recommend relevant products. Projects are identified as short periods of no more than two weeks with high purchase activity.

The evaluation of the three algorithms on the context aware dataset shows that all three of them increased their accuracy thus proving the importance of the context for a recommendation engine.

The proposed system is considered successful and relevant to the goals of this dissertation because manages to produce recommendation using solely purchase history which falls in the category of implicitly gathered information, incorporated context and time for improving recommendation algorithms accuracy, moreover describes a way of incorporating context in specific domain and lastly demonstrates the importance of purchase history and association rules since the algorithm with highest the accuracy was the bigram matrix created from association rule mining.

3.1.6 Logistic Matrix Factorization for Implicit Feedback Data

In [14] the research is also focused on implicit feedback systems and matrix factorization. Moreover the paper demonstrates how the model can scale using technologies like Hadoop and Spark. For evaluation the proposed algorithm is being compared to the traditional Implicit Feedback Matrix Factorization presented in [8] and a popularity algorithm.

The algorithm concerns a probabilistic approach of implicit matrix factorization. The approach is similar to [8]. It also infers to lower rank vector that model user and item factors. The difference is that instead of trying to minimize the cost function mentioned, this algorithm implements a probabilistic model.

The preferences of users towards items can be described using a logistic function taking into account the sum of the inner products of user and item factor vectors. Moreover it is also affected by user and item biases. So the logistic function is given by

$$p(l_{ui} \mid x_u, y_i, \beta_u, \beta_i) = \frac{\exp(x_i y_i^T + \beta_u + \beta_i)}{1 + \exp(x_i y_i^T + \beta_u + \beta_i)}$$

Notating with $l_{u,i}$ the interaction of a user u with item i, β_u β_i are the user and item biases respectively and x_u and y_i are the user and item factor vectors respectively.

Users will tend to have differences among them concerning the number of items they interact with. User biases models this behaviour for each user. Similarly some items will be more popular than others and are being preferred by a wider range of customers while others only preferred by small groups.

In this approach it is also utilized the confidence variable in the same manner it is implemented in [8] given by:

$$c = a\, r_{ui}$$

The likelihood of observations given the user and item factor vectors and biases is given by:

$$L(R\,|X, Y, \beta) = \prod_{u,i} p(l_{ui} \mid x_u, y_i, \beta_u, \beta_i)^{ar_{ui}}\,(1 - p(l_{ui} \mid x_u, y_i, \beta_u, \beta_i))$$

In addition it uses Gaussian priors on user and item factor vectors for regularization. The final logistic function after further processing is given by:

$$\log p(X, Y, \beta \mid R) =$$

$$\sum_{u,i} ar_{ui}(x_u y_i^T + \beta_u + \beta_i) - (1 + a\, r_{ui}) \log(1 + \exp(x_u y_i^T + \beta_u + \beta_i)) - \frac{\lambda}{2}\|x_u\|^2 - \frac{\lambda}{2}\|y_i\|^2$$

Finally the goal is to find the user and item vector as well as biases that maximize the log posterior. In a similar fashion were the alternating least squares where used for minimizing the cost function, here for finding the maximum an alternating gradient ascent procedure is implemented.

Furthermore due to the fact that iterations scale linearly with the increasing number of users and items the process is limited to small datasets. In situations with larger dataset it is suggested to sample fewer negative samples alongside reducing the “a” parameter of confidence equation.

The “a” value in confidence serves as a balance between negative, meaning zero interaction, and user item interaction, because the bigger the value of “a” is the more weight is given to the observed interactions.

Another novelty in the systems is that in each iteration in order to increase the user or item vector the step size can be chosen implementing AdaGrad. Notating with t the iteration and as x_u the user factor vector, the next vector is given by:

$$x_u^t = x_u^{t-1} + \frac{\gamma g_u^{t-1}}{\sqrt{\sum_{t'=1}^{t-1} g_u^{t'^2}}}$$

In each iteration of the alternating gradient ascent procedure the computation of the gradient for all factor vectors. Because each gradient consists of functions related to a specific user item pair this makes the proposed implementation suitable for MapReduce programming paradigm.

For the scaling using MapReduce paradigm the preference matrix R is being divided into K x L partitions with K rows and L columns where K is smaller than the total number of customers and L smaller than the total number of items. Additionally the user and vectors are also being divided to the number of rows the R divided, K, and to the number of columns the R divided, L, respectively. The following equations compute the user factor vector and bias during the iterations where item factor vector is fixed.

$$u_{ui} = ar_{ui}y_i - \frac{y_i(1 + ar_{ui})\exp(x_u y_i^T + \beta_u + \beta_i)}{\exp(x_u y_i^T + \beta_u + \beta_i)}$$

$$\beta_{ui} = ar_{ui} - \frac{(1 + ar_{ui})\exp(x_u y_i^T + \beta_u + \beta_i)}{1 + \exp(x_u y_i^T + \beta_u + \beta_i)}$$

In the same way the computation of item factor vector and bias is being calculated on the

iterations that user vector is fixed. During the reduce phase all the user vector and biases are aggregated and when all the computations that take place on user vector or items regarding iteration are finished the system proceed into updating the user or item factor vector using the equation presented.

The evaluation of the system made using the same metric used in the classical implementation. Again the reason behind the certain evaluation is the limitation imposed by nature of the problem where negative feedback from users regarding items is not existent in an implicit feedback dataset. Evaluation of the methodology showed that the logistic matrix factorization model perform slightly better that classical implicit matrix factorization. Another positive characteristic of the logistic matrix factorization is that presented better results with taking into account less latent factors. Computation for less latent factors has many advantages in terms of execution time and reducing resource cost of the algorithm. Moreover increasing latent factors requires more interaction observations in order to produce optimal user and item vectors.

The proposed algorithm is a successful experiment since it manages to outperform even slightly the cornerstone of implicit feedback matrix factorization algorithm. Moreover an interesting characteristic provided is the use of parallelization for cases where the system scaled greatly. Lastly the proposed algorithm is actually used by Spotify in order to produce recommendations.

3.1.7 Life-stage Prediction for Product Recommendation in E-commerce

In [19] the presented system takes into consideration the life stage of the customers. More specifically considers products for babies and based on customer's choice of products tries to infer in which life stage he/she is (newborn, 1-3, etc.) and recommends the related products. It is an approach which relates to dynamic user profiling and aims to provide personalized recommendations.

It is stated in the paper that according to sociologists and marketing researchers there is a strong relevance between the life stage of a user and his preferences regarding items. The proposed

system incorporates a methodology for discovering the life stage of the customer and using this knowledge with a probabilistic model for recommendations.

Customers past purchase sequence is used with a new Maximum Entropy Semi Markov model, proposed by the authors, in order to perform stochastic life stage segmentation. The probability of life stage at time t depends on the previous life-stage, the time the user was in the previous stage and his behavior sequence. Notating the user behavior sequence as X, life-stage probability as y and duration of a life stage as d, the system aims in finding the best life stage sequence and corresponding duration of each.

$$\{y_1, \dots, y_k, d_1, \dots, d_k\} = \underset{k, d_y, y_k}{\operatorname{argmax}} \prod_{t=1}^{k} P(y_t \mid y_{t-1}, d_{t-1}, X_t) P(d_t \mid d_{t-1})$$

As l_{min} and l_{max} is denoted the minimum and maximum time of life stages as they were defined by domain experts. For the estimation of the probability a multinomial logistic regression model is implemented.

The features of the classifier that the system use are categorized as follows. Category features, instead of using the ids of the products interacted with the customer as features, the category ids associated with those product are used. This is done due to possible sparsity products and because the store change its items frequently. Queries, user's search queries provide valuable information about the current life stage of the customer. Product properties, products are described in the system using feature-value pairs. Values of the product features many times concern specific age. Product title, titles also many times provide information regarding age that are used for. Temporal effect of the features, the described features provide different interpretation of customer's current life stage if the time where they were collected is taken into consideration.

For recommendations the paper proceeds in calculating the probability of a user buying a product for a specific stage. Denoting "a" the age of the baby, with j the item for which we want to

compute prediction for, $P(pproduct_j)$ the probability the customer with purchase the item and $P(a\ |pproduct_j)$ the probability of purchasing during age "a". This probability is calculated by:

$$P(pproduct_j, a) = P(a\ |pproduct_j)P(pproduct_j)$$

Then the products are ranked using the equation:

$$P(pproduct_j) \int p(a|pproduct_j)p_u(a)da$$

For the computation of $P(pproduct_j)$ the following logistic regression model is used:

$$P(pproduct_j) = \frac{1}{1 + e^{-w^T x}}$$

During evaluation the system was tested gradually adding the features presented in order to examine the impact each of them had on the accuracy of the algorithm. Item properties, customers search queries and product titles improved the accuracy of the algorithm yet slightly. Temporal effect of the features had the biggest impact on accuracy triggering an increase of more than 5%.

The proposed approach successfully incorporates time in the specific domain increasing the accuracy of the recommendations. The system similarly to other papers not only utilizes implicit information but also is based on the idea that customers preferences and needs change over time. It goes one step further implementing an association of the customers' needs with a specific event in his life.

3.1.8 Socially enabled Preference Learning from Implicit Feedback Data

In [20] the aim of the research is to implement a matrix factorization model on an implicit user item preference matrix enhanced with social information.

Notating users u, items i, x_u the user factor vector, y_i the item factor vector, a_{uk} is the weight of the influence of the friend k over the specific user and as S the social friendship matrix. Then for including the social network into the algorithm the cost function is given by:

$$\mathrm{F}_{ui} = x_u y_i + \sum_{k \in F_u} \frac{a_{uk}}{|\, S_u|}\, x_k y_i$$

Given this cost function the authors define an objective function with respect to user factor vector x, item factor vector y and the social influence weight as:

$$\min_{x,y,A} J = \sum_{(i,j) \in Y} c_{ui}\, (x_u y_i + \sum_{k \in F_u} \frac{a_{ui} x_u y_i}{|S_u|} - Y_{ui})^2 + \Omega_{x,y,A}$$

As A is notated the defined matrix

$$A_{uk} = a_{uk}\,, \forall u\ \forall k\ \in S_u, 0$$

As Ω is notated the regularized term given by:

$$\Omega_{x,y,A} = \lambda_1\, \|x\|_F^2 + \lambda_2 \|y\|_F^2 + \lambda_3 \|A\|_F^2$$

The c variable notated is a predefined weight which serves the purpose of adding weight to the loss function when computing over observed interaction.

Furthermore in the presented approach the Gauss-Siebel approach is being used for the optimization of the function. In each iteration two of the three matrixes are being fixed while the third is updated.

This proposed approach evaluated against four approaches. The Implicit Matrix factorization model described in [8] referred as iMF. The recommendation system proposed in [21] referred as LLA which takes into consideration social and contextual information which recommends friends and items to the user. Another social recommender described in [22] which penalizes the l2 distance between friends in the objective function. The fourth approach which used for evaluating is described in [23] and incorporates social trust information in an optimization process of a loss function regarding explicit feedback referred as Trust Ensemble.

The results of the evaluation showed that it outperformed all the algorithms except the classic implementation of Implicit Matrix Factorization. Thus regarding only social recommenders it did presented the best results.

3.1.9 A hybrid online-product recommendation system: Combining implicit rating-based collaborative filtering and sequential pattern analysis

In [24] the proposed system recognizes the problem that many ecommerce stores have regarding insufficient explicit information. The system is referred as HOPE and aims in constructing ratings for items based on transaction history of the users. Furthermore investigates the possible combination of collaborative filtering algorithm and sequential pattern analysis.

The method proposed for combining collaborative filtering and sequential pattern analysis is to calculate preference of a user regarding an item with CF and SPA separately and then proceed by calculating a weighted aggregation of the two results and creates the final prediction for a user item pair.

For the collaborative part of the approach the first task is to construct and user item rating matrix derived from the transaction history. The preference AP of a user u for an item i is given by:

$$AP_{ui} = \ln\left(\frac{t_{ui}}{T_u} + 1\right)$$

As T_u is notated the total number of transactions for a specific customer while a t_{ui} the number of transactions of a user which contain the item i. Due to the fact that the provided equation does not take into consideration various characteristics of the item that influence items purchase frequency. Moreover the number of transaction cannot provide a solid understanding of a user's preference since it can lead into misinterpretation if not compared to the behavior of other users. Thus the author proposes the user of relative preference RP as given by:

$$RP_{ui} = \frac{AP_{ui}}{\max_{c \in U}(AP_{ci})}$$

Where U denotes the users that purchased item i. Furthermore the final implicit rating is given by the following equation:

$$IR_{ui} = round\ (5\ RP_{ui})$$

In the final step the relative preference is multiplied and rounded up to create a rating set ranging from 1 to 5.

For measuring user similarity the proposed system proceeds in computation of Pearson correlation coefficient, cosine similarity and distance measure. The three approaches are evaluated to derive the algorithm that brings the best result in the specific domain and dataset. Then the system proceed in the prediction of the user preference using ratings from the k most similar users. The prediction is computed with:

$$CFPP_{ai} = \overline{R_a} + \frac{1}{\sum_{b=1}^{k} |\ sim(a,b)\ |} \sum_{b=1}^{k} sim(a,b)(R_{bi} - \overline{R_b})$$

In the equation is denoted with k the number of similar users used, which is constant tested in various values, a and b used to describe two different users and sim(a,b) is the similarity calculated in one of the three methods applied.

For sequential pattern analysis preference score the method considers the sequences of transactions of all users except the one calculating the prediction for. The derived sequences are compared to all subsequences of the target user in order to extract item recommendations. Calculation of the sequential pattern analysis preference score is done with the equation:

$$SPAPP_{ai} = \sum_{s \in SUB} Support_s^i$$

In this equation as SUB is denoted the set of the users subsequences and as Support the support of the item in the specific subsequence s.

The final preference score FPP for the user item pair is calculated using the normalized CFPP and SPAPP and the constant a as weight in the following manner:

$$FPP_{ui} = a\ NCFPP_{ui} + (1 - a)NSPAPP_{ui}$$

A series of experiments were conducted first the experiment concerned with discovering the more suitable similarity function for calculating user similarity during collaborative filtering part. Next experiments were conducted to discover the optimal minimum support to be used for sequential pattern analysis. The next experiment aimed in discovering the weight value that brings the best final results when imposed on the CF and SPA preferences score during final preference score computation. Having discovered the optimal values for the system HOPE was compared to Collaborative filtering and Sequential Pattern Analysis recommendations in precision, recall and F1.

The results reported by the experiments show that the use of implicit rating is an acceptable solution to insufficient explicit feedback. Furthermore the proposed implementation scored significantly higher in precision, recall and F1 though that is not case with accuracy comparison between HOPE and SPA recommendations.

3.1.10 Resolving data sparsity by multi-type auxiliary implicit feedback for recommender systems.

In [25] research is focused in implicit feedback information and the problem of data sparsity that collaborative filtering algorithms suffer from. Data sparsity problem regards dataset with a big number of users and items and small number of interactions between them. This problem hinders the accuracy of CF models especially neighborhood recognition approaches where in such cases are unable to accurately cluster customers and items.

In the specific paper aims to address the data sparsity problem by incorporating various implicitly gathered information as opposed to solely using purchases. The various implicit information are referred as multi-type auxiliary data. As auxiliary data are considered different ways that the user interacts with items. Search queries for items, viewing, add to wishlist, share are some examples of candidate auxiliary data.

The methodology at first adopts a regression model to examine the correlation of multi-type auxiliary data with the target data. As target data, or positive feedback, is referred the purchase of the item from the customer. In the next phase the neighbors of the current user are being selected. The selection is performed by comparing the created auxiliary feedback of each possible neighbor with the specific user's positive feedback. Last it is implemented a ranking model that incorporates both auxiliary and initial data.

Notating with A the auxiliary data, x_a the specific auxiliary piece of information, β_0 and β_α are the coefficients to learn and y is the label for the target feedback the logistic regression is formalized by:

(1)

$$y = \begin{cases} 1 \quad if\ \beta_0 + \sum_{a=1}^{|A|} \beta_a x_a > 0 \\ 0 \end{cases}$$

The data used for evaluating auxiliary data types come from Shobazaar and Xing. Shobazaar is a social fashion while Xing are gathered information regarding job ads users. In the case of Shobazaar eight different types of auxiliary data where evaluated, product clicked from archive, product details viewed in archive product marked as wanted from archive, pixel initialized, pixel order, pixel order without reference, product clicked in details, product marked as wanted. In Xing the three auxiliary types of data evaluated, clicked on a job posting, bookmarked a job, removed a job.

The results revealed strong correlation of two specific types of auxiliary data with the purchase of the item in both datasets. In Shobazaar clicking an item and pixel initialized are the ones strongly related. Similarly in Xing click a job posting and bookmark are also strongly correlated with target feedback. Another two types have positive correlation while some are not related in any way with purchase of an item in the case of Shopbazaar. In the case of Xing the remaining type of data which concerns the removal of job ad is associated negatively.

For generating the new feedback dataset from auxiliary information two ways are proposed. The first is by using a linear regression. Using (1) and the confidence calculated by the following equation.

$$c_{uj} = \beta_0 + \sum_{a=1}^{|A|} \beta_a x_a^{uj}$$

As x_a^{uj} is notated the score calculated from the user to the specific item using auxiliary information.

The second way is through multi-dimension similarity. In this approach each item is compared to the items purchased by the customer using cosine similarity.

In the next phase of the approach the paper proposes a model which incorporates auxiliary and original data for ranking the items. The approach is a variant of Bayesian Personalized Ranking model with Generated data and confidence. The model concludes in the following equation:

$$ln\ p(\theta \mid >_u) = \sum_{(u,i,j)\in D} c_{ui} ln\sigma(\hat{p}_{uij}) - \lambda_\theta \|\theta\|^2$$

With θ is notated the mode parameters, D is a set of triplets of user u purchased i but not j (u,i,j), $\hat{p}_{uij}$ is notated the ranking difference between the pairs of specific user u and item i,j and λ_θ are the regularization that prevents over-fitting of the model. For maximization the author proposed the minimization of the negative of the same equation via Stochastic Gradient Descent.

For the evaluation of the system the proposed approach were compared to with four other systems on both datasets. The results report that the GcBPR outperformed all the other approaches. The approach with the lowest score is the one without implementing auxiliary data at all. Notably the method outperforms significantly even the second best approach. The increase on average is up to 67.38% regarding Shobazaar dataset and 9.74% for Xing.

The proposed methodology presents an interesting approach regarding use the of implicit information firstly due to the fact that implements an evaluation of the implicit feedback against

the purchased items of a user where in most cases these systems aim to predict purchases. And because it demonstrates an effective way incorporating the auxiliary implicit feedback in a recommendation prediction model.

3.1.11 A Recommendation System using Association Rules and Collaborative Filtering

In this approach a hybrid approach for a recommendation system incorporating collaborative filtering as well as association rule mining algorithms. Furthermore the data used in the current research are also implicitly gathered. The goal is to effectively predict which sellers the customer is going to visit. Lastly as the recommendation engine is planned to run on mobile devices it incorporates the devices GPS and takes into consideration the proximity of the seller to the customer for the final recommendation.

During the first phase of the system the association rule mining part takes place. Using K-means based algorithm for rule mining predicts categories that might be of interest to the customer. The Apriori algorithm is used for extraction of relevant rules.

During the next phase the collaborative filtering part is implemented. In this phase the system recommends items unused by the customer. The prediction of CF is achieved in synergy with the rules derived from the previous phase. The similarity between users is computed in regard with their past choices on sellers. Furthermore Pearson correlation coefficient was used to measure similarity among users.

The ratings of similar users over unused by the specific user items are used for creating the recommendations. The presented implementation does not aim into improving the algorithm of recommender systems rather than implement a combination of the two approach in the context of a mobile app recommending sellers.

3.1.12 Collaborative filtering with initialized factor matrices

In [26] the paper aims in improving the accuracy of classic collaborative filtering algorithms by addressing the problem of data sparsity in datasets. More specifically it deals with the optimization of matrix factorization technique.

Matrix factorization aims at creating user and item latent factor matrices and use them to predict unknown values on a user item matrix. Approaches like Singular Value Decomposition require the missing values of the matrix to be filled with averages before creating the user item latent factor matrices. In other approaches such as the one used in our implementation and those in papers [8], [14] the approach of matrix factorization involves randomly initializing the user item factor matrices and implementing a repeated optimization method on the matrices. In each repetitions the matrices are being updated minimizing the error of a cost function.

In the later approach of matrix factorization the random initialization of the factor matrices is not optimal for the optimization method. The aim of the proposed approach is to present another way for initialization of factor matrices and thus lead to increased accuracy of the algorithm. For this task the implementation of SVD is proposed by the authors.

In the first phase of the methodology latent user and item factor matrices are extracted from the user-item matrix. The initialization method consists of the following steps. First additional values are being filled in the original user-item matrices where the rating is unknown. This is accomplished by using a method such as averaging the known user ratings for the specific item. There have been tested eight approaches regarding filling the unknown rating values prior to SVD. These include using the median rating of user, the median rating of item, the total median of items, the average of user and item median, the average ratings of the user, the average ratings of the item, total average of all ratings and the average from user and item averages.

For evaluation of the approach the proposed approach was tested against a randomly initialized matrices matrix factorization algorithm with the same optimization method. The show report that the method improved the system. Specifically it decreased the RMSE by 0.06. In addition the initialization of the factor matrices caused the optimization method to require less iterations to

find the matrices with the less error. Especially it is reported that for cases where the number of factors to discover is increased the proposed method was twice as fast compared to the non-initialized algorithm.

The approach describes a method of increasing accuracy and overall performance for discovering latent factor models. What it should be noted though is that taking average of ratings for filling the user-item matrix before SVD requires further examination and testing over other possible alternatives that could further increased performance.

4 Recommender System Design

4.1 Overview

Our approach concerns a product recommendation system in the context of e-commerce. It is a recommendation approach based solely on implicitly gathered information. The main technique used is a matrix factorization algorithm. In addition, association rule mining is used for optimization of the system, as well as providing extra recommendations in parallel to the main method [27].

4.2 Approach

4.2.1 Implicit Matrix Factorization

The main algorithm of our system is an implementation based on the work of [8] . It regards implicit feedback instead of commonly used ratings. For the implementation of the algorithm the raw data need to be transformed in a user-item matrix with values representing the implicit rating associated with the pair. As described our dataset provides observations over different visitors' behavior. Thus, the first stage of the algorithm involves preprocessing raw data to transform them in a user-item matrix.

In [8] the dataset used concerns visitor transactions and the user-item matrix required for matrix factorization is constructed by aggregating the number of times a customer viewed a specific show. A user's interest towards a show is inferred by the number of times he viewed the show. In our case we infer user interest towards an item from his behavior. The difference of our system is that instead of one measurement we consider three. Thus interaction score includes all three observations and is given by:

$$r_{ui} = v_{ui} + b_{ui} + t_{ui} \quad (1)$$

The number of times the visitor viewed the item is notated as v, the number of times the visitor added the item to his cart is notated as b and the number of times the visitor bought the item as t.

During the purchase process the visitor usually navigates through several items before making his decision. While browsing products it is possible that he views products that he will then exclude for a number of reasons, one being that he did not like them. Viewing items, even though being an indication that the visitor wanted to learn more about the specific item does not always mean that the customer prefers it. On the other hand, adding to the cart and transaction are strong indications of preference, since the visitor did actually decide to purchase them. Thus the three observations have different weight regarding visitor's preference over an item. Taking into consideration the above weighting of the observations is required the interaction score is transformed into:

$$r_{ui} = v_{ui} + b_{weight} * b_{ui} + t_{weight} * t_{ui} \qquad (2)$$

As t_{weight} we denote the transaction weight and as b_{weight} the "add to cart" weight. After validation we defined transaction weight to be equal to 100 while add to cart to be 50. It was considered not giving additional weight to the add to cart since for the customer to proceed to a transaction it is required to add the item to cart. It is included because the cases where the customer added the item to the cart but did not complete the transaction are also a strong indication of preference though not as strong as actually buying it.

Like the proposed algorithm the preference is a binary variable which takes the value of 1 for the cases where a user interacted with an item and 0 for the rest. With r_{ui} we notate the interaction score for a user u with an item i.

$$p_{ui} = \begin{cases} 1 & r_{ui} > 0 \\ 0 & r_{ui} = 0 \end{cases}$$

Also, in our case, it is needed to consider differences in preference level and so a confidence level is defined. Similarly, repetition of interaction is considered as a strong indication of preference. Confidence of each user item pair is given by:

$$c_{ui} = 1 + a(r_{ui})$$

A predefined constant used as scale regulator is notated as a. 40 was found after validation to perform well in our model.

The next phase of the algorithm after creating the user item matrix is calculating the user item factor vectors. For calculation of the vectors the system proceed in calculating the vectors by minimizing the cost function proposed in [8]:

$$\min_{x,y} \sum_{u,i} c_{ui}(p_{ui} - x_u^T y_i)^2 + \lambda \left(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2 \right)$$

As x is denoted the user factor vector and as y the item factor vector. The λ $(\sum_u \|x_u\|^2 + \sum_i \|y_i\|^2)$ is used to prevent overfitting.

After having calculated the factor vectors we can get a prediction for user item pair by calculating the dot product of their factor vectors.

$$rec_{ui} = x_u^T . y_i$$

rec_{ui} is the predicted preference score for a specific user towards a specific item. The systems proceeds in ranking all products for a user and a list ordered by preference can be extracted and used as recommendations.

4.2.2 Association Rules mining

Association rule mining aims to extract patterns from the user transactions. Many recommenders have been developed using association rules where they recommend items based on extracted rules with high Support and Confidence. Support is the number of times the rule applies against the total number of transactions. Confidence is the number of times the rule applies against the total number of transactions with the head item in them.

Association rules in our system are used as a standalone algorithm for creating recommendations, though they can also be used for enhancing a collaborative filtering algorithm.

For extracting association rules, the system uses the apriori algorithm. Moreover, it is able to extract three different kinds of rules, based on the data provided for the extraction.

In the first category, item level association rules, rules are being extracted based on item ids, treating each transaction separately. This method provides information about items usually sold together. Though this method does not provide many rules from the specific dataset we used, mainly due to the large number of items.

A second approach, user level association rules, involves extracting rules based on items ids but treating all transactions for each user as a single transaction. They provide information about items that are usually bought together throughout the whole purchase history.

The third approach, category level association rules, associates each item with its categories and extracts rules regarding categories. This method extracts information regarding categories usually bought together. These categories of rules are used externally to the collaborative filtering algorithm to provide recommendation for cases where the confidence level is high.

Using association rules, the transactions of the user are checked and if found not to satisfy rules or if the user adds to his cart a base item from a rule then the appended item can be his recommendation. In addition this feature can form static recommendations in the bottom of base products in the form of “usually bought together” items.

4.2.3 User-item matrix enhancement

One of the common problems that collaborative filtering algorithms face is data sparsity. Data sparsity problem occurs when there is a large number of users and items and users interact with only a small proportion of the items. Matrix factorization techniques aim to calculate user item factor vectors based on the user item matrix. The denser the matrix is the more accurate the results become.

An interesting implementation is described in [28] where association rule mining is used in order to make the user-item preference matrix denser by extracting rules and filling values accordingly.

Even though in the proposed implementation this method did not cause a significant improvement on accuracy, it is worth testing due to the fact that in our dataset we have actual transactions instead of ratings and views.

Association rules are utilized to fill in missing values on the user item matrix. For this approach the second category of user level association rules is used. The item level category of rules does not produce the necessary number of rules required to produce a significant change in the matrix and thus in the result. The category level produces category recommendations and, since the ratings needed to be filled concern items, this category does not serve the purpose. Even though they could apply an additional weight to the product belonging to the recommended categories based on user transaction, are not implemented on the current algorithm.

After extracting the association rules and the standard preprocessing of the raw data provides the user item matrix the algorithm proceeds with enhancing it. The system fills the matrix in specific user items where the user has bought the base item, but not the appended item. The score assigned to the previously unassigned user item pair should take into account the score of the base item. Thus, we could model the cases where the user bought the base item more than once. In addition, since the confidence provides information about the correlation between base and appended items this should also be considered in scoring. From the above, the score is equal to the score of base item times the confidence of the rule. Notating user with u, items with i, i_a are the appended items and i_b are the base items and nr is a newly assigned rating score that was previously absent for the specific user item pair.

$$nr_{ui_a} = r_{ui_b} * confidence$$

After enhancing the user-item score the system proceeds by discovering the user item factor vectors of the enhanced matrix and the rest of phases of the recommendation system.

4.2.4 Initializing user item factor vectors

As described earlier, the system implements a matrix factorization algorithm which aims at discovering the user and item factor vectors and use its dot product to produce recommendations. For discovering the factor vector it minimizes a cost function changing the factor vector with a step function in each iteration. For the above to be implemented an initial state of vectors is required. The proposed methodology in [8] uses random values for initialization of the vectors.

In [26] a way of increasing prediction accuracy by initializing the user and item vectors before minimizing the cost function is presented. Similar to the proposed implementation the system decomposes the user item matrix and extracts user and item vectors which are used as initial vectors during the optimization method of alternating least squares.

4.3 Dataset Information

The dataset used for the evaluation of the system is the Retailrocket recommender system dataset . Retail Rocket is a company that develops personalization technologies for their customers and through this way help them increase marketing level and bring them profit. The dataset consists of three different files each of which is associated with a different aspect of an e-commerce system.

Events.csv contains information regarding visitor behavior collected in a time period of 4.5 months. The monitored visitor behavior includes three events, “view”, “addtocart” and “transaction”. Events.csv contains 2756101 records, from which 2664312 concern views, 69332 depict add to cart event and 22457 transactions. The number of unique users in the dataset is 1407580. The columns in the raw file are timestamp which shows the Unix time that the event took place, visitorid which is a number that uniquely identifies visitors, itemid which is uniquely identifies items, event which provides the name of the event that corresponds to the row and can take one of the values, view, addtocart, transaction and transactionid which uniquely identifies each transaction, all column values are filled for each row except for the transactionid which is

only filled when then event column is equal to transaction. Five random rows of the file are presented in the following table.

timestamp	visitorid	event	Itemid	transactionid
1433221332117	257597	view	355908	
1433223236124	287857	addtocart	5206	
1433222276276	599528	transaction	356475	4000
1433221512084	1124962	view	213464	
1433221512427	1402325	view	20889	

This is our main file in our implementation due to the fact that contains all the necessary information in order to proceed with matrix factorization

item_properties.csv (part 1, part 2) contain information about the items. It consists of 20275902 rows regarding 417053 items and their various properties. The file has four columns, timestamp the unix time that the property was set, itemid, property which is the id that identifies the property and the value of the property. All values except the categoryid and available properties are anonymized in the dataset for privacy reasons. Five random rows of the file are presented in the following table.

timestamp	itemid	property	value
1433041200000	183478	561	769062
1431226800000	8921	categoryid	1188
1433041200000	352564	available	0
1436670000000	327059	663	1297729 n156.000 606827
1441508400000	77208	468	n12.000 272976

This file is very useful regarding content-based algorithms. Though in our current implementation was not extensively utilized. Our approach uses this file to extract association rules for the categories of the items.

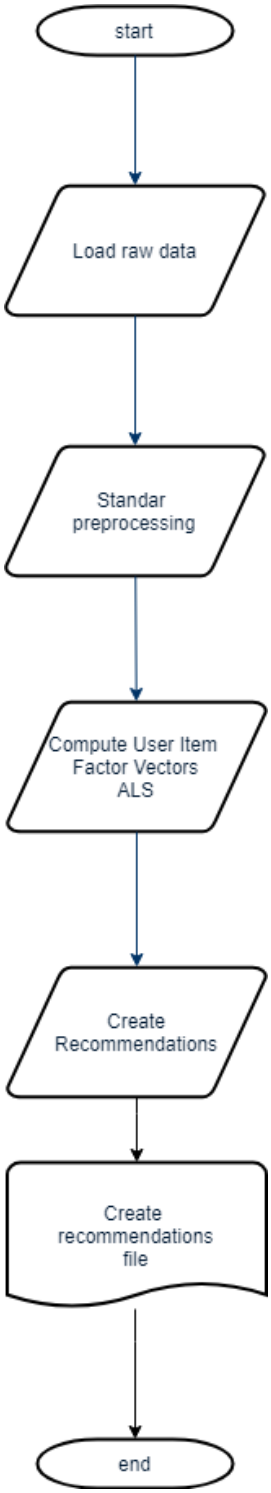
Category_tree.csv contains information regarding the associations between item categories. It consists of 16690 rows with two columns. Its row describes the parent-children relationship between two categories. Five random rows of the file are presented in the following table.

categoryid	parentid
1016	213
809	169
570	9
1691	885
536	1691

This information can be used in content based approaches or for higher level association rule mining.

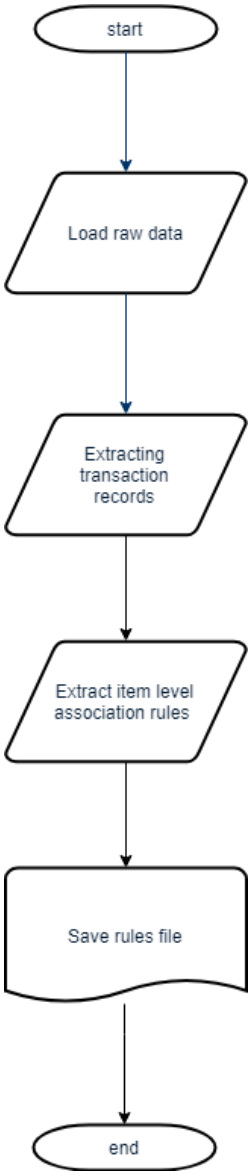
4.4 Flowchart diagrams

4.4.1 Implicit ALS main flow

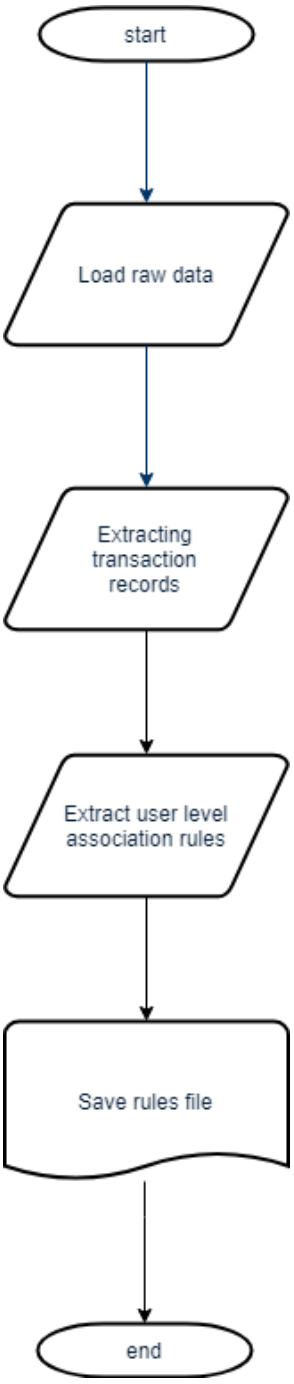


4.4.2 Association rules extraction diagrams

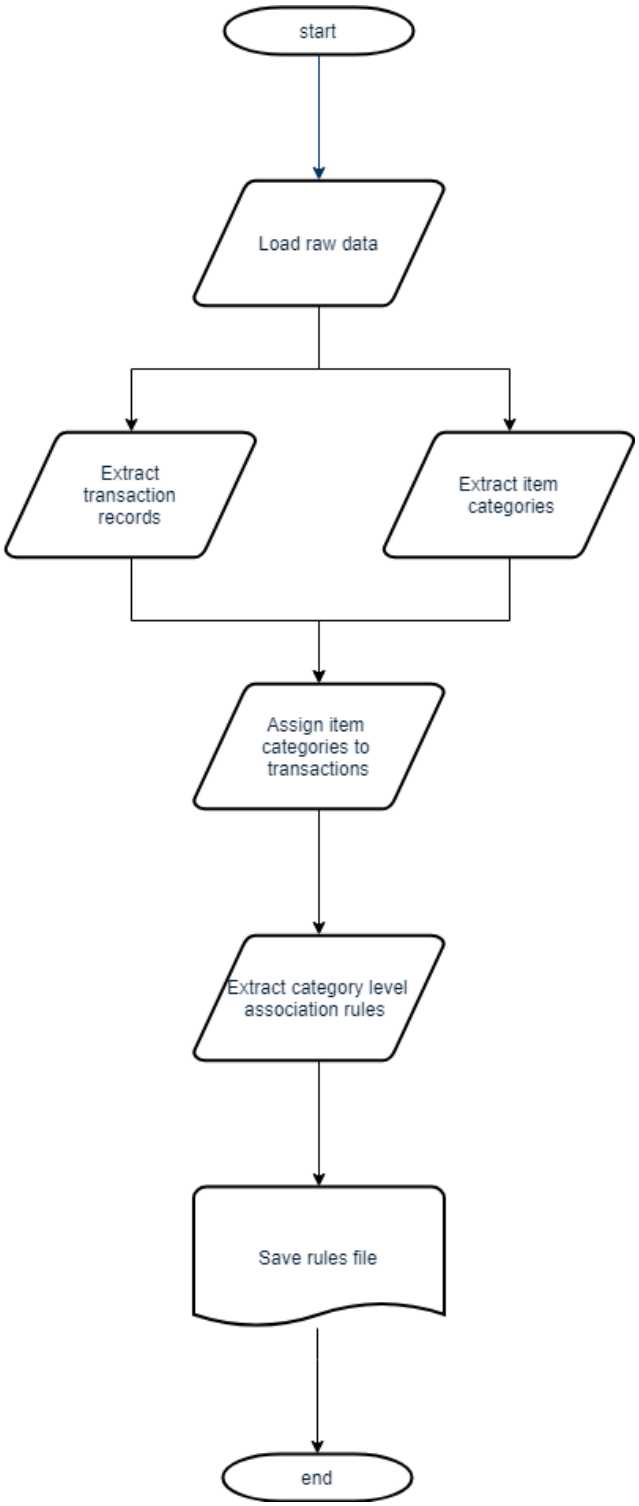
Item-level association rules



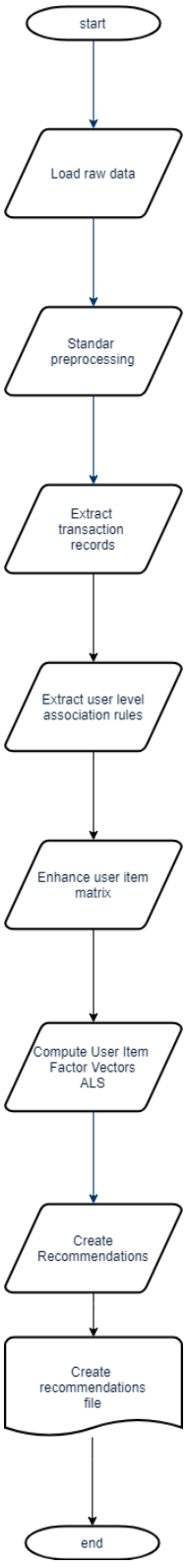
User-level association rules



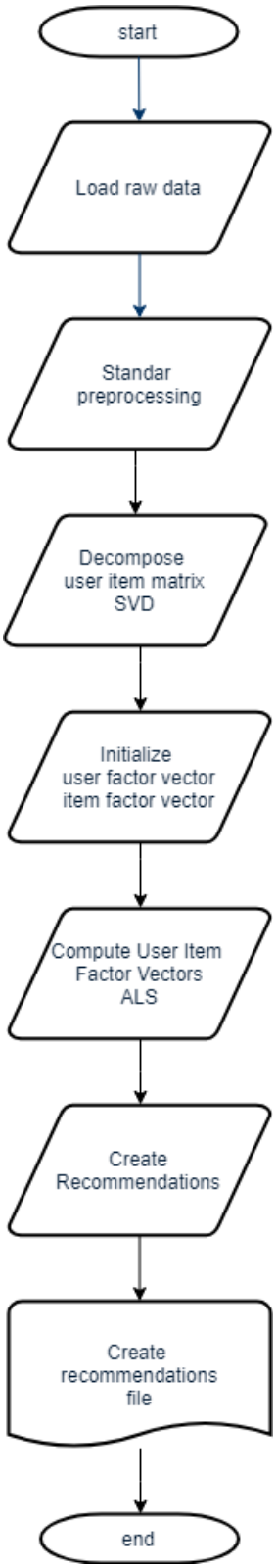
Category-level association rules



4.4.3 Enhanced user-item matrix



4.4.4 Initialized user and item factor vectors



4.5 Evaluation

Regarding the evaluation of the system mean percentile ranking is the quality measure of the system. For the evaluation the datasets were split in training and testing set according to time. The data consist of three months' worth of user behavior, for the training set we included all but the last week which we used as test set. The distinction of dates was made by converting the timestamp into the actual date time representation.

The variables we test in order to tweak the system are: a which is the proposed by the literature to be multiplied with the number of user item interaction in our data and form the preference score, the transaction weight which is used to model the difference in importance between views and transactions, the number of factors that the algorithm will use in order to calculate the vectors of the matrix and the iterations which the alternating least squares algorithm will use for the calculation.

5 Implementation of the System

The implementation was done in python v3.6.7rc1. The IDE used for the development of the algorithm is the PyCharm 2018.2 Community Edition. In total there were developed four different files each of which contributes to the overall flow of the algorithm. The files were divided based on their scope and specific purpose they serve. Moreover we make use of two external libraries implicit.py and apyori.py to aid with the implementation of the system.

5.1 Files

The main file is the main.py. The preprocessor.py file is where the preprocessing of our dataset is done in order to construct the user-item matrix. The arm.py is the file responsible for extracting the association rules needed by the system. Last erp.py holds the function responsible for the evaluation of the system.

5.2 External Libraries

5.2.1 Implicit.py

Implicit.py is a python library which provides implementations of various well known algorithms regarding recommendations on implicit feedback datasets. It includes the main flow of Alternating least squares which is the core algorithm of our implementation.

Other notable implementations included are the Bayesian Personalized Ranking and Item-Item Neighbour models with Cosine similarity, TFIDF or BM25.

5.2.2 Apyori.py

Apyori provides the system with a simple implementation of the Apriori algorithm. It used to extract the association rules in the various phases of the system that is required.

5.3 Functions

5.3.1 main.py

initialize_factor_matrices

This function is responsible for the initialization of the user and item factor vectors. It implements SVD algorithm on the user item matrix and set the initial vectors.

5.3.2 preprocessor.py

standar_preprocessing

This function is responsible for reading our dataset from the csv files. Next it proceeds in transforming the data in a better form. At this point the preference score is computed based on the monitoring of user behavior and proceeds in eradicating the different user actions. In addition splits the dataframes based on timestamp. Returns a dataframe for training and a dataframe for testing.

get_training_matrix

This function is the one responsible for returning the user item matrix. Moreover, it also returns separately user and item indices as well as the testing dataset. It is called the standar_preprocessing function. It is called in the beginning of the main.py file when the standard matrix is used for training.

get_all_purchases_rules_enhanced_training_matrix

This function is responsible for creating the enhanced user item matrix. It calls the standar_preprocessing function to receive the initial user item matrix. Next calls the apyori_rules_all_purchases function of the arm.py file and receives the user-level association rules. Then it proceeds with discarding the rules with 100% confidence and fills with the rest the user item matrix where necessary. Last likewise get_training_matrix returns the enhanced user-item matrix, the user and item indices separately as lists and the training matrix. It is called in the beginning of the main.py file when the enhanced matrix is used for training.

5.3.3 arm.py

get_transaction_records

This function is responsible to read the csv files and create the transaction dataframes. First reads the events csv and keeps only the information about transactions. Next it reads the item properties files and assigns to each transaction a list of category ids that the purchased items belong to. Next creates a copy of the transaction dataframe in which the transactions are grouped together based on the userid associated with them. Then it returns a dataframe which consist of transactions ungrouped and the item ids and category ids of each transaction and one dataframe which consists of each users overall transactions grouped and the ids of the items they purchased. The function is called by the apyori_rules_items, apyori_rules_categories, apyoti_rules_all_purchases for extraction association rules.

apyori_rules_items

This function is responsible for returning a list of the item-level association rules extracted from the transactions. It calls the get_transactions_records and with the returned records creates a list of the item ids in the transactions which feeds to the apriori function of the apyory library. Apriori function provides with the association rules which next are converted to a list and returned.

apyori_rules_categories

This function is responsible for returning a list of the category-level association rules extracted from the transactions. It calls the get_transactions_records and with the returned records creates a list of the category ids in the transactions which feeds to the apriori function of the apyory library. Apriori function provides with the association rules which next are converted to a list and returned.

apyori_rules_all_purchases

This function is responsible for returning a list of the category-level association rules extracted from the transactions. It calls the get_transactions_records and with the returned user grouped records creates a list of the item ids in the transactions which feeds to the apriori function of the apyory library. Apriori function provides with the association rules which next are converted to a list and returned.

5.3.4 erp.py

save_results

This functions is responsible for saving the recommendations resulted by the algorithm. It is in the main.py file each time the recommend function returns the predicted recommendations for a specific user. The function converts and saves the recommendations as csv file in the results folder.

get_mean_percentile_rank

This function is responsible for calculating the percentile ranking of the predictions. It is called in the main.py file each time the recommend function returns the predicted recommendations for a specific user and after the save_results function has been called. It reads the saved recommendations for a specific user from the result folder and proceeds in calculating the mean percentile ranking of the specific user which then returns.

5.3.5 External functions

AlternatingLeastSquares (implicit library Class)

AlternatingLeastSquares is the function used by the implicit library and implements the optimization of the cost function. It is called in main.py file after the get_training_matrix or the get_all_purchases_rules_enhanced_training_matrix. It provides the necessary functions for ALS

fit (implicit library)

The fit is the main function of AlternatingLeastSquares class and implements the optimization method on the user item matrix. It is called in the main.py file after the get_training_matrix or the get_all_purchases_rules_enhanced_training_matrix which provide the training matrix or after the initialize_factor_matrices function which initializes the user factor vector and item factor vector with the decomposed user-item matrix after SVD.

recommend (implicit library)

The recommend function is responsible of returning a list of items, for specific user, ranked based on the dot product of the discovered user and item factor vectors. It is called in the main.py file after the fit function has completed computing the factor vectors.

6 Evaluation and Future Work

6.1 Evaluation

For the evaluation of the system we tested the algorithm on various transformations of the user-item matrix. Since negative feedback for the recommended items is not in the scope of our work,

accuracy metrics are not suitable for the evaluation of the system. Thus, the metric used for evaluating the quality of the proposed methodology is the Mean Percentile Ranking (MPR).

The first thing that needs to be evaluated is if the algorithm performs better considering all event types or taking into account only transaction events. Thus two user item matrices are constructed using the proposed methodology, one constructed based solely on transactions and one including all three interactions. We refer to the user-item matrix that includes all interactions as multi-type MT matrix and the one that contains only transactions as single-type matrix ST.

Furthermore, the algorithm is evaluated on how well it performs when applying methods for increased accuracy. Thus the tests on each of the previous user-item matrices implement four different methods, standard method, enhanced by use of association rules method, by initializing the user item factor vectors method and by enhancing the matrix as well as initializing the factor vectors method. With 4 different methods on two user item matrices we calculate the mean percentile ranking for eight different cases.

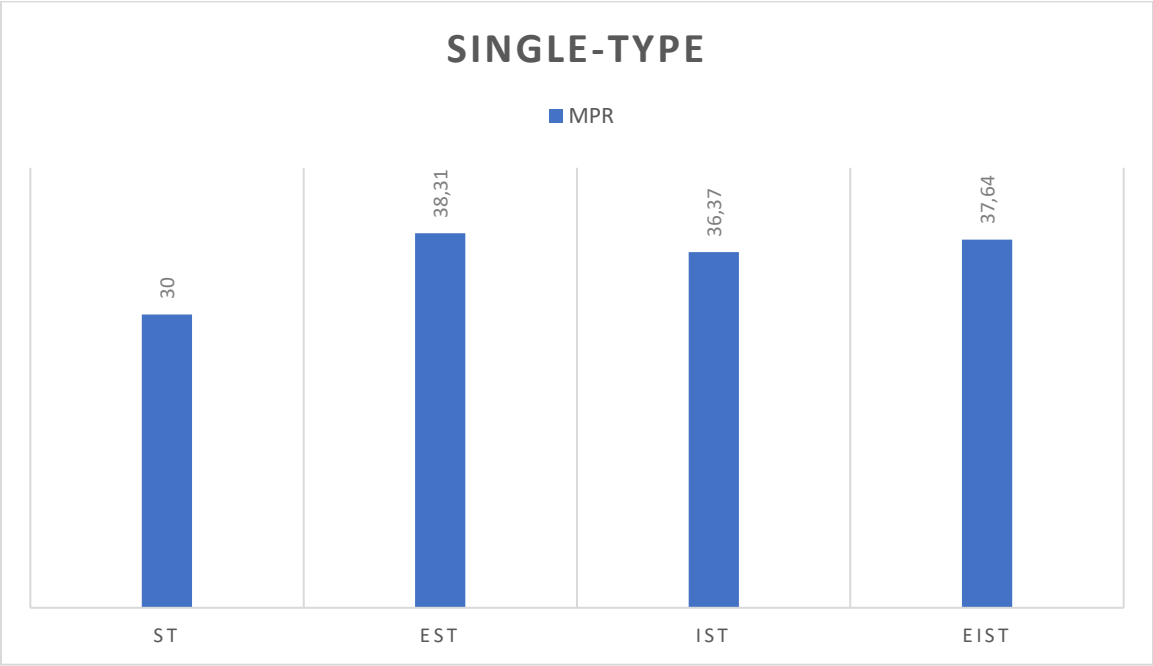
For association rule enhancing the user level category of extracted rules were used as described in the presentation of the design. In total 23 rules have been extracted using minimum confidence of 0.5. Implementing changes where it was necessary lead to 44 changes in the initial user-item matrix.

For initialization of the factor vectors the SVD decomposed the initial user-item matrix without any prior filling of missing values.

In addition, on both matrices, visitors who have not viewed more than 5 items were deemed as unnecessary noise and removed from the training set since the information provided is too little to produce meaningful recommendations. In addition this serves the purpose of reducing the total size of the evaluated training set and reducing its sparsity. Even though this hinders the systems recommendation coverage, addressing the cold start issue is out of scope of this dissertation.

6.1.1 Single-type evaluation with transaction events

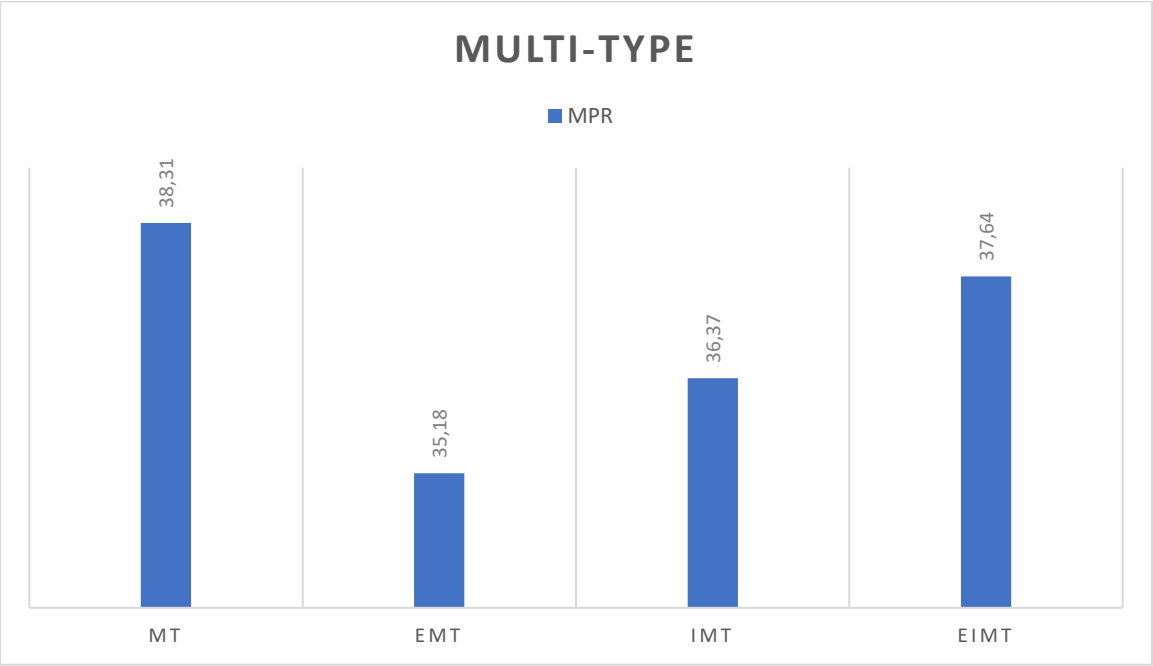
single-type	**ST**
enhanced single-type	**EST**
initialized single-type	**IST**
enhanced & initialized single-type	**EIST**



In this case the standard methodology provides the best results. Unexpectedly all the applied methods aiming at increasing the accuracy of the system actually hindered its accuracy greatly. The assumption to these results is that in this case the user-item matrix uses a small number of users and items and is dense enough for the algorithm to perform well, in which case any attempt to add values or initialize the factor vectors adds noise to the matrix.

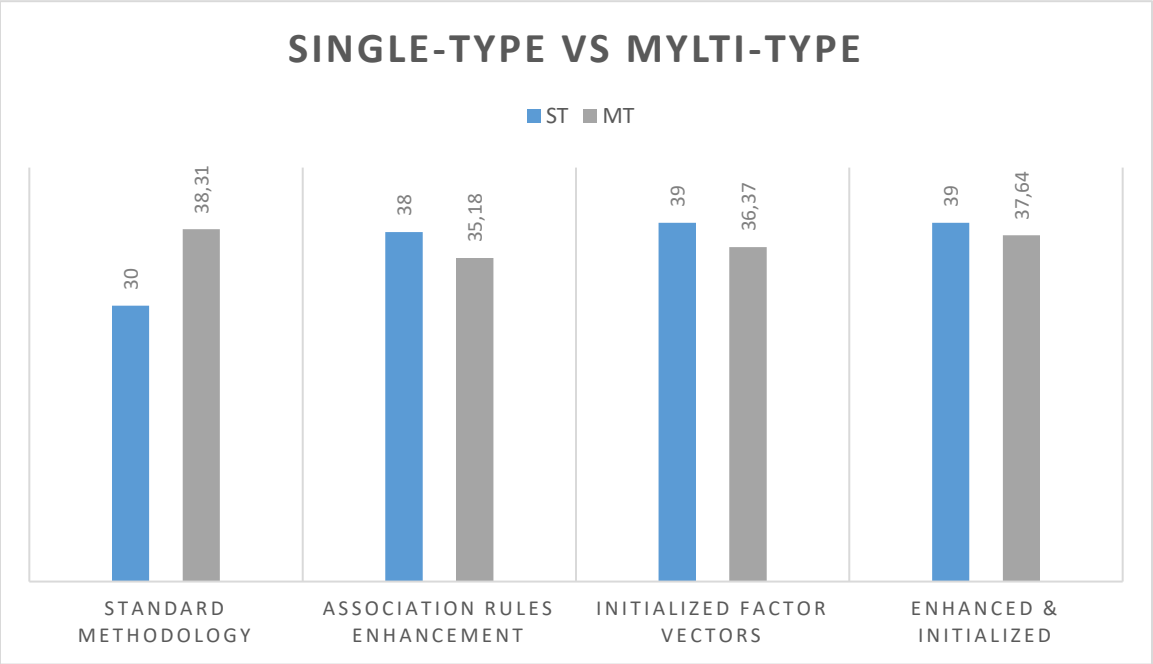
6.1.2 Multi-type evaluation with transactions, views and addtocart events

multi-type	**MT**
enhanced multi type	**EMT**
initialized multi-type	**IMT**
enhanced & initialized multi-type	**EIMT**



In this case all of the applied methods for increased accuracy actually did increase the performance of the algorithm. Notable association rules enhancing introduced the largest improvement of 3.13%. Following association rules initialization of the factor vectors improved the standard methodology by nearly two percent, 1.94 precisely. Though it was assumed that implementing both enhancement and initialization would improve performance even further that is not the case. Combination of both improvement methodologies improved the algorithm by a mere 0.67%.

6.1.3 Single-type vs multi-type comparison



By comparing the results considering single-type and multi-type matrices it is notable that single-type user-item matrix performs a lot better than any other type of matrix evaluated. Specifically compared to the best performing case of multi-type matrices, the association rules enhanced multi-type matrix, it still performed 5.18% better.

The reason that the single-type performs better is not exclusively that it is a better model. The main difference in the two user-item matrices is data sparsity. Sparsity greatly affects matrix factorization techniques. In both matrices users with only a few interactions have been eliminated. This resulted in a denser single-type matrix. More specifically the multi-type matrix has a sparsity degree of 0.01%, which means that only the 0.01% of user-item pairs have ratings. On the other hand single-type matrix has a sparsity degree of 1% which means that only 1% of all possible user-item ratings are filled. Even though single-type matrix is also too sparse it is ten times denser than the multi-type matrix.

6.1.4 Coverage

In contrast to accuracy a quality measure often required in recommendation systems is recommendations coverage. In terms of coverage the single-type matrix can create recommendations for 1056 users on 106 different items. In contrast the multi-type matrix is able to recommend 16022 different items to 43827 users. In terms of coverage the multi-type matrix is more suitable for creating recommendations.

6.2 Conclusions

The conclusions of this dissertation project are categorized regarding the multi-type or single-type used, the association rules extracted, the method followed enhancing user-item matrix with association rules and the method followed for initialization the factor vectors.

Regarding the use of different types of implicit information with the implemented method the results show that a lot of noise is added hindering the accuracy of the algorithm. Though the predictions are far from random, thus considering also the difference in coverage between single-type and multi-type (14966 more users and 15916 more items in multi-type) deems the multi-type a promising approach which requires further research for finding a more suitable approach to include these information.

Regarding the association rule mining, three different categories of extracted rules were introduced. Item level rules which extracted 23 rules, User level rules which also extracted 23 rules and Category level which extracted 173 rules. Category rules seems the most promising since it models the associations between product categories and can be used in real time recommendations based on currently added cart items.

User level categories are the most suitable of the three for the enhancement of the user-item matrix since they concern products and model better the association between them in the long-term. Even though the number of extracted rules is very small for addressing the data sparsity on

their own, the results of system evaluation report that they can benefit matrices with data noise such as multi-type matrix in the described methodology.

Initialization of the factor vectors similar to association rules benefit matrices with noise and can improve their accuracy. Though further research is required regarding the decomposition of the matrix as to discovering the best approach for filling the user-item matrix from which the initial vectors will be extracted.

6.3 Future Work

Future research should examine different ways of including different implicit information in the user-item matrix and best performing models. The literature for predicting future transactions proposes treating other implicit information as auxiliary data and implements a regression model for discovering the impact they pose in the final transaction [25].

Implicit datasets that offer transaction history provide opportunities for sequence analysis and rule based recommenders in general. In this dissertation we considered only association rules and let sequential pattern analysis as one of the best next steps for improving the algorithm.

For accuracy improvement the initialization of the factor vectors should be further researched. Specifically, an adequate way of filling the initial matrix prior to decomposition should be discovered. Literature proposes averages proved to improve accuracy greatly. In addition, other ways for singular value decomposition should be examined.

The data sparsity issue should be examined further. Association rules alone did not manage to increase the matrices significantly. One possible way could be via product similarity based on item properties.

Information about products is also available in the dataset giving ground to the implementation of Content-based filtering techniques. A possible next step could be implementing dynamic user profiling based on properties of products bought, viewed or added to cart. Additionally, since a

basic criterion during purchase decision is item price, the impact of pricing on the users preferences should be included in future research.

Similarly a lot of products are suitable only for specific time periods such as seasons or a specific age. Thus, timing should also be considered in product recommendation research. Due to the fact that often implicit dataset include a huge amount of information research on implementing such a model in distributed systems should be considered.

References

[1] C. O. Riordan and H. Sorensen, "Information Filtering and Retrieval: An Overview," 12 2000.

[2] F. Ricci, L. Rokach, B. Shapira and P. Kantor, Recommender Systems Handbook, 2011.

[3] S. Sivapalan, A. Sadeghian, H. Rahanam and A. Madni, *Recommender Systems in E-Commerce,* 2014.

[4] B. Smith and G. Linden, "Two Decades of Recommender Systems at Amazon.com," *IEEE Internet Computing,* vol. 21, pp. 12-18, 5 2017.

[5] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. V. Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, B. Livingston and D. Sampath, *The YouTube video recommendation system,* 2010, pp. 293-296.

[6] C. A. Gomez-Uribe and N. Hunt, "The Netflix Recommender System: Algorithms, Business Value, and Innovation," *ACM Trans. Management Inf. Syst.,* vol. 6, pp. 13:1-13:19, 2015.

[7] O. Nalmpantis and C. Tjortjis, "The 50/50 Recommender: a Method Incorporating Personality into Movie Recommender Systems," *CCIS Communications in Computer and Information Science*, pp. 498-507, 2017.

[8] Y. Hu, Y. Koren and C. Volinsky, "Collaborative Filtering for Implicit Feedback Datasets," in *Proceedings - IEEE International Conference on Data Mining*, 2008.

[9] G. Jawaheer, M. Szomszor and P. Kostkova, "Comparison of implicit and explicit feedback from an online music recommendation service," 1 2010.

[10] M. J. Pazzani and D. Billsus, "Content-Based Recommendation Systems," in *The Adaptive Web: Methods and Strategies of Web Personalization*, P. Brusilovsky, A. Kobsa and W. Nejdl, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 325-341.

[11] J. B. Schafer, D. Frankowski, J. Herlocker and S. Sen, "Collaborative Filtering Recommender Systems," in *The Adaptive Web: Methods and Strategies of Web Personalization*, P. Brusilovsky, A. Kobsa and W. Nejdl, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 291-324.

[12] J. S. Breese, D. Heckerman and C. Kadie, "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," in *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, San Francisco, CA, USA, 1998.

[13] Y. Koren, R. Bell and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," *Computer*, vol. 42, no. 08, pp. 30-37, 2009.

[14] C. C. Johnson, "Logistic Matrix Factorization for Implicit Feedback Data," 2014.

[15] L. Baltrunas, "Towards Time-Dependant Recommendation based on Implicit Feedback," 2009.

[16] S. A. Bahrainian, S. M. Bahrainian, M. Salarinasab and A. Dengel, "Implementation of an Intelligent Product Recommender System in an e-Store.," Toronto, 2010.

[17] R. Dutta and D. Mukhopadhyay, "Offering A Product Recommendation System in E-commerce," 9 2011.

[18] B. Pradel, S. Sean, J. Delporte, S. Guerif, C. Rouveirol, N. Usunier, F. S. Fogelman and F. Dufau-Joël, "A case study in a recommender system based on purchase data," 2011.

[19] P. Jiang, Y. Zhu, Y. Zhang and Q. Yuan, "Life-stage Prediction for Product Recommendation in E-commerce," 2015.

[20] J. Delporte, A. Karatzoglou, T. Matuszczyk and S. Canu, *Socially Enabled Preference Learning from Implicit Feedback Data*, 2013.

[21] S.-H. Yang, B. Long, A. Smola, N. Sadagopan, Z. Zheng and H. Zha, "Like Like Alike: Joint Friendship and Interest Propagation in Social Networks," in *Proceedings of the 20th International Conference on World Wide Web*, New York, NY, USA, 2011.

[22] H. Ma, D. Zhou, C. Liu, M. R. Lyu and I. King, "Recommender Systems with Social Regularization," in *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, New York, NY, USA, 2011.

[23] H. Ma, I. King and M. R. Lyu, "Learning to Recommend with Social Trust Ensemble," in *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, NY, USA, 2009.

[24] K. Choi, D. Yoo, G. Kim and Y. Suh, "A hybrid online-product recommendation system: Combining implicit rating-based collaborative filtering and sequential pattern analysis," *Electronic Commerce Research and Applications*, vol. 11, pp. 309-317, 2012.

[25] G. Guo, H. Qiu, Z. Tan, Y. Liu, J. Ma and X. Wang, "Resolving Data Sparsity by Multi-type Auxiliary Implicit Feedback for Recommender Systems," *Knowledge-Based Systems*, vol. 138, 10 2017.

[26] M. Nasiri and B. Minaei, "Increasing prediction accuracy in collaborative filtering with initialized factor matrices," *The Journal of Supercomputing*, vol. 72, 5 2016.

[27] I. Schoinas and T. Chirstos, "MuSIF: A Product Recommendation System Based on Multi-source Implicit Feedback," in *15th Int'l Conf. on Artificial Intelligence Applications and Innovations (AIAI 19) Springer*, 2019.

[28] T. Osadchiy, I. Poliakov, P. Olivier, M. Rowland and E. Foster, "Recommender system based on pairwise association rules," *Expert Systems with Applications*, vol. 115, 8 2018.

[29] B. Schafer, J. Konstan and J. Riedl, "Recommender Systems in E-Commerce," *1st ACM Conference on Electronic Commerce, Denver, Colorado, United States*, 10 1999.

[30] S. Pathan, K. Panjwani, N. Yadav, S. Lokhande and B. Thakare, "Product Recommendations System Survey," *International Journal of Computer Applications,* Vols. 131-9, p. 3, 2015.

[31] G. Prashanti and K. Narendra, "E-Commerce for Cold Start Product Suggestion using Micro Blogging Data Through Connecting Social Media," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT),* vol. 4, no. 2, pp. 24-30, 2018.

[32] A. Karageorgos, A. Dimitra, C. Tjortjis and G. Ntalos, "Agent-Based Digital Networking in Furniture Manufacturing Enterprises," Prague, 2010.

[33] F. Karimova, "A Survey of e-Commerce Recommender Systems," *European Scientific Journal,* vol. 12, 2016.

[34] S. Sterlin, A. Sandhya, S. S. Merlin and B. B. Sam, "A review on e-commerce recommender applications," 2017.

[35] C. Xu, "A novel recommendation method based on social network using matrix factorization technique.," *Information Processing & Management,* vol. 54, no. 3, pp. 359-474, 2018.

[36] Y.-H. Lee, P. J.-H. Hu, T.-H. Cheng and Y.-F. Hsieh, "A cost-sensitive technique for positive-example learning supporting content-based product recommendations in B-to-C e-commerce," *Decision Support Systems,* vol. 53, no. 1, pp. 245-256, 2012.

[37] G. Takács, I. Pilászy and D. Tikk, "Applications of the Conjugate Gradient Method for Implicit Feedback Collaborative Filtering," in *Proceedings of the Fifth ACM Conference on Recommender Systems*, New York, NY, USA, 2011.