INTERNATIONAL
HELLENIC
UNIVERSITY

Sentiment analysis applied on a book recommendation system

Belogianni Paraskevi

SID: 3308190003

Supervisor: Prof. Christos Tjortjis

Supervising Committee Members: Assoc. Prof. Apostolos Papadopoulos

Dr. Leonidas Akritidis

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Data Science

JANUARY 2021

THESSALONIKI – GREECE

INTERNATIONAL
HELLENIC
UNIVERSITY

Sentiment analysis applied on a book recommendation system

Belogianni Paraskevi

SID: 3308190003

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Data Science

JANUARY 2021

THESSALONIKI – GREECE

Abstract

This dissertation is the final part required for the completion of the MSc in Data Science at the International Hellenic University (IHU). The main goal of this study is to develop a book recommendation system based on collaborative filtering. Furthermore, a comparison is held between the performance of the system when the recommendations are based on simple ratings and when they are based on the result of sentiment analysis over customer comments. The algorithms examined are based on the KNN methodology and Singular Value Decomposition (SVD). The technique used to conclude in the algorithm with the best performance and generalization was cross-validation and the metrics that were examined were precision and recall.

Our findings show that the utilization of user comments as the input in our system instead of ratings, results in an increase in precision and a decrease in recall. For a book recommendation system, precision can be a more important indicator as it refers to the number of relevant recommendations which, eventually is the purpose of such a system. In particular the best performance was achieved with a 10-fold cross validation for the top 10 suggested items and resulted in an increase of 10% in the prediction score. As mentioned already the was a simultaneous decrease in the recall score which also influenced the F-score negatively and in particular it led to a decrease of 0.06%.

Paraskevi Belogianni

1/1/2021

Acknowledgements

I would like to grasp the chance and thank my supervisor Prof. Christos Tjortjis for the constant support and guidance. His ideas and additions were very constructive during the writing of this dissertation. In addition, I would like to thank the International Hellenic University, that gave me the opportunity to select and complete this dissertation. Finally, I am more that grateful to my family and friends who supported me all this time, especially under the critical and stressful moments we are facing due to the pandemic of Covid-19.

Contents

Abstract…… 3
Acknowledgements……..…. 4
Contents…….…. 5
Table of captioned objects…… 6
Chapter 1: Introduction 8
1.1 How and when the recommendation systems started 8
1.2 What is a recommendation system? 9
1.3 Fields of implementation 10
1.4 Benefits of RS 12
1.5 Open Challenges 13
1.6 Dissertation outline 14
Chapter 2: Background 16
2.1 Types of RS 16
2.1.1 Content based filtering 17
2.1.2. Collaborative Filtering 18
2.1.3 Knowledge based filtering 19
2.1.4 Hybrid model 20
2.2 Types of input data 21
2.3 Similarity measures 22
2.4 Evaluation of recommendation systems 23
Chapter 3: Literature Review 30
3.1 Background and examples of online systems and engines for book recommendations 30
3.2 Related work 30
3.2.1 Building a Book Recommender system using time-based content filtering 30
3.2.2 A graph-based recommender system for digital library 31
3.2.3 Book Recommendation System through content based and collaborative filtering method 31
3.2.4 Hybrid attribute and personality-based recommender system for book recommendation 33
3.2.5 Web-based personalized hybrid book recommendation system 33
3.2.6 An Implicit Feedback model for Goodreads Recommendations 35

3.2.7 Recommending user generated item lists 36
3.2.7 Hierarchical Gating Networks for Sequential Recommendation 37
3.2.9 The 50/50 Recommender: a Method Incorporating Personality into Movie Recommender systems 39
3.2.10 Promoting Diversity in Content Based Recommendation using Feature Weighting and LSH 39
3.2.11 MuSIF: A product Recommendation System Based on Multi-source Implicit Feedback 40
Chapter 4: Dataset 42
4.1 Dataset Description 42
4.2 Dataset Preprocessing 43
Chapter 5: Methodology 45
5.1 Surprise Library 45
5.2 SVD algorithm 46
5.3 K-NN algorithm 48
5.4 Grid search for hyper-parameter selection 50
Chapter 6: Results and Evaluation 58
a) user ratings 58
b) ratings of sentimental analysis 60
Chapter 7: Conclusions & Future Work 64
References 66

Table of captioned objects

Equation 1: Cosine similarity.....
Error! Bookmark not defined.
Equation 2: Pearson similarity.....
Error! Bookmark not defined.
Equation 3: Jaccard similarity.....
Error! Bookmark not defined.
Equation 4: Precision.....
Error! Bookmark not defined.
Equation 5: Recall.....
Error! Bookmark not defined.
Equation 6: F1-Score.....
Error! Bookmark not defined.
Equation 7: True positive rate.....
Error! Bookmark not defined.

Equation 8: False positive rate……
Error! Bookmark not defined.
Equation 9: Mean Absolute Error……
Error! Bookmark not defined.
Equation 10: Mean Squared Error……
Error! Bookmark not defined.
Equation 11: Root Mean Squared Error……
Error! Bookmark not defined.
Equation 12: Spearman correlation coefficient……
Error! Bookmark not defined.
Equation 13: Alternative Least Square Algorithm…… 33
Equation 14: KNN (basic) predictions…… 46
Equation 15: KNN (with Z-score) Predictions…… 47
Equation 16: KNN (with Means) Predictions…… 47
Equation 17 : Mean squared difference…… 50

Table 1: Differences of Recommendation Systems 20
Table 2: Computational time of all algorithms (a) 54
Table 3: Computational time of all algorithms (b) 56
Table 4: Precision results (a) 58
Table 5: Recall results (a) 58
Table 6: Precision results (b) 60
Table 7: Recall results (b) 60

Figure 1: Architecture of a Recommendation System [12] 10
Figure 2: Evaluation results…… 33
Figure 3: Comparative results for the proposed models…… 35
Figure 4: Recall@k and NDCG@k results 38
Figure 5: Preview of the first 5 rows of the dataset 43
Figure 6: Rating's distribution 43
Figure 7: Percentage distribution of the ratings 44
Figure 8: Grid search results and computational time for SVD algorithms (a) 50
Figure 9: Grid search results and computational time for KNN algorithms (a) 51
Figure 10: Grid search results and computational time for SVD algorithms (b) 52
Figure 11: Grid search results and computational time for KNN algorithms (b) 53
Figure 12: 5-fold cross validation results (a) 54
Figure 13: RMSE plot (a) 55
Figure 14: MAE plot (a) 55
Figure 15: 5-fold cross validation results 56
Figure 16: RMSE plot (b) 56
Figure 17: MAE plot (b) 57
Figure 18: SVD precision and recall plot (5-fold cross validation) (a) 59
Figure 19: SVD precision and recall plot (10-fold cross validation) (a) 59
Figure 20: SVDpp precision and recall plot (5-fold cross-validation) (b) 61
Figure 21: SVDpp precision and recall plot (10-fold cross-validation) (b) 61

Chapter 1: Introduction

The rapid growth of the Internet in the 21st century has led to major changes in all aspects of everyday life. Every day more than 2.5 quintillion bytes of data are created providing huge amounts of various information. Furthermore, the technological advance has given the possibility, even to regular users, to access any kind of information from many different devices, at any time.

More and more people use the World Wide Web in order to receive updates about events, work from home, entertain themselves, and purchase products online. During 2020, the pandemic of Corona-virus forced, even more, the digitalization of things since there was a need for completing almost every procedure without the physical presence of people. Even though access to any kind of information is facilitated, the huge amounts of data create a more severe problem, which is the choice of the correct information. This phenomenon is well known as “Information Overload”.

The urgent need for appropriate results enhanced the introduction of Recommendation Systems. They are considered one of the most powerful digital tools of the century. They appear all around the web and find application in many fields such as on-line retail, entertaining platforms, informative blogs, and newspapers, etc. The system acts as a guide for the average user and focuses on proposing the right content at the right time. The item of the proposal can be almost anything, from a music song or a video to a new product. That way the user does not have to spend time and effort in evaluating every information in order to conclude to the one he really needs.

1.1 How and when the recommendation systems started

The systems originated in the 70s, and two references exist for early recommender systems. The first one derives from Duke University and employs a distributed discussion system about newsgroup named “Usenet”. In this system, the users could make posts that were classified into newsgroups and even in sub-categories, if necessary. The second one refers to a recommender system named “Grundy”. It was implemented for book recommendations to people that have been already organized in categories according to their stereotypes.

Later, in the 90s, content filtering made its appearance to fulfill the need for large-scale information systems. Afterward, in 1992 the first system of collaborating filtering named “Tapestry” was designed by Xerox Palo Alto. This system permitted

the search of documents in a database based on their content, as well as, on other users' reactions. For example, the system could retrieve all the documents that include the word "car" and user A has considered "fast". It was based on the opinion that if the users are involved in the process, the information filtering would be more effective. Unlike other recommender systems of that time, "Tapestry" based its proposals not only on the evaluation of an item when it arrived, but, also on the humans' perspective about it.

Later, in 1994, the first recommender system that used rating data appeared with the name "Grouplens", and, in 1997, "Movielens" became the first movie recommender system, which, also, provided a very important dataset for the research community. "Grouplens" goal was to make suggestions about Usenet articles based on other people's ratings. This way the system could identify people with the same taste based on their likes and dislikes. Afterward, it proposed articles to the user that people with the same taste also liked. This is well known as the near neighbor technique.

In general, after the 90s many recommender systems were introduced such as "Firefly" for music fans, "RINCO" for music recommendations based on social information filtering, "Fab" which allowed users to create content-based filters, and many more. Gradually, the recommender systems gained popularity in the field of commerce and marketing intending to increase revenue from sales, and facilitate the user experience by reducing the effort and time spent. Then, the Amazon recommender system was created and was based both on collaborative and content-based filtering as well as input from the user via web browsing.

One of the greatest revolutions in the field was in 2006 when Netflix released an open competition for the best collaborating filtering model. This model should be able to use previous user ratings and provide predictions about film ratings. In 2009 the BellKor's Pragmatic Chaos team won the prize of 1000000 dollars achieving a 10.06% prediction on the pre-existing Netflix algorithm [6].

From 2010 and onwards there are many systems-oriented in social media, human decision making, personalization for mobile services, and information heterogeneity.

1.2 What is a recommendation system?

A Recommender System (RS) is a tool that tries to predict the user's preferences and make suggestions. It is considered to be part of the general concept of information filtering since it combines all information available at a time to propose "items". With the term "item" we refer in a more abstract way to the produced recommendations.

The main goal of an RS is to collect and propose documents, products, information, or services that the user might want to find.

It can be defined as a means of assistance for the users when there is not enough time, experience, or knowledge to evaluate all the perspectives on a web page [32].

Due to the current "information overload", which can be explained as a situation where the received information is too much to process and understand, RS can play a significant role in decision making by providing personalized and exclusive suggestions. This is why in recent years many different systems are developed in increasingly diverse areas of activity. Fig. 1, describes the architecture of a recommendation system.

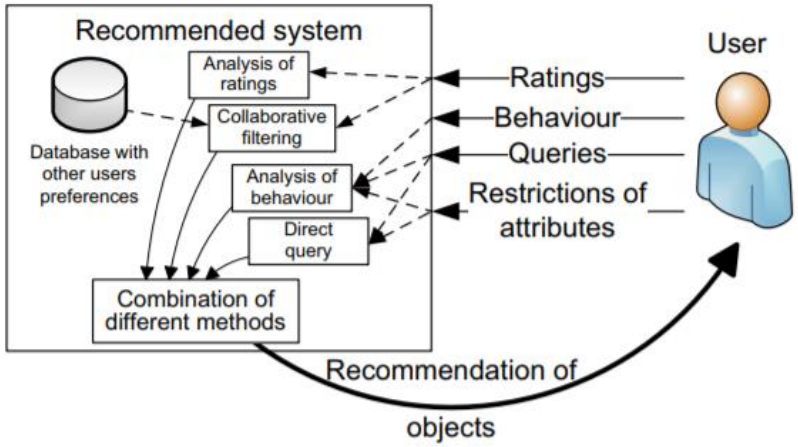

Figure 1: Architecture of a Recommendation System [12]

1.3 Fields of implementation

Recommender systems have gained great popularity over the past years on many online platforms. This section is devoted to some of the most well-known cases.

Amazon

Amazon is very famous for the personalization experience that offers to the customers. In the beginning, Amazon started as an online bookstore but gradually became one of the most well-known shopping platforms that include a huge variety of products, and nowadays even services. In the middle of the 90s, the main algorithmic approach for recommendations was item-based collaborative filtering. Furthermore, it was focused on the users, since the first step of the algorithm was to find similar patterns in other users and then discover what items those users have seen or picked,

but the current user has not found yet. Over the years, the Amazon algorithm for a recommendation has improved so much that currently as soon as a user visits the site, he finds recommendations based on his past researches and purchases on the homepage. In addition, the shopping cart proposes extra items that may be complements to the already selected items. At the end of the user's purchase, more proposals appear for later buys. This way, Amazon was able to build a personal store for every client who visits the online platform in the past 20 years. The prototype algorithm has found a great response over the World Wide Web and was implemented in many online sites. Nowadays, Amazon is the world's largest online retailer and a great part of its revenue annually derives from the item's proposals [38].

Facebook

Facebook was the first social networking site that met such a great success and was created in 2004. Originally, it was designed as a platform only for college students but soon gained great popularity all over the world, hosting over 2.6 billion active users. Facebook proposal framework gives recommendations on "people you may know". These suggestions depend on user profiling. In other words, the system evaluates user information such as work, education, mutual friends, and groups in which a person might participate and proposes new people with similar profiles [13].

Some researchers have tried to approach the algorithm over the past years with the Baatarjav approach being the most popular. He designed a group recommendation system using hierarchical clustering and decision trees in order to suggest to the users the most appropriate groups based on their profiles [4].

YouTube

YouTube is currently the largest platform for video creating and sharing with 300 hours of video uploading every minute. It was established in 2005 and it stores more than a billion videos. The existence of such a huge variety of content makes the user's choice very difficult and complex. This is why the platform has developed many features for suggestions and has also launched its recommendation system. The goal of the system is to provide the right video suggestions to the user so that he will be satisfied and entertained. For this purpose, YouTube implements a top-N recommender that reviews the user's latest activity and refreshes regularly. Furthermore, the system tries to promote the large variety of content that exists on the site, but, concerning the client's privacy and personal data protection. In order to provide suggestions, the YouTube RS collects metadata and statistics of the video, as well as user data such as likes and comments. The algorithm behind the prementioned procedure has known a great success over the years, making the platform the most popular one in this kind of online service [10], [42].

Netflix

Netflix is the most popular Streaming Services platform around the world. It includes thousands of movies, including even its own productions and series. Since the options are limitless Netflix as well developed its RS, which is based on a combination of different algorithms according to the user's needs. One of those algorithms is a Personalized Video Ranker (PVR), which is mainly used for suggesting the genre of a video given the user's profile. This algorithm, also, needs to be general enough in order to provide a general ranking in the whole catalog of movies, so, there are many limitations in personalization. Furthermore, a Top-N video ranker is devoted to finding the best proposals in the whole movie catalog for each user in correspondence to his needs regardless of the movie genre. This permits a higher level of personalization and user profiling. Additionally, there are other algorithms as "trending now", "continue watching" and "video similarity", which, as the ones mentioned, rely on machine learning techniques and statics to provide the best recommendations to the active user [15].

In 2006, Netflix released a dataset of 100 million anonymous ratings and offered the prize of 1 million dollars to the team that could beat Cinematch's accuracy, which was the current system used. This move triggered the programming community worldwide and many new recommendation systems were created, some of which are still used by the platform [6].

1.4 Benefits of RS

The great success of the RS derives from the fact that they are beneficial both for the users and the company that implements them.

From the website's point of view, an RS can contribute to revenue growth and an increase in total sales. An RS can bring more users to the website via personalized emails and directed blasts. Furthermore, it can promote relevant or complementary content and propose items that the user will be more willing to buy. Additionally, the systems can promote new or unpopular items to targeted users, creating new sales opportunities. They enhance quick and automated profiling and personalization in the web pages. So, the products in a platform can change and are situated according to each client and his preferences. Nevertheless, an RS can provide accurate and up to date reports to the website about the traffic and the direction of the promotive actions [29].

It is, also, very important to mention that such a system can improve customer satisfaction and loyalty. It can increase sales by generating tailored needs for the visitors and propose similar products connected to the ones they already purchased. So, it supports customer loyalty as it shows that the business takes time to understand

the needs of its clients. To continue, with the implementation of those systems the users can get engaged easily to the website since they are in the position to dive into the product offering without performing search after search. As a consequence, it is very likely that a shopper will turn into a customer and that a current customer will remain loyal to the website as he is fulfilled by its services [29].

From the user's point of view though, an RS presents also many advantages. It can reduce the time and effort a user spends while shopping by proposing things that will be more interesting and useful to the active user. Moreover, the system can aid in the discovery of new items which is a service many users appreciate. Lastly, the RS releases the user from the position of having to manage the chaotic amount of information within a web page [29].

1.5 Open Challenges

Some of the main open topics regarding the implementation of a recommendation system and the work that has been done so far are presented below.

Obtain user's preferences and generation of profiles

In many cases, implicit user feedback is widely available and can be acquired with minimum effort. For example, it is very easy to track the users clicks on a web site and receive it as feedback for a specific item. Past actions can contribute to predicting future ones. So, more and more approaches focus on using consistent and accessible implicit information. In such occasions, the problem that arises is that in such an implementation we can only receive records that are either positive or missing. In the missing category, the items that exist can be both the ones the user ignored as well as the ones that would never be proposed to the user by his preferences [39].

Another interesting aspect of the data acquisition problem can be the sentiment analysis of the users' feedback. This approach is constantly evolving and many techniques are being developed with a view to evaluating the emotions and the attitude of the consumer [39].

User interface

A very important challenge according to RS is to examine the mechanisms through which the user's inputs are received regardless of the models that are used to provide the recommendations. This means that while interacting with a consumer apart from

the final decision of selecting or not an item, there are many more information that can be collected.

Even though nowadays we are able to collect more and more information about the user's preferences it is not yet clear if all these data bring more benefit than loss to the system [39].

Privacy

It is very essential to ensure the safety of the data used in the construction of an RS. Especially, after the institutionalization of the GDPR law, the protection of the user's information has become one of the main issues that need to be taken into consideration, since the user actions are a vital part of the recommendation process. In many cases, the feedback can reveal apart from the preferences of a user, details about his political or relational beliefs, sexual preferences, and in general sensitive information. Thus, huge privacy concerns are aroused and the need of finding a balance regarding the amount of information asked and collected is more crucial than ever [39].

1.6 Scope of the dissertation

The scope of this study is to create and evaluate a book recommendation system based on collaborative filtering. After explaining the different variations of an RS and stressing some of the most relevant and important work that has been done already, a real dataset is being examined. The purpose is to test whether and how the comments of the users can be more helpful in a recommendation than the actual ratings concerning the generation of better suggestions by the system.

1.7 Dissertation outline

In the first chapter, an outline of the history, as well as the definition of the recommender systems, is presented. Furthermore, some of the most popular cases of implementation in the past 30 years in different fields, highlight their significance in online retail, commerce, and services. Additionally, the main advantages that an RS can offer to the user and the website are analyzed. Finally, we highlight some of the main challenges regarding the recommendation systems, define the scope of this dissertation and, provide a short description for each of the following chapters.

In the second chapter, we explain the types of recommendation systems and the different inputs of data and ratings for an item. Furthermore, we describe the main similarity measures that are used for the implementation of the systems and provide

an overview of the most important evaluation metrics that are used to assess the performance of the systems.

In chapter three, a literature review for the state of the art in RS is presented. We examine the work that has been done so far and searches for similarities and differences with our approach. We investigate the problems that have come up and try to understand the reasons behind them.

In chapter four, we describe the details of the selected dataset and the preprocessing that has been done in order to use the data in our study.

In the fifth chapter, we describe the Python code that has been used in the study and provide a holistic overview of the purpose of each action and the libraries used. Also, we outline the evaluation of our system and compare the findings.

Finally, in chapter six we present the conclusions regarding the dissertation and propose some ideas for future work.

Chapter 2: Background

In this chapter, we present the main concepts of recommender systems and provide the core knowledge that is a prerequisite for this dissertation. In more detail, we introduce the main types of recommendation systems and the most common users' inputs. Furthermore, we examine the similarity measures with the most frequent implementation in an RS and we display the main criteria used in the evaluation of the performance of an RS.

2.1 Types of Recommendation Systems

Recommender systems can be personalized, non-personalized, item to item correlated, attributed based, and people to people correlated. They can be either of short duration or long one according to their usage. They can be automated in the case that the system needs no input or only minimal input from the user and manual if the user must put a lot more details. Personalized systems are automatic and have their basis on the user's preferences and personal tastes such as favorite music, color, and movie. Non-personalized ones are based only on the ratings of the users of the website. These systems are more straight forward and easy in implementation as they require less effort to produce results and are also considered automatic as the user input is unnecessary. These systems are also long-lived since they can be implemented in many users regardless of their characteristics. Attributed-based recommender systems' information is generated by the item's features. This system is considered manual as the user has to search for a specific kind of product to generate a recommendation and it can be considered as long-lived or not, regarding the time that it remembers the user's preferences. Item to item correlation system recommends items based on other items for which the user has already expressed a preference. These are widely used in the web sites when new products are released and they generate proposal based on what the user has already placed in his cart. They are manual as they require input from the user (already chosen items) and short-lived. People to people systems search for similarities between different users and propose items other customers had bought or rated in the past. They are considered manual as it is mandatory that previous ratings or buys have been made and their duration depends on the design [37].

Over the past years, many types of RS have been developed and applied in different fields of the Web. Furthermore, they are extensively used in search engine machines as well. Even though various models have been employed over the past two decades, most of them have their base in four main recommendation types [34]. The approaches vary on the domain they address, the knowledge that they use, and especially in the algorithmic methods.

2.1.1 Content based filtering

Content-based filtering mainly focuses on the item, and its descriptive characteristics are used to create recommendations. The RS proposes options based on the user's previous interactions with similar items and combines them with the content information of each one of them. The term content denotes the items' descriptions. By analyzing the user's actions, the system creates a profile based on past preferences and this is how the user is represented. So, the main goal is to match a profile with an item that the customer would like to see or buy [2].

This approach originates from information retrieval and information filtering research. The recommended item indicates textual information and it is described with keywords and according to weights. The techniques mainly used, are divided into two categories, heuristic-based methods, and Model-based ones. The techniques used for the first class are KNN, clustering, and TD-IDF for information retrieval, while for the second are Bayesian classifiers, Artificial neural networks, decision trees, and clustering with a view to recommending appropriate content, based on the characteristics of the item and the preferences of the user [41].

The main advantage of this system is that it can be used with a small dataset even if there are not enough ratings and still generate good recommendations. This is because the active user might have rated items with common features. As a result, the RS will be able to make use of these ratings and combine them with the item's content in order to make suggestions even if there are no past references to the specific item [3].

The main disadvantages of such a system are limited content analysis, overspecialization, and new user problem.

New user problem

Even if those systems are very effective at recommending new items, they are not able to handle the new users because they need past user references and ratings for the training procedure, which are not available. Therefore, it is very difficult to generate valuable and accurate recommendations for a new customer item [3].

Limited content analysis

In order to make appropriate recommendations the system depends heavily on the item's features. For this reason, the attributes must be in a form that is easily evaluated by the system, for example in a textual one. Information retrieval techniques work particularly well with this type of data but they really straggle with multimedia records such as audio and video. Another important issue with limited content analysis is that since the items are characterized by a set of features it is

possible that two different items have the same set. When this occurs, the recommendations can be affected since it is impossible to differentiate between the items [2].

Overspecialization

Content-based filtering makes proposals based on the items that the user has liked, purchased, or rated before. Therefore, the recommendations consist of similar items and the introduction of new ones is almost impossible. An easy way to overpass this problem is by proposing some random items to the active user. Furthermore, another problem with overspecialization is that the suggestion of comparable items is not desired every time. For example, it is likely that a customer would not be interested in books that refer to the same event (e.g., World War 2) since he has already purchased or reviewed one or more of this genres or categories [2].

2.1.2. Collaborative Filtering

Collaborative filtering uses mainly the past behavior of a user, like his ratings and transactions without the need of creating a new profile for the user. It is dedicated to relating participants' dependencies among products to identify a potential association between the user and the product [21].

It can be divided into heuristic-based and model-based methods depending on the methods utilized for rating estimation. The heuristic method uses rating data, duration time, purchasing binary data, and clicks as inputs and acquire results based on the total customer database. The system compares the current user with every participant in the dataset in terms of similarity using different metrics as correlation coefficient and Jaccard similarity. Afterward, the value of the product, which is about to be proposed, is predicted using a KNN procedure. Eventually, the active customer will be recommended the highest rated products of his neighbors. This system usually is based on web mining techniques, KNN, decision trees, graph theory, and SVMs.

Model-based includes the training of data and checking about the validity of the model by implementing test data. Then, the system predicts the rating value of the products the active costumer has not yet evaluated. In terms of comparison with the heuristic method, the model-based methodology only introduces a part of the active user's data and gives prediction values and presents. Some of the main procedures the system involves are clustering, association rule mining, Artificial Neural Networks, Linear regression, maximum entropy, latent semantic analysis, and Markov processes.

Collaborative filtering can overpass some of the drawbacks mentioned in the content-based approach since different users' ratings are being collected and implemented and

as a result, it is feasible to propose dissimilar items from the ones reviewed in the past [2].

But there are important disadvantages as well, such as cold starting problem, sparsity problem, gray sheep problem, and scalability problem with the two first being the more important [41].

Cold-start problem

This problem is met in the new entries either for a product or a user as the system does not have enough data. So, when a new user appears there are no references for his behavior and the RS cannot come up with a relevant proposal. The same applies in the case a new item appears since it still does not have any ratings.

Sparsity problem

It is one of the most important problems since in the majority of cases the items do not have enough ratings in order to be recommended. This is because a normal user can evaluate only a few of the items in comparison with the total number that exists on the website and the needs for recommendations are much more demanding. This problem can cause complexity in finding similar users [2].

2.1.3 Knowledge based filtering

This approach is particularly useful when there is a lack of ratings. It can apply in fields such as automotive, real estate, and luxurious products and services. It is evident that items included in the prementioned categories are not purchased very often and therefore they are not reviewed as well. As a result, the recommendations in these cases are very poor. This problem can be resolved with the implementation of a knowledge-based approach in which the ratings are not used in the recommendation process. On the contrary, the proposals are presented based on the requirements or constraints of the user in comparison with the item characteristics. During this process, the systems also use knowledge bases which include information about similarity guidelines with a view to providing better results. In reality, the knowledge bases are so essential that the whole approach is named after them.

Knowledge-based systems can be categorized into constraint-based and case-based recommender systems. In constraint-based, the recommendations are heavily influenced by the rules the users establish on the characteristics of the item. These restrictions consist of the domain knowledge for the system and they facilitate the generation of recommendations. In the case-based approach, the user identifies specific marks or goals and the system tries to retrieve items with similar characteristics by implementing similarity metrics [3].

The main advantages of these systems are that they are very sensitive to the changes in user's choices and also, they overcome the "ramp-up" problem. The "ramp-up" problem refers to the difficulty of a system in making suggestions before it has collected a specific number of user preferences and items' ratings and its concept is very close to the cold-start problem. But, on the other hand, some drawbacks also exist in this approach. The most important is, that for the operation of the system domain expertise or perhaps knowledge in engineering is needed and the cost and effort for construction can be exceptionally high. In addition, the recommendation abilities of the system are not dynamic, so it is impossible for the model to adapt to short term changes [8].

In table 1, the differences of the three categories examined so far are depicted.

Approach	Input	Output
Content Based	Ratings + item content	Recommendations based on the attributes of the item combine with the past users' actions
Collaborative	Ratings of the user + neighbors' ratings	Recommendations based on the actions of the user and the matching users
Knowledge based	User rules + item content + domain knowledge	Recommendations based on specification regarding the features of the item

Table 1: Differences of Recommendation Systems

2.1.4 Hybrid model

The three mentioned approaches take advantage of different inputs and can be implemented in different cases with great success. But, since all of them present specific weaknesses, hybridization can be implemented as a method to get the maximum benefits. This means that mainly the first two approaches, content-based and collaborative filtering, are combined in a way to generate more valuable and accurate recommendations. The hybrid approach is highly influenced by ensemble learning in the field of machine learning, where, different algorithmic models are combined with a view to creating a hyper-model [3].

Over the past year, several ways of creating a hybrid model have been introduced. The main approaches are the following:

1) Employ both content-based and collaborative filtering independently and then mix the results.

2) Introduce some features from one approach to the other

3) Compose a hyper-model that includes characteristics of both content-based and collaborative filtering.

The hybrid approach can combine different systems and various inputs so that it can offer solutions to the major limitations of the individual approaches. For example, the cold start problem which is very difficult to overcome in collaborative filtering can be resolved with the initialization of a content-based system that does not depend on previous ratings [2].

2.2 Types of input data

In order to create a functional system, a great amount of data should be collected. Usually, the systems use information that includes demographics, item features, and customer preferences. We can divide the input data into five main categories. The first one would be rating data and it basically includes rating scores, likes and dislikes as well as comments about an item or a purchase. The simplest way to collect this type of information is to voluntarily get the users' opinion for purchases they have already made and build a valid and sufficient dataset. The second type is transaction data that includes metadata about the purchases such as the date, quantity, possible discount, and price. The Behavior Pattern data is another important type of inputs. This category involves information such as the browsing duration, how many times a user has clicked, the links that a web page might include, and all the commands that a user can make in a site for example save, copy, bookmark, scroll, search, refresh and even download some content. The fourth type is production data. In the case of music or movie recommendations, those data refer to the singer or actor, release date, ticket price, production firm, and so on. In the case of websites or documents, it is related to the content description using keywords, links to others, the topic, and the viewed times. Finally, another important category of input data is demographics. Demographics include general and overall information about the users such as preferences of specific age or gender group, financial data, job, and hobbies. This information can be extracted from the user's profile and can be used to create categories for the users [37].

According to [35], there are various types of ratings as well.

- Numerical: use a number of stars usually from 1 to 5 to evaluate an item
- Ordinal: use a set of words to describe the satisfaction of the user according to each item such as "excellent, very good, good, bad, very bad"
- Binary: the user is offered only two choices, "good" and "bad" in order to evaluate an item
- Unary: this type of rating designates that the is a positive reaction from the user which can be a purchase, a good evaluation, or even just the search and

observation of the item. If this type of rating is missing it indicates that there is no information available about the item or the user.

- Tags: the user has the capability of commenting on the items the system is proposing

Input data can also be classified as explicit or implicit. We can characterize as explicit the user input regarding the rating of an item. For example, likes and dislikes, as well as scale rating systems, are included. Rating scores may vary depending on the product and the website but the most common systems are the ones of 5 or 10 stars. On the other hand, implicit input refers to the user's actions. Some of the most common examples are the time spent on a page or an item, adding an item to a wish list, and the purchase history. Even though it is hard to obtain implicit data, they provide more insight details about the user's behavior.

2.3 Similarity measures

Similarity measures are of high importance especially in collaborative filtering where the comparison between users or items needs to be examined. The most well-known metrics are cosine, Jaccard and Pearson similarity [9].

Cosine similarity (equation 1) is a metric used to define how similar two non-negative vectors are, by calculating the cosine angle between them. The main disadvantage of this method is that it classifies null or missing values as negative ones.

$$\cos ine(a,u) = \frac{\sum_{\{i \in S_a \cap S_u\}} r_{ai} \times r_{ui}}{\sqrt{\sum_{\{i \in S_a \cap S_u\}} r_{ai}^{\,2} \sum_{\{i \in S_a \cap S_u\}} r_{ui}^{\,2}}}$$

Equation 1: Cosine similarity

Pearson similarity (equation 2) is a similarity metric used to compute the linear correlation between two vectors and receives values from -1 to 1 with 0 indicating no correlation at all [9].

$$pearson(a,u) = \frac{\sum_{\{i \in S_a \cap S_u\}} (r_{ai} - \overline{r_a}) \times (r_{ui} - \overline{r_u})}{\sqrt{\sum_{\{i \in S_a \cap S_u\}} (r_{ai} - \overline{r_a})^2 \sum_{\{i \in S_a \cap S_u\}} (r_{ui} - \overline{r_u})^2}}$$

Equation 2: Pearson similarity

Jaccard similarity / coefficient as denoted in equation 3 takes into consideration the number of mutual choices between users. It measures the volume of items that have been reviewed and does not refer to the absolute ratings the users have provided to the products. Two users will be considered as similar when they have both evaluated a lot if common items. In reality this means that if two users have evaluated the same item, one positively and the other negatively, according to Jaccard similarity the users are considered as alike. So, the major disadvantage of the method is that it produces limited results so the users cannot be easily discriminated [9].

$$jaccard(a,u) = \frac{|\{S_a \cap S_u\}|}{|\{S_a \cup S_u\}|}$$

Equation 3: Jaccard similarity

When Jaccard similarity is used in combination with another similarity measure then the drawbacks can be eliminated and the results are more accurate and beneficial.

2.4 Evaluation of recommendation systems

A very significant issue regarding an RS is its evaluation. How well does a system behave and how can we measure its results? An RS can be evaluated using two different approaches, online and offline. In the online method, the valuation is heavily related to the active user's reactions regarding the current recommendation. This way the straight influence on the end-user can be measured. However, it is usually very difficult to collect this type of validation since they demand the constant involvement of the active user, and as a result, this technique cannot provide valuable results for the research. Offline evaluation methods can be implemented in an RS and can provide more detailed insights about its performance. This method gives the ability to test different datasets and combine them with a view to obtaining a better generalization performance. When we refer to offline evaluation, accuracy is not an adequate metric for the evaluation so it is essential to determine extra criteria that will aid in the assessment of an RS [3], [9].

a. Accuracy metrics

Despite the significance of complementary metrics, accuracy still remains the sole more important measurement in a RS evaluation. Offline evaluation can be implemented by assessing the accuracy of the rating values that are predicted or by

assessing the accuracy of the ordering of the proposed items [3], [9]. The most frequent accuracy metrics are described below.

Predictive accuracy metrics

The predictive accuracy metrics evaluate how similar and precise the predicted ratings the system produces about an item are, in comparison to the actual ratings. The main tools are denoted below.

Precision – Recall – F-score

To define these metrics, we assume that the items are divided into two categories. The first consists of the items that are retrieved and the user has full awareness of every item, and the second consists of the items that were not recommended.

As precision, we define the ratio of the relevant items that were recommended over the total items that were recommended as depicted in equation 4. It resembles the probability that a recommended item is actually relevant to the user's preferences [17].

$$P = \frac{N_{rs}}{N_s}.$$

Equation 4: Precision

As recall, we define the ratio of the relevant items that were recommended over the total relevant items that exist for the user's profile as descripted in equation 5. Recall, resembles the chance that an appropriate item will be retrieved [17].

$$R = \frac{N_{rs}}{N_r}.$$

Equation 5: Recall

F-score, is defined as the harmonic mean between precision and recall and it is displayed by equation 6. It combines both of the prementioned metrics and results in a more representative quantification [17].

$$F_1 = \frac{2PR}{P+R}.$$

Equation 6: F1-Score

ROC curves

Another way to express the relationship between precision and recall is the Receiver Operating Characteristics curve (ROC curve) which can be characterized as an improvement of the f-score since as a metric it is not dependent on the size of the recommendation set. The ROC curve introduces the True positive rate (TPR) which is shown in equation 7 and is equal to the recall, and the False positive rate (FPR) which is presented in equation 8. The false-positive rate denotes the ratio of irrelevant items that were incorrectly proposed to the user as relevant and can be characterized as a "wrong" recall [9].

$$TPR(t) = Recall(t) = 100 \cdot \frac{|\mathcal{S}(t) \cap \mathcal{G}|}{|\mathcal{G}|}$$

Equation 7: True positive rate

$$FPR(t) = 100 \cdot \frac{|\mathcal{S}(t) - \mathcal{G}|}{|\mathcal{U} - \mathcal{G}|}$$

Equation 8: False positive rate

The ROC curve is created by plotting the "wrong" recall – FPR on the x-axis and the correct recall – TPR on the y-axis. Both, TPR and FPR are calculated in percentage so the range of the axis is between 0 and 100. The area under the ROC curve indicates the effectiveness of the RS used and it is called AUC (or Area Under the Curve) area. Like, recall and precision the ROC curve makes a binary hypothesis of relevant and non-relevant items, so each time an item can be correctly proposed or not. If every relevant item on the recommendation list is proposed before the irrelevant ones then the perfect ROC curve is constructed. The major disadvantage of this method is that a big volume of data is needed in every step, in order to produce valuable results and distinguish accurately between relevant items and non-relevant ones [9].

Mean Absolute Error – Rooted Mean square Error -Mean Square Error

The Mean Absolute Error (MAE) as described in equation 9 calculates the mean absolute divergence of a predicted rating and the actual one the user has eventually provided. The main advantages of this method are that in the first place it is a method particularly easy in understanding and implementation and secondly it can produce statistical results of high importance. In addition, this method does not incorrectly punish the large errors as there is not a squared term [9].

$$\mathrm{MAE} = \frac{\sum_{(u,j)\in E} |e_{uj}|}{|E|}$$

Equation 9: Mean Absolute Error

Another variation of the MAE is the Mean Absolute error which is depicted in equation 10. The advantage of this method is that it can be very efficient with small values in producing accurate results. The square root of it is known as root mean squared error (RMSE) and is described in equation 11. The main gain of using it instead of MSE is that the results are expressed in the same unit as the user ratings and not on the square of that quantity as in MSE. At the same time, the main drawback is that due to the fact that the squared term is in the summation, RMSE tends to incorrectly penalize the large errors [3].

$$MSE = \frac{\sum_{(u,j)\in E} e_{uj}^2}{|E|}$$

Equation 10: Mean Squared Error

$$RMSE = \sqrt{\frac{\sum_{(u,j)\in E} e_{uj}^2}{|E|}}$$

Equation 11: Root Mean Squared Error

Rank accuracy metrics

This metric evaluates the ability of an RS at suggesting items with the same order as the user would have selected them. It is an appropriate way of measuring the RS

algorithms that are used with a view to generating ordered recommendation lists for the users when the recommendations are not of a binary type. The main disadvantage of this metric is presented in cases where the user does not care about the items' rating.

Ranking metrics are not adequate to evaluate the accuracy of an RS at predicting the order of a single item, so in order to obtain an overall view of the system's performance, it is important to be combined with the predictive accuracy metrics that were described in the previous section [17].

Prediction rating correlation

As it is already clarified, when evaluating an RS, it is very essential to examine the relationship between the produced rating list and the actual sequence of items that the user selects which is commonly known as "ground truth". Since the ordering sequence is discrete, it is very normal for ties to exist in the ground truth, so there is a need not to incorrectly "punish" the system in the case it rates improperly two items that are in the same position. The most popular method is to use a rank correlation coefficient as it is capable of comparing even non-binary rankings, it can provide a holistic evaluation for the RS and it can be easily understood by the scientific and research community [17]. The two main approaches are:

Spearman rank correlation coefficient

It measures how dissimilar two impartial ratings are, regardless of the true features' values and it is depicted in equation 12 [17].

$$\rho = \frac{\sum(u - \bar{u})(v - \bar{v})}{n * \ stdev(u)\ stdev(v)}.$$

Equation 12: Spearman correlation coefficient

In order to calculate the spearman coefficient, we need to place both the predicted recommendation items and the ground truth in the correct sequence. Then, the Spearman correlation is similar to the Pearson (described already), for these orders of items. The values that can take are from -1 to 1 and it is more efficient with small values. The Spearman coefficient can be computed for every user and then, get an average value for all the users, or it can be computed overall ratings at once. An important problem with this method is that the ties that may exist between the items can produce noise to the assessment of the RS [3].

Kendall rank correlation coefficient

It measures how strong the relationship of two ordered recommended lists is and to what direction it moves- positive, negative, neutral. Even, though the Kendall coefficient is a very simple and easy to implement the method, it is not widely used for the evaluation of an RS.

b) Non-accuracy metrics

It is essential for an RS not only to be accurate but also to provide an adequate and pleasing experience to the end-user. So, apart from being accurate, an RS must also be efficient. For example, a system might be able to predict only the very obvious items, which are the ones that the end-user does not actually a suggestion for. Or, the same might apply for the least obvious or irrelevant items that there is no possibility that the user will buy. Therefore, other evaluation metrics should be determined that validate the system's effectiveness beyond accuracy. The most important of them are presented below [3], [17].

Coverage

It measures for what part of the dataset the system can generate recommendations and it denotes the domain of items that can be suggested. A system with small coverage is usually less satisfactory for the users since there are a lot of limitations in the proposals they can make. The easiest way to define the coverage of a system is by measuring how many of the total items, recommendations can be generated, and take the percentage. To do so, we can just select arbitrary pairs of user-item, request a forecast for every set, and measure the rate for which an expectation was given. An important note is that coverage should be measured in parallel with accuracy so as the users will not intentionally change their predictions and affect the final outcome.

Learning rate

RS include also learning algorithms, and it is evident that the quality and quantity of suggestions can vary depending on the amount of data available. More information should result in more accurate recommendations. The quantity of data needed can differ from system to system as there are cases where a model can produce adequate results with only a few entries and cases where the model requires a greater number of entries. The learning rate of a system can be distinguished into three classes. The first is the overall learning rate, which depends on the total number of ratings in the systems. The second is the per item learning rate indicates the quality of

recommendations for an item regarding the whole number of reviews existing for it. The last one is the per-user learning rate that denotes the quality of suggestions for a user regarding the number of reviews the user has provided.

Serendipity

This “metric” refers to the innovation in an RS system. It happens many times that an RS might propose a very popular item or an item that has been already purchased by the user. In both cases, the proposal is meaningless as it adds no extra value, and it can be characterized as an “obvious” recommendation. Even though the users might sometimes appreciate recommendations of items they are already familiar with, the introduction of “serendipity” is inevitable. That way, potentially interesting new recommendations can be generated that otherwise might never be examined by the end-user. There is no simple way to measure “serendipity” since it rates how much a recommended item is attractive or unexpected for a user.

Confidence

Most of the time, the users might wonder about the certainty of the system’s recommendations which can be tackled as a matter of confidence. In other words, they may ask about “how guaranteed is it that the generated suggestions are correct?”. The goal of the RS is to create a positive experience for the user and assist him in buying what he needs, so it is essential to incorporate confidence in this procedure. Although, confidence is really hard to measure. since it is a complex phenomenon and it cannot be described by one-dimensional measures, it is clear that if a system does not include confidence to some extent, the recommended results will be less useful and accurate.

Chapter 3: Literature Review

3.1 Background and examples of online systems and engines for book recommendations

During the past years, interesting work and research have been done in the field of recommendation systems. Even though they are mainly used in the field of e-commerce and on-line services, they are many cases where the systems were introduced in the area of book recommendations. Some of the most significant online book retailers are presented below.

Whichbook.net is a platform that uses collaborative filtering to generate suggestions about what book the user should read next, and it is based on the idea that similar users will like the same books. So, based on the author or the title the system proposes items that match each customer's reading list. Common items with the ones existing in the reading list are proposed to the user [11].

Librarything.com is an online service that supports book cataloging for the user. It implements a hybrid RS based on the site's library categories combined with the user's preferences according to their reading lists. Furthermore, it proposes links for relevant bookstores [31].

Booklamp.com is yet another book recommendation system that was purchased from Apple in 2014. The system predicts user preference by analyzing the book's viewpoint, tone, and writing in order to propose books in a similar style. The main advantage is that it can remove noisy features from the datasets such as marketing and advertising, and so it is one of the most effective systems so far [31].

Bookexplorer.com is an engine that supports the functionality of searching for books. Books are stored in lists and as soon as a user selects a book the whole category is also presented. It implements collaborative filtering within each list [31].

3.2 Related Work

3.2.1 Building a Book Recommender system using time-based content filtering

The authors in [31] suggest an RS that combines both user preference and item features with time. The system takes into consideration the period that an item stays in the favorite list over a specific time window. Each item in the list has a counter that is

updated every time the user checks that item. The highest the result of the counter the higher the possibilities of it being recommended. The system takes into consideration the counters of all users, and not only of the particular one, in order to achieve variety in the recommendations. Over the defined time frame the books' counters can change and become more or less favorite, ensuring that the list is updated at any given point. Content-based filtering is implemented to process user preferences and results in recommendations of a specific category of books. In addition to that, the user is given the possibility to search for a category or a particular book, and, when this procedure finishes, the preferences are registered. It is essential to regularly update the preferences, as the system is dynamic and the lists can change over time.

The paper concludes that the addition of the time dimension in the system results in better book recommendations. In particular, time is a criterion that was not used before in order to explain the accuracy of the system which from a point and on was stuck in generating the same proposals. The proposed model, not only is able to provide meaningful recommendations with fewer ratings than usual but also, the extra temporal dimension can lead to higher user satisfaction and more useful and related suggestions. The system was set under examination by a group of participants that were asked to evaluate how the addition of the time impacts the recommendations and the final outcome of the survey was used to evaluate the proposed RS in the research. Out of the total, 74% concluded that they observed meaningful changes to their recommendations, that could change on a daily, weekly, or monthly basis. Furthermore, over 70% of the people, found these changes valuable and gave the system an overall score of 8 out of 10 or above.

3.2.2 A graph-based recommender system for digital library

In [18], the authors introduce a two-layer graph recommender system that applies to content-based, collaborative, and hybrid filtering. The system incorporates user-to-user, book-to-book, and book-to-user correlation, and the tested dataset was obtained by a foremost Chinese bookstore, containing information about books, costumes, and transactions. The books and customers are represented as feature vectors. The book's vector contains information regarding the book and its content such as the title, genre, abstract, etc. and the customer vector contains the client's demographics. Then, the vectors are compared using similarity measures. The second step, introduces the modeling of books, customers, and transactions in a two-layer graph, one layer for the books and one layer for the customers, using the weights obtained by the similarities in the first stage. In the book layer, every node denotes a book and every link denotes the similarity between them. In the customer layer, each node denotes a client and each link denotes the similarity in terms of demographics. Finally, a link between the two layers represents a purchase, so the inter-links can be referred to as the transaction's history.

The system implemented acts as a graph search engine, which focuses on discovering convincing relationships between the items by using different searching methods. The produced model is simple and understandable and also, can be applied to all of the main approaches concerning an RS. If only the book-to-book similarity is taken into consideration then the model becomes a content-based one, whereas, if only the customer-to-customer correlation is used, the approach is clearly collaborative. If all the correlations are combined then the model results in a hybrid one.

In the paper, all the prementioned approaches were tested in terms of precision and recall, using low and high degree association search methods. The evaluation part was split into two different approaches. Initially, a hold-out test, similar to the concept of cross-validation, was held by selecting randomly 100 clients and their book purchases in date order. The purchases were divided into the older ones that consisted of the training dataset and the more recent consisted of the test dataset. The first set was used to train the algorithm and produce recommendations and the second set was used to compare the findings with the real purchases. In addition, ANOVA tests were run to examine the impact of the degree of associations of the suggestions on precision and recall. The hybrid system outperformed the content-based and collaborative filtering with a precision of 2,76% and recall 14,52% on the low-degree association and 3,32% and 13,51% respectively on the high-level association. Furthermore, a statistically significant difference with regards to precision and recall was noticed between the three approaches, whereas there was no significant difference between the level of association.

After the hold-out testing, another evaluation called “subject evaluation” was implemented. Two well-educated Taiwan citizens were selected to suggest books based on their knowledge, experience, and information available on online book stores. Three clients were randomly selected and six recommendation lists were generated based on their history purchase, which was available to the two “subjects” as reference. Using the lists, the “subjects” could get a basic insight on the clients’ interests, and, afterward, they were asked to create a new list of recommendations using books that were included in the initial six lists. The final list was used as a benchmark to compare the different system designs. The outcome was that the content-based filtering outperformed the other two methods both in precision and recall with a score of 16% and 36,3%, respectively, for the low degree association and with a score of 18,3% and 38,1%, respectively, for the high degree association. The difference between low and high degree association was not found significant and the fact that the content-based design was the best one can be explained by the idea that the “subjects” have selected most of their recommendations based on the theme and content of the books available.

The findings conclude that the hybrid model was the one with the best performance based on the results of the hold-out test and the content-based model was the best based on the results of the subject test, but in both cases, there was no important improvement regarding the level of associations.

3.2.3 Book Recommendation System through content based and collaborative filtering method

The authors in [26] introduce a hybrid approach that combines content-based and collaborative filtering with keyword-based filtering and data mining. Regarding the keyword-based filtering, the user is asked to passwords matching his preferences that are stored in a database and will be used for future recommendations. In terms of data mining, the ECLAT (Equivalence class Clustering and bottom-up Lattice Traversal) algorithm is introduced to spot the repeated items. This algorithm can parse the items from the beginning and only one time to identify the most relevant and frequent ones, which makes it very quick and efficient. The proposed procedure is completed in 6 steps. Firstly, the whole book dataset is scanned and the irrelevant items are removed. In the next stage, a data preprocessing procedure results in the extraction of only the useful information for the next steps. In the third stage, the transactions are filtered and the books are classified by category and sub-category. In the fourth step, content-based filtering is implemented based on the preferences of the user and in the fifth one, collaborative filtering is used to evaluate the book's content by combining ratings and reviews. Finally, the book with the most ratings is recommended to the user. The authors highlight that the main challenges for the RS were regarding the fact that there is no way of evaluating the validity of the users' ratings and also, the fact that the system can be used only by people that have the knowledge of using a computer and the Internet. The study concludes that with the implementation of this system, the suggestions generated reached a high level of fulfillment regarding the user's requirements and also, benefited efficiency and productivity.

3.2.4 Hybrid attribute and personality-based recommender system for book recommendation

In [16], the authors introduce a hybrid approach using an attribute-based system, enriched by personality features. The first step is to create a user profile by forming a preference matrix. Attributes included in the matrix vary according to the user's prerequisites and objectives and include the title, author, price, genre, etc. The second stage is to generate a neighborhood by finding the similarities between the active user and the rest of the users. Except for collaborative filtering, the personality factor is also implemented as the model takes into consideration the relationship between the interests of the users and their rating behavior. In the third step, the MSV-MSL (Most Similar Visited Material to the Most Similar Learner) method is introduced to calculate the score of the book. Since all the mentioned steps are completed the

recommendations are generated based on the predicted ratings. The datasets used in the study are the Bookcrossing and the Amazon review, and the metrics used to evaluate the performance of the system were precision, recall, and f-score. During the evaluation of the RS different approaches of the MSV-MSL regarding the user similarity have been experimented, such as including patterns only with ratings or only with personality or with a combination of both. For the Bookcrossing dataset, the approach of only utilizing the personality achieved the higher f-score (0,035) in comparison to the other two. The paper reaches the conclusion that at least for this dataset the addition of personality factor can increase the F-score and hence, provide better results. On the other hand, for the Amazon dataset, the inclusion of the personality leads to a decrease in the F-score by almost 75%. The final results proved that the system was more accurate for the Bookcrossing dataset as it involves real features in comparison with the Amazon review dataset. Furthermore, the study reaches the conclusion that the performance of the RS and the quality of the generated suggestion depend heavily on the datasets used and the included attributes.

3.2.5 Web-based personalized hybrid book recommendation system

In [20], the authors introduce a book web-based RS. A computer server is used to support the procedure which is connected to the database where the information for the users and the demographics are stored. Furthermore, the database involves the books dataset and the relevant ratings combined with the factor of time as the older ratings are eliminated. Furthermore, the database includes another dataset that includes the information derived from the web. According to the authors, different software can be used to achieve the desired outcome such as Apache Hadoop or a relational database such as MySQL. After the required data are stored in the repository two main approaches of filtering are applied to generate recommendations, content-based, and collaborative. In the next step, demographic filtering is also executed based on the personal features of the user such as age and sex, with a view to generating more personalized proposals. Furthermore, web scraping is used to extract useful and meaningful information for the recommendation process. Even though, issues such as confidence in the system and data security arise, the authors conclude that the addition of the web factor can surpass the limitations originating from the filtering approaches and enhance the power and efficiency of the RS.

3.2.6 An Implicit Feedback model for Goodreads Recommendations

.

In [22], the author suggests a RS that is based on implicit data. To implement so, Spark library and ALS algorithm are used for the building and evaluation of the model. The dataset used for the study is the Goodreads book dataset which consists of 876K of users and 223M of interactions. Due to limitations of speed and memory, eventually only the 25% of the original dataset was used and all the users and books with less than 10 interactions were excluded. Afterwards, the dataset was split in train set, validation set and test set with percentages of 60%, 20% and 20%, accordingly. The alternative least squares algorithm (ALS) was implemented to perform matrix factorization and calculate the user-item matrices. It can be described as follows:

$$(R^*, V^*) = argmin_{(R,V)} \|R - U^T V\|^2 + \lambda(\|U\|^2 + \|V\|^2)$$

Equation 13: Alternative least square algorithm

where R represents the ratings matrix and U and V the user-books matrices. The aim is to minimize the least square errors of the ratings and regularize. The algorithm is an iterative one which implies that it will repeat until convergence. For each user, a data frame that consisted of 500 predicted items and the ranked order of items the users have interacted with, was created and was used to implement a ranking metrics' evaluation. In particular, the metric used to identify the best fit that should be used in the next step for the test set were Mean Average Precision (MAP), precision at k and Normalized Discounted Cumulative Gain (NDCG). Before the evaluation a tuning on the hyperparameters, rank, regParam, and alpha, was performed to optimize the model's performance. After a grid search, the best model based on NDGG and precision at k was found with rank = 30, redParam = 0.05 and alpha = 15. The evaluation results on the validation set are depicted in figure 2:

MAP	Precision at 500	NDCG at 500
0.01262	0.003846	0.075214

Figure 2: Evaluation results

Finally, the scores do not differ on the test set which implies that the model did not overfit. The author concludes that further work can be accomplished with more computational and time resources that can support multiple combinations of hypermeters and a bigger portion of the original dataset.

3.2.7 Recommending user generated item lists

In the paper [23] the authors stress the fact that the existing RS mainly proposes individual items and attempt to study the generated list items, a feature that starts to find implementation over different recommendation engines. The different properties of these lists are examined, and an algorithm based on a Bayesian ranking model, called Lire, is proposed for the generation of personalized item lists. Different baseline algorithms are also introduced in order to test the performance of the newly generated model. In particular, the approaches that are initially followed are a popularity-based approach, collaborative filtering with implicit data, and a Markov-based approach for recommendations.

The popularity-based approach is grounded on the idea that the item lists are sorted by the number of votes received and every user is recommended the top lists that have received the most votes. This algorithm is called GLB (GLoBal) as it is not taking into consideration personalized statistics but global ones. Two more variations of this approach are exploited, the PPOP (Personalized Popularity) and the PIF (Personalized Item Frequency). The first one can be described by the assumption that the user is only interested in lists that include items that he is familiar with. So, instead of simply recommending the most popular lists across all users, the algorithm recommends the most highly rated lists that include at least one item the user has interacted with. The second algorithm is based on the same assumption as PIF, but it ranks the lists by the number of items the user has interacted with and recommends the top ones by supporting the idea that the more familiar items existing in a list, the more interesting the list is for the user.

Regarding the collaborative filtering approach, since the feedback is implicit which means that the feedback is in terms of votes, the authors adopt the BPR (Bayesian Personalized Ranking) method. Every user and every item are associated with a latent factor, and then the similarity to other factors can be predicted by their dot product.

For the Markov-based approach, the adapted algorithm is LME (Latent Markov Embedding) where every item is associated with a latent factor. In this case, there is a difference regarding the fact that is algorithm is mainly used for generating playlists, where the items included are supposed to be in sequential order. To achieve this, the authors calculate the take into consideration the probability of each list being created as a series of Markov transitions.

Finally, the LIRE model is proposed which is based on the idea that each user, item, and list are associated with a joint latent factor of k dimensionality. The two main points under examination is the overall quality of the list and the user's interest in the items included in it. The model is further categorized into two variations. The first one is called UNIFORM and the second one is called DCG (Discount Cumulative Gain). The main difference is that the second model tries to also capture the influence that the order of items appearing in the list might have on the user.

The final step is to evaluate the algorithms and compare their quality. The book list dataset was obtained by Goodreads and a sample of 10% of all the user-generated lists was used for the evaluation. The metric utilized to measure the performance of the algorithms was the AUC score and the results are presented in the plot. The LIRE models outperformed all the baseline ones and the score of the UNIFORM was higher than the DCG. As a final comment, the authors stress out the fact that the UNIFORM algorithm is much slower than the DCG during training as it has to "visit" all the items included in an item list.

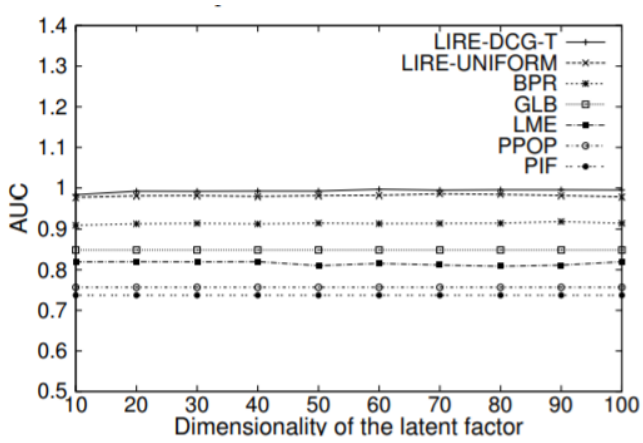

Figure 3: Comparative results for the proposed models

3.2.7 Hierarchical Gating Networks for Sequential Recommendation

The authors in [25] introduce a hierarchical gating network (HGN) as an attempt to overcome the two main problems the sequential recommender systems face, which are the difficulty of modeling the interests of the long-term users and the difficulty of identifying the interests of the short-term users. The proposed model is using the Bayesian Personalized ranking model (BPR) with a view to capturing the preferences of both types of users. The users' interests are represented by a user-item series in chronological sequence and the tasks focus on recommending to every user a list of items from a set S of items and evaluate if the items will appear in the generated recommendation list. The training data consists of ordered implicit feedback and the model is constructed by a feature gating module, an instance gating module, and an item to item product module. The architecture of HGN can be presented by three steps, the embedding layer, the gating layer, and the prediction layer. The input in the embedding layer is an order of items, where every item is denoted by a unique index.

When the data enters the layer, it gets transformed into a real-valued vector of a lower dimension. The gating module afterward acts like a "filter" and selects which data can pass to the next level where the latent features and items are selected. Furthermore, before the prediction layer, an item to item product module is implemented to capture the relations between the items and to identify how similar or dissimilar they are. Finally, the prediction layer calculates a score that is influenced by the user's long-term interests, the user's short-term interests, and the item-item product.

To examine the performance of the suggested model, five datasets from different fields were used, the MovieLens-20M, Amazon Books, Amazon-CDs, Goodreads-Children, and Goodreads-Comics. Only the users with more than 10 interactions participated in the final datasets, as well as the items with more than five ratings, in order to eliminate the noise. For every user, 70% of the interactions were used for training, 20% for testing, and the last 10% for tuning of the hyperparameters. The evaluation metrics utilized in the study are Recall@k and NDCG@k. The first one denotes the percentage of the user's rated items that appear in the top k recommended ones and the second metric is the normalized discounted cumulative gain at k which also takes into consideration the position of the recommended items. To compare the performance of HGN classical methods of implicit feedback and state-of-the-art methods were tested. In particular, the methods that participated are BPRMF (Bayesian Personalized Ranking based matrix), GRU4Rec (gated recurrent unit for recommendations), GRU4Rec+ (improved method of GRU4Rec), NextItNet (next item recommendation net), Caser (convolutional sequence embedding model), and SASec (self-attention based sequential model). The proposed method (HGN) outperformed all of the other methods in both Recall@k and NDCG@k and the final results are presented in figure 4.

	BPRMF	GRU4Rec	GRU4Rec+	NextItRec	Caser	SASRec	HGN	Improv.
Recall@10								
MovieLens-20M	0.0774	0.0804	0.0904	0.0833	0.1169	0.1069	**0.1255***	7.36%
Amazon-Books	0.0260	0.0266	0.0301	0.0303	0.0297	0.0358	**0.0429*****	19.83%
Amazon-CDs	0.0269	0.0302	0.0356	0.0310	0.0297	0.0341	**0.0426****	19.66%
GoodReads-Children	0.0814	0.0857	0.0978	0.0879	0.1060	0.1165	**0.1263***	8.41%
GooReads-Comics	0.0788	0.0958	0.1288	0.1078	0.1473	0.1494	**0.1743*****	16.67%
NDCG@10								
MovieLens-20M	0.0785	0.0815	0.0946	0.0828	0.1116	0.1014	**0.1195***	7.07%
Amazon-Books	0.0151	0.0157	0.0173	0.0174	0.0216	0.0240	**0.0298*****	24.17%
Amazon-CDs	0.0145	0.0154	0.0171	0.0155	0.0163	0.0193	**0.0233****	20.73%
GoodReads-Children	0.0664	0.0715	0.0821	0.0720	0.0943	0.1007	**0.1130***	12.21%
GoodReads-Comics	0.0713	0.0912	0.1328	0.1171	0.1629	0.1592	**0.1927*****	18.29%

Figure 2: Recall@k and NDCG@k results

Furthermore, the suggested method also outperformed the others in training time for all of the five datasets.

3.2.9 The 50/50 Recommender: a Method Incorporating Personality into Movie Recommender systems

In paper [30], the authors introduce a movie RS that takes into consideration the user's personality. For this purpose, collaborative filtering methods are being implemented and further combined with a personality test. The dataset used for the implementation of the model is MovieLens 100k which includes 100000 ratings for 1700 different movies and from 1000 unique users. The first step is the rating function which follows the basic steps of collaborative filtering. Initially, the function isolates the ratings of the active user and the algorithm K-NN is implemented based on Pearson correlation to find the users that are more similar to the active one. Then, the products that the similar users preferred are identified and the predictions are generated based on the Prediction Rating Formula. The formula generates the hypothetical rating that the active user would give to the referred products and the top 10 items are recommended based on their scores. The next step is the Big Five Personality test, where the active user is called to reply to 50 questions that identify how his/her personality is structured. The final step is to combine the personality tests with the K-NN algorithm in different two different variations of percentages, 50/50 and 80/20. For the evaluation of the proposed model, 30 people were assigned to try a modified version of the RS. The users tried an RS based only on K-NN, a 50/50 model, and an 80/20 model, and they had to give a score from 1 to 3 to their preferred one, with 3 being the highest rating. 36.92% of the users voted in favor of the 50/50 system, and only 28.72% preferred the 80/20 model. The results imply that considering the user's personality when designing an RS can improve the quality of the system. The authors conclude that more different variations of percentages can be tested to improve the results and raise the point that does to the evaluation process, which cannot guaranty that it is completely random and impartial, the approach can face validity risks.

3.2.10 Promoting Diversity in Content Based Recommendation using Feature Weighting and LSH

In [5] the study focuses on stimulating the diversity of recommendation systems in e-commerce by combining a content-based filtering approach, feature weighting, and LSH (Locality-Sensitive-Hashing). The architecture of the method is constructed in three steps. The first one calculates a weighted matrix to denote the set of products. Then, the Minhash technique is implemented to generate compressed representations of the set of products and to find the highest similarity between sets, using the Jaccard

formula. Lastly, the system generates suggestions based on LSH Forest. Two datasets are used to evaluate the proposed system, Best price dataset, which was created by real e-commerce data, and a prepossessed form of the already existing RetailRocket dataset. In the first step, since each product is presented by a text that contains information about its features, the goal is to generate matrices that indicate the importance of the terms for every item. Afterward, a weight is given to each term by utilizing the TF-IDF formula. The second step is to use the Minhash method to enhance computational speed. The result of this step is that the TF-IDF matrices created at step one for each item, are now represented by the Minhash signature which is much smaller in length. Finally, the LSH Forest method is introduced to find the most similar items by using the Minhash signatures created in the previous step and recommend the k top similar items. The evaluation was constructed separately for each dataset. For the first dataset 11 scenarios were taken into consideration, each one with different feature weights, in order to evaluate the diversity of the recommendations. The results indicate that indeed the system integrates diversity as each generated top 10 recommendation set includes products that belong in different brands, subcategories, and in some cases even in different categories. For the second dataset, the generated dataset of the top 10 recommendations was compared to the actual products the user "visits" in the next product pages. Apparently, in 94% to 97% of the cases, at least one of the recommended items matches the real views of products, unrelatedly to the feature importance. The authors conclude that the results were very encouraging about the performance of the system since diversity is a very important factor for the success of an RS and can be as significant as similarity.

3.2.11 MuSIF: A product Recommendation System Based on Multi-source Implicit Feedback

The authors in [36] propose a new recommender system, MuSIF for product recommendations incorporating multi-source implicit feedback. The model is based on Collaborative filtering with Matrix Factorization and Association Rule Mining. The architecture of the system is implemented in three stages and the main algorithm is a variation of IF CF using Alternative Least Squares (ALS). The first step is the implicit Matrix Factorization. To do so, user-item matrices must be generated as well as implicit rating values that will represent the user-item score. Since the selected algorithm performs better with binary values the cases are divided into "interaction with the item" (denoted by 1) and "no interaction" (denoted by 0). Also, a formula that calculates confidence is created to represent the preference level of a user towards an item. Then, the model calculates the factor vectors by minimizing the proposed cost function and predictions can be achieved by calculating the dot product of the user-item factor vectors. The second step is the Association Rule Mining (ARM), in order to overcome the sparsity problem of the data. It is implemented in order to make the matrices created in the previous step denser with rules extraction and fill in the

missing values. Finally, the last step is the preparation of the factor vectors before the ALS optimization, which is achieved with the introduction of the SVD algorithm for the decomposition of the user-item matrices. The evaluation of the system was conducted with different transformations of the user-item matrix and the chosen metric was the Mean Percentile Ranking (MPR). The dataset used for the evaluation was the Retail rocket including items, users, and events-actions ("view", "transaction", and "add-to-cart"). The model was initially evaluated in terms of performing taking into consideration only the transaction events and then all the three types. The case when only "transactions" participate in the evaluation was called "Single-source evaluation" and the case when all the events participate was named "Multi-source evaluation". To continue with, the system was evaluated in terms of performance when methods that enhance accuracy were applied. So, eventually, four comparisons were constructed between the single-source and multi-source matrices utilizing different methods, the standard, the implementation of ARM, the initialization of user-item factor vectors, and the enhancement of the matrix combined with the initialization of factor vectors. After the implementation of all the methods, the authors conclude that the single-source matrix outperformed the multi-source one in all the methods tested. Even, the best performance of multi-source was not enough to compete with the single-source one which obtained a higher score of 5.18%. The authors support that the results were mainly influenced by the sparsity of the matrices and not necessarily because the single-source is a better model. More specifically, the multi-source model is ten times less dense than the single-source one. Finally, the two models were tested also in terms of coverage and resulted that the multi-source model is more appropriate for generating recommendations as it was able to propose 16022 different items to 43827 users, while the single-source model was able to generate only 106 items to 1056 users.

Chapter 4: Dataset

4.1 Dataset

The dataset introduced in this dissertation is the “Goodreads book reviews” which was created in 2017 from Goodreads.com by collecting more than two million books, found on public shelves. For this reason, all the users along with their reviews are represented by an ID in order to preserve anonymity. The books are further categorized by gender in “Children”, “Comics”, “Fantasy & Paranormal”, “History & Bibliography”, “Mystery, Thriller & Crime”, “Poetry”, “Romance” and “Young Adult”. For the sake of speed and computing power, only one gender was selected for investigation in the dissertation, the “Comics” genre. The dataset is saved under JSON format and contains the user's reviews and ratings [27], [28].

Data description

The subset “goodreads_reviews_comics_graphic.json.gz” consists of 542,338 rows, one for every rating, and 11 columns with relevant information about the reviews. The most important attributes are:

- user_id: unique number for identification for each user
- book_id: unique number for identification for each book
- review_id: unique number for identification for each review
- rating: the user rating on the book
- review_text: the text of the review that the user gave to the book

The first five random rows of the dataset are depicted in figure 5:

user_id	book_id	review_id	rating	review_text	date_added	date_updated	read_at	started_at	n_votes	n_comments
2cae805882878eebb6a32	18471619	66b2ba840f9bd36d6d27f46136fe4772	3	Sherlock Holmes and the Vampires of London \n ...	Thu Dec 05 10:44:25 -0800 2013	Thu Dec 05 10:45:15 -0800 2013	Tue Nov 05 00:00:00 -0800 2013		0	0
200cda7cb2b6acd60cd73	6315584	72f1229aba5a88f9e72f0dcdc007dd22	4	I've never really liked Spider-Man. I am, howe...	Wed Aug 10 06:06:48 -0700 2016	Fri Aug 12 08:49:54 -0700 2016	Fri Aug 12 08:49:54 -0700 2016	Wed Aug 10 00:00:00 -0700 2016	0	0
200cda7cb2b6acd60cd73	29847729	a75309355f8662caaa5e2c92ab693d3f	4	A very quick introduction, this is coming out ...	Thu Apr 21 07:44:00 -0700 2016	Thu Apr 21 07:59:28 -0700 2016	Thu Apr 21 07:59:28 -0700 2016	Thu Apr 21 00:00:00 -0700 2016	0	0
200cda7cb2b6acd60cd73	18454118	c3cc5a3e1d6b6c9cf1c044f306c8e752	5	I've been waiting so long for this. I first st...	Mon Mar 03 17:45:56 -0800 2014	Mon Mar 03 17:54:11 -0800 2014	Sat Mar 01 00:00:00 -0800 2014	Sat Mar 01 00:00:00 -0800 2014	1	0
200cda7cb2b6acd60cd73	2239435	cc444be37ab0a42bfb4dd818cb5edd10	4	The only thing more entertaining than this boo...	Wed Apr 03 12:37:48 -0700 2013	Wed Apr 03 13:03:36 -0700 2013	Wed Apr 03 13:03:36 -0700 2013		0	0

Figure 3: Preview of the first 5 rows of the dataset

4.2 Dataset Preprocessing

In this part, the preprocess and selection of the appropriate variables is descripted and justified. Due to limited resources and computational power only the 50% of the datasets was kept for building the recommender systems. By, using the "train_test_split" function of the "sklearn" library, a random separation of the datasets was achieved. After this procedure the "books" dataset consisted of 44706 entries and the "reviews" dataset of 271169.

Figure 6 and figure 7 provide an insight on the distribution of the user's ratings and their corresponding ratio. As it is evidence, the majority of the books has received a rating over the average and, in particular, the most frequent rating among the reviews is 4. It is important to note that the reviews with rating equal to zero were excluded from the dataset as they did not provide consistent information.

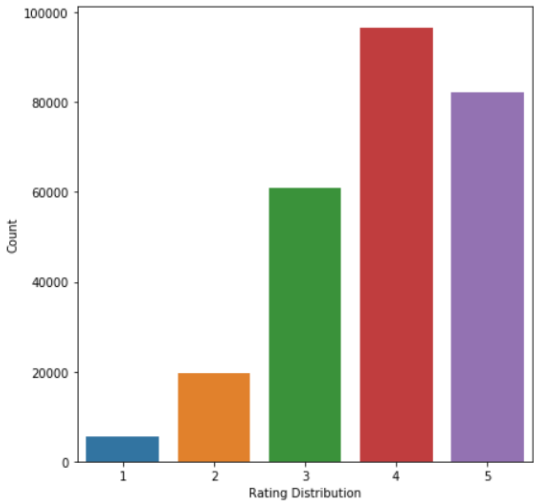

Figure 4: Rating's distribution

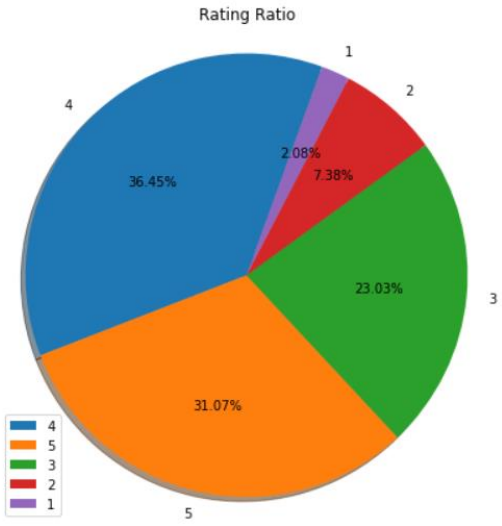

Figure 5: Percentage distribution of the ratings

To continue with, we further process the dataset with a view to avoiding the "cold-start" problem in our recommendation system. This problem refers to the difficulty of the system in generating relevant recommendations when the information provided for the users or the items are not adequate. In our case, this problem can be detected when there are not enough ratings for a book or there are users that have rated very few items. To overcome it, we select to continue only with the books that have been rated more than 10 times and with the users that have rated 50 or more items in our dataset. This step also improves the speed of our program as after the selection of books the dataset entries are decreased to 96220 and after the selection of the users the final dataset consists of 27547 rows. Furthermore, by selecting only a subset of our original dataset we achieve to make it less dense.

Chapter 5: Methodology

Our approach focuses on the creation of a recommender system based on collaborative filtering and sentiment analysis with a view to is to recommend products to customers based on their previous ratings for other products. In order to understand, how and if the comment reviews are a better identifier for the future rating, we initially construct the model using the user ratings, and then we substitute them with the score obtained after the sentiment analysis. For this purpose, the programming language used is Python, as it is one of the most powerful and user-friendly languages, and is widely used in the field of scientific computing. The version of Python implemented in this thesis project is 2.7.13 and the procedures of experimenting and building of the recommender system were done on Jupiter notebook. Jupiter notebook is an open-source web-based application that facilitates the creation and sharing of live codes but also supports procedures like data cleaning, machine learning, and visualization. Different libraries are utilized for the purpose of this study like “Surprise” for the construction and evaluation of the model, “nltk” for text preprocessing, and “TextBlob” for the sentiment analysis.

5.1 Surprise Library

Surprise is a build-in library for the creation and evaluation of recommender systems and specifically focused on rating predictions. It includes classical datasets, as the MovieLens, directly in the package, but user-defined datasets can also be applied by incorporating pandas data frames. The library provides various algorithms for predictions from simple similarity-based ones to algorithms that depend on Matrix Factorization. Furthermore, it involves methods for the evaluation of the models such as cross-validation and methods for hyper-parameter selection like grid search. For this study, the algorithms that are taken under examination are SVD with one variation and K-NN with the other three variations. In addition, the grid-search tool is used for selecting the best parameters for the models and cross-validation for their evaluation [19].

5.2 Sentiment analysis

Sentiment analysis can be described as a process of identification of feelings and emotions from texts, speeches, social media posts, etc. By implementing Natural Language Processing (NLP), the process can categorize the text as negative, positive or neutral and assign a score within the range -1 to 1, with -1 being completely negative and 1 being completely positive.

Initially, the “NLTK” library was used for text processing. The Natural Language Toolkit plays a very important role in transforming the data in a format that can be easier to extract sentiment from, and can support different machine learning algorithms [14]. The steps that were followed to construct the text data in the appropriate form are described below:

1) Transform the reviews data to “string form”

2) Lowercase all reviews (helps increasing the speed of the sentiment calculation)

3) Remove punctuation and special characters (special characters can be described as non-alphabetic and non-numeric values)

4) Remove the stop words (which are very common words with no special predictive power)

5) Remove the suffixes by using the Stemming algorithm with a view to reducing all words with the same root to a common form

Then, the “Text Blob” library is implemented to calculate the sentiment scores. Text Blob is a python library that can extract noun phrases by utilizing NLTK features [24].

By the end of this step, each review is now related to a new rating score that is based on the sentiment extracted from the review text and those scores are going to be used as ratings for the predictions.

5.2 SVD algorithm

The SVD algorithm is met in literature under different names. Some of them are “factor analysis”, “principal component decomposition (PCD)” and “empirical orthogonal function (EOF)”. The name SVD was promoted by Simon Funk during the Netfix Prize competition in 2008. The singular value decomposition is a technique of generating low-rank approximation. The algorithm can generalize easier to higher dimensions. The method decomposes a matrix into three other matrices following the formula [33]:

SVD(A) = U x S x V^T

where:

A is an m x n matrix

U is an m x n orthogonal matrix

S is an n x n diagonal matrix

V is an n x n orthogonal matrix

The diagonal entries (s_1, s_2,..., s_r) of S have the property that $s_i > 0$ and $s_1 \geq s_2 \geq ... \geq s_r$ and are called singular values. The columns of matrix U are called left singular values and the columns of matrix V are called right singular vectors. The algorithm has the property of generating the best low-rank linear approximation of the original matrix A. Since the S entries are ordered, the reduction procedure is implemented by recalling the first k singular values in order to produce the reduced matrix S_k. Then, the matrices U and V are also reduced to generated the matrices U_k and V_k. The U_k and V_k matrices are created by eliminating (r-k) columns and (r-k) rows respectively form the original matrices. The multiplication of the three new reduced matrices produces the matrix A_k, which is the nearest approximation of the original matrix A. The new generated matrix is a rank-k matrix and can be described with the formula:

$A_k = U_k \text{ x } S_k \text{ x } V_k$

According to research, the low-rank approximation of the original space can be an improved solution due to the decreased no=ise that can be introduced into our data deriving for the user-item relation. The SVD algorithm generates an array of uncorrelated eigenvectors where each user and each item are represented by their resembling eigenvector. The lower dimension representation can be really valuable for Collaborative filtering as it contributes in mapping users that have rated similar items into the space traversed by the same eigenvectors.

When the ratings matrix is decomposed the generation of predictions can be done be computing the dot products between m pseudo-users and n pseudo-items [33]. The surprise library in python includes the basic SVD algorithm and an extension to that which is denoted as SVDpp or SVD++. The predictions of SVD are calculated based on the formula:

$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T \text{ x } p_u$

where μ represents the average rating, b_i, b_u represent the observed deviations of items and users from the average (bias) and the dot product represents the interest of user u towards item i. To obtain the factors b, q and p the model minimizes the regularized squared error on the array of known ratings as described by the formula:

$\min_{p,q,b} \sum_{u,i \in \kappa} (r_{ui} - \mu - b_u - b_i - q_i^T p_u)^2 + \lambda (\|p_u\|^2 + \|q_i\|^2 + b_u{}^2 + b_i{}^2)$

where

- k is the array of user-item pairs for which the ratings are known
- λ is the regularization term

So, the model is training by fitting the already observed ratings with a view to generalizing to the future predictions [21].

The extension of SVD takes into account also the implicit ratings by initializing the y_j

terms that describe the case of a user u rated the item j regardless of the rating value and the above prediction formula is transformed as follows:

$\hat{r}_{ui} = \mu + b_i + b_u + q_i^T \text{ x } (p_u + |I_u|^{-1/2} \sum_{j \in I_u} y_j)$

The most important parameters of the models that will be tested in order to find the best values for them during the grid search are "n_epochs", "lr_all" and "reg_all" [19].

- n_epochs describes the iterations of stochastic gradient descent which is implemented for the minimization of regularized squared error
- lr_all expresses the learning rate of the parameters
- reg_all expresses the regularization of parameters

5.3 K-NN algorithm

K-NN is one of the simplest, yet most famous algorithms in supervised machine learning which can be further applied for recommendation systems. It belongs to the category of "lazy" algorithms and it is a non-parametric method based on the similarity between the distance of two instances x and y, using different distance measures like Euclidian, Manhattan distance or cosine similarity. The core idea behind the algorithm is that if the majority of the k most similar neighbors of our entry belong to a certain category, then our entry belongs to the same category. In our case, K-NN can be applied to calculate both user and item similarities. One of the hardest decisions than need to be taken refer to selection of the appropriate neighbors' number k [1]. This obstacle is tackled by implemented a grid search for identifying our model's best parameters as will be described in the next section.

In surprise library different variations of the classic K-NN algorithm are included as described below:

- KNN Basic

  The prediction of the ratings is given by the following formula(equation 14) which is based on the similarities between users:

$$\hat{r}_{ui} = \frac{\sum\limits_{v \in N_i^k(u)} \text{sim}(u,v) \cdot r_{vi}}{\sum\limits_{v \in N_i^k(u)} \text{sim}(u,v)}$$

Equation 14: KNN (basic) predictions

- KNN with Z score

This variation also takes into account the z-score normalization for each user and the predicted ratings are given by the formula, where μ_u is the score for each user:

$$\hat{r}_{ui} = \mu_u + \sigma_u \frac{\sum\limits_{v \in N_i^k(u)} \text{sim}(u,v) \cdot (r_{vi} - \mu_v)/\sigma_v}{\sum\limits_{v \in N_i^k(u)} \text{sim}(u,v)}$$

Equation 15: KNN (with Z-score) Predictions

- KNN with Means

This variation takes into account the mean ratings and can be implemented both for the user and the item. So, in the first case it considers as well the mean rating of each user and in the second one, it considers the mean rating of each item. The predicted ratings can be calculated with the following formula (equation 16):

$$\hat{r}_{ui} = \mu_u + \frac{\sum\limits_{v \in N_i^k(u)} \text{sim}(u,v) \cdot (r_{vi} - \mu_v)}{\sum\limits_{v \in N_i^k(u)} \text{sim}(u,v)}$$

Equation 16: KNN (with Means) Predictions

The most important parameters of all the variations of KNN that will be set under examination during the grid search are the “bsl_options”,“k” and “sim_options” where,

- bsl_options represent a dictionary of options for the baseline estimates computation. Baselines can be defined in two ways, using Stochastic Gradient Descent (SGD) or using Alternating Least Squares (ALS) [19].
- k denotes the maximum number of neighbors that should be taken into consideration [19].
- sim_options denotes the distance measure that should be used from the available ones and also involves the parameter “user_based” which can

take the values True or False depending on whether we want to compute similarities between users or items [19].

5.4 Grid search for hyper-parameter selection

Grid search is a tuning technique widely known in the field of machine learning that tries to compute the optimum values of hyperparameters. In this study we introduce the grid search method which is simply an exhaustive searching over a manually determined set of the hyperparameter space. The performing metric used to complete the search is cross validation of 5 folds and the accuracy metrics to define the best parameters are Root Square Mean Error (RMSE) and Mean Average Error (MAE) that where described in section 2.4. In the following images we can see the part of the python code used for the grid search, the outcome for every algorithm and the computational time of the procedure.

a) user ratings

```
#grid search for the SVD algorithms
import time
start_time = time.time()
svd_param_grid = {'n_epochs': [20, 25], 'lr_all': [0.007, 0.009, 0.01], 'reg_all': [0.4, 0.6]}

svd_gs = GridSearchCV(SVD, svd_param_grid, measures=['rmse', 'mae'], cv=5, n_jobs=5)
svdpp_gs = GridSearchCV(SVDpp, svd_param_grid, measures=['rmse', 'mae'], cv=5, n_jobs=5)

svd_gs.fit(data)
svdpp_gs.fit(data)

# best RMSE score
print(svd_gs.best_score['rmse'])
print(svdpp_gs.best_score['rmse'])
# best MAE score
print(svd_gs.best_score['mae'])
print(svdpp_gs.best_score['mae'])

# combination of parameters that gave the best RMSE and MAE scores
print(svd_gs.best_params['rmse'])
print(svdpp_gs.best_params['rmse'])
print(svd_gs.best_params['mae'])
print(svdpp_gs.best_params['mae'])
computational_time = time.time() - start_time
print('\n Computational Time : %0.3fs' %(computational_time))
```

```
0.8481507126869428
0.8473491717109984
0.6727363888531724
0.6713612504301303
{'n_epochs': 25, 'lr_all': 0.01, 'reg_all': 0.4}
{'n_epochs': 25, 'lr_all': 0.009, 'reg_all': 0.4}
{'n_epochs': 25, 'lr_all': 0.01, 'reg_all': 0.4}
{'n_epochs': 25, 'lr_all': 0.01, 'reg_all': 0.4}

 Computational Time : 2172.061s
```

Figure 6: Grid search results and computational time for SVD algorithms (a)

```
start_time = time.time()
knn_param_grid = {'bsl_options': {'method': ['als', 'sgd'],
                                  'reg': [1, 2]},
                  'k': [15, 20, 25, 30, 40, 50, 60],
                  'sim_options': {'name': ['msd', 'cosine', 'pearson_baseline']}
                  }

knnbasic_gs = GridSearchCV(KNNBasic, knn_param_grid, measures=['rmse', 'mae'], cv=5, n_jobs=5)
knnmeans_gs = GridSearchCV(KNNWithMeans, knn_param_grid, measures=['rmse', 'mae'], cv=5, n_jobs=5)
knnz_gs     = GridSearchCV(KNNWithZScore, knn_param_grid, measures=['rmse', 'mae'], cv=5, n_jobs=5)


knnbasic_gs.fit(data)
knnmeans_gs.fit(data)
knnz_gs.fit(data)

# best RMSE score
print(knnbasic_gs.best_score['rmse'])
print(knnmeans_gs.best_score['rmse'])
print(knnz_gs.best_score['rmse'])
# best MAE score
print(knnbasic_gs.best_score['mae'])
print(knnmeans_gs.best_score['mae'])
print(knnz_gs.best_score['mae'])

# combination of parameters that gave the best RMSE and MAE scores
print(knnbasic_gs.best_params['rmse'])
print(knnmeans_gs.best_params['rmse'])
print(knnz_gs.best_params['rmse'])
print(knnbasic_gs.best_params['mae'])
print(knnmeans_gs.best_params['mae'])
print(knnz_gs.best_params['mae'])
computational_time = time.time() - start_time
print('\nComputational Time : %0.3fs' %(computational_time))
```

```
0.9312870320866022
0.8716973615038824
0.8723805619110282
0.7349301162050467
0.6804197713587044
0.6790398561759651
{'bsl_options': {'method': 'als', 'reg': 1}, 'k': 20, 'sim_options': {'name': 'msd', 'user_based': True}}
{'bsl_options': {'method': 'als', 'reg': 1}, 'k': 60, 'sim_options': {'name': 'cosine', 'user_based': True}}
{'bsl_options': {'method': 'als', 'reg': 1}, 'k': 60, 'sim_options': {'name': 'cosine', 'user_based': True}}
{'bsl_options': {'method': 'als', 'reg': 1}, 'k': 15, 'sim_options': {'name': 'msd', 'user_based': True}}
{'bsl_options': {'method': 'als', 'reg': 1}, 'k': 60, 'sim_options': {'name': 'cosine', 'user_based': True}}
{'bsl_options': {'method': 'als', 'reg': 1}, 'k': 60, 'sim_options': {'name': 'cosine', 'user_based': True}}

Computational Time : 1611.386s
```

Figure 7: Grid search results and computational time for KNN algorithms (a)

In particular, the best parameters for both of the SVD algorithms regarding the results of rmse are “n_epochs” = 25, “lr_all” = 0.01 and “reg_all” = 0.4. Regarding the mae the best parameters are “n_epochs” = , “lr_all” = and “reg_all”= .The computational time for this search was approximately 2172.061 seconds. We select to continue with the best parameters of RMSE in order to decrease the computational time and furthermore because both RMSE and MAE “agree” on the outcome.

For the KNN algorithms regarding the results of RMSE the best parameters are the alternative least squares for the estimation of the baseline approach, For the basic KNN the similarity metric is MSD and the similarities should be computed on the users. The number of k differs for RMSE and MAE but we decide to proceed with the results of RMSE so the number of nearest neighbors is 20. MSD stands for Mean Squared Difference which computes the similarity between all pairs of users (or items). Only the common users or common items are taking into consideration and the formula to calculate the similarities can be described as follows:

$$\text{msd}(u, v) = \frac{1}{|I_{uv}|} \cdot \sum_{i \in I_{uv}} (r_{ui} - r_{vi})^2$$

Equation 17: Mean squared difference

For the other two approaches the number for nearest neighbors is 60 and the best similarity metric is cosine similarity, the formula of which is defined in section 2.3. Regarding the MAE the best parameters agree with the parameters of RMSE. The computational time for this search was approximately 1611 seconds. Again, we select only to continue with the best parameters in terms of RMSE.

b) ratings after sentiment analysis

```
import time
start_time = time.time()
svd_param_grid = {'n_epochs': [20, 25], 'lr_all': [0.007, 0.009, 0.01], 'reg_all': [0.4, 0.6]}

svd_gs = GridSearchCV(SVD, svd_param_grid, measures=['rmse', 'mae'], cv=5, n_jobs=5)
svdpp_gs = GridSearchCV(SVDpp, svd_param_grid, measures=['rmse', 'mae'], cv=5, n_jobs=5)

svd_gs.fit(data)
svdpp_gs.fit(data)

# best RMSE score
print(svd_gs.best_score['rmse'])
print(svdpp_gs.best_score['rmse'])
# best MAE score
print(svd_gs.best_score['mae'])
print(svdpp_gs.best_score['mae'])

# combination of parameters that gave the best RMSE and MAE scores
print(svd_gs.best_params['rmse'])
print(svdpp_gs.best_params['rmse'])
print(svd_gs.best_params['mae'])
print(svdpp_gs.best_params['mae'])
computational_time = time.time() - start_time
print('\n Computational Time : %0.3fs' %(computational_time))
```

```
0.2085892248131433
0.20821449921781326
0.15317631900808623
0.15292155997294468
{'n_epochs': 20, 'lr_all': 0.007, 'reg_all': 0.6}
{'n_epochs': 20, 'lr_all': 0.007, 'reg_all': 0.6}
{'n_epochs': 20, 'lr_all': 0.007, 'reg_all': 0.6}
{'n_epochs': 20, 'lr_all': 0.007, 'reg_all': 0.6}

 Computational Time : 2027.476s
```

Figure 8: Grid search results and computational time for SVD algorithms (b)

According to figure 10, the best parameters for both of the SVD algorithms regarding the results of RMSE and MAE are “n_epochs” = 20, “lr_all” = 0.007 and “reg_all” = 0.6. The computational time for this search was approximately 2027 seconds.

```
start_time = time.time()
knn_param_grid = {'bsl_options': {'method': ['als', 'sgd'],
                                  'reg': [1, 2]},
                  'k': [15, 20, 25, 30, 40, 50, 60],
                  'sim_options': {'name': ['msd', 'cosine', 'pearson_baseline']}
                  }

knnbasic_gs = GridSearchCV(KNNBasic, knn_param_grid, measures=['rmse', 'mae'], cv=5, n_jobs=5)
knnmeans_gs = GridSearchCV(KNNWithMeans, knn_param_grid, measures=['rmse', 'mae'], cv=5, n_jobs=5)
knnz_gs     = GridSearchCV(KNNWithZScore, knn_param_grid, measures=['rmse', 'mae'], cv=5, n_jobs=5)


knnbasic_gs.fit(data)
knnmeans_gs.fit(data)
knnz_gs.fit(data)

# best RMSE score
print(knnbasic_gs.best_score['rmse'])
print(knnmeans_gs.best_score['rmse'])
print(knnz_gs.best_score['rmse'])
# best MAE score
print(knnbasic_gs.best_score['mae'])
print(knnmeans_gs.best_score['mae'])
print(knnz_gs.best_score['mae'])

# combination of parameters that gave the best RMSE and MAE scores
print(knnbasic_gs.best_params['rmse'])
print(knnmeans_gs.best_params['rmse'])
print(knnz_gs.best_params['rmse'])
print(knnbasic_gs.best_params['mae'])
print(knnmeans_gs.best_params['mae'])
print(knnz_gs.best_params['mae'])
computational_time = time.time() - start_time
print('\nComputational Time : %0.3fs' %(computational_time))
```

```
0.2211655447480705
0.21689884120517444
0.21719709949101068
0.1653368323354964
0.15973849856136405
0.15815741479399043
{'bsl_options': {'method': 'als', 'reg': 1}, 'k': 50, 'sim_options': {'name': 'msd', 'user_based': True}}
{'bsl_options': {'method': 'als', 'reg': 1}, 'k': 60, 'sim_options': {'name': 'msd', 'user_based': True}}
{'bsl_options': {'method': 'als', 'reg': 1}, 'k': 60, 'sim_options': {'name': 'msd', 'user_based': True}}
{'bsl_options': {'method': 'als', 'reg': 1}, 'k': 25, 'sim_options': {'name': 'msd', 'user_based': True}}
{'bsl_options': {'method': 'als', 'reg': 1}, 'k': 50, 'sim_options': {'name': 'msd', 'user_based': True}}
{'bsl_options': {'method': 'als', 'reg': 1}, 'k': 60, 'sim_options': {'name': 'msd', 'user_based': True}}

Computational Time : 2437.201s
```

Figure 9: Grid search results and computational time for KNN algorithms (b)

For the KNN algorithms regarding the results of RMSE (figure 11) the best parameters are the alternative least squares for the estimation of the baseline approach. In addition, the similarity metric is MSD and the similarities should be computed on the users as the "user_based" parameter was set to "True". The number of neighbors differs between the models and the two metrics but we decide to proceed with the parameters computed based on RMSE. So, for the basic KNN the number of k is 50 and for the other two modes it is 60. The computational time was approximately 2437 seconds.

5.5 Cross-validation for the selection of the best algorithm

K-fold cross–validation is an evaluating technique for machine learning models during which, the input data are divided randomly into k equal segments. In each

iteration, it uses k-1 folds of data for training and the remaining data for estimation. Using accuracy as the assessment metric, the predictive performance of the model is evaluated on each fold [7]. In our study we select to perform a 5-fold cross validation to calculate the "rmse" and "mae" of our algorithms in order to decide which one will provide the best predictions.

a) user-ratings

In the first case where we use the ratings provided by the users as input for the future predictions, we test the proposed models using a 5-fold cross validation. The accuracy metrics taking into consideration are again RMSE and MAE. We also calculate the fitting and testing time as well as the over computational time. The results are presented in Fig. 12.

	Model	RMSE	MAE	Fit Time	Test Time
0	SVD	0.847311	0.671564	5.244589	0.148903
1	SVDpp	0.847516	0.672370	56.076903	0.875285
2	KNNBasic	0.931686	0.736672	0.118993	0.470716
3	KNNWithZScore	0.872473	0.679365	0.547099	0.656536
4	KNNWithMeans User-User	0.869168	0.680460	0.518675	0.585679
5	KNNWithMeans Item-Item	0.868836	0.679756	1.853398	1.255250

Figure 10: 5-fold cross validation results (a)

The overall computational time is presented in the following table. It is obvious that the fastest algorithm is the KNN basic but, it is not the one with the best performance.

Table 2: Computational time of all algorithms (a)

Model	Computational time (in seconds)
SVD	33.132
SVDpp	328.222
KNN basic	4.297
KNN with Z score	7.931
KNN with Means (user-user)	7.3
KNN with Means (item-item)	18.875

Furthermore, we visually present the results for the different algorithms in a RMSE and MAE plot (fig. 12-13). The best scores are achieved in both of the cases by the

SVD algorithm, which will be the one selected to generate the future recommendations.

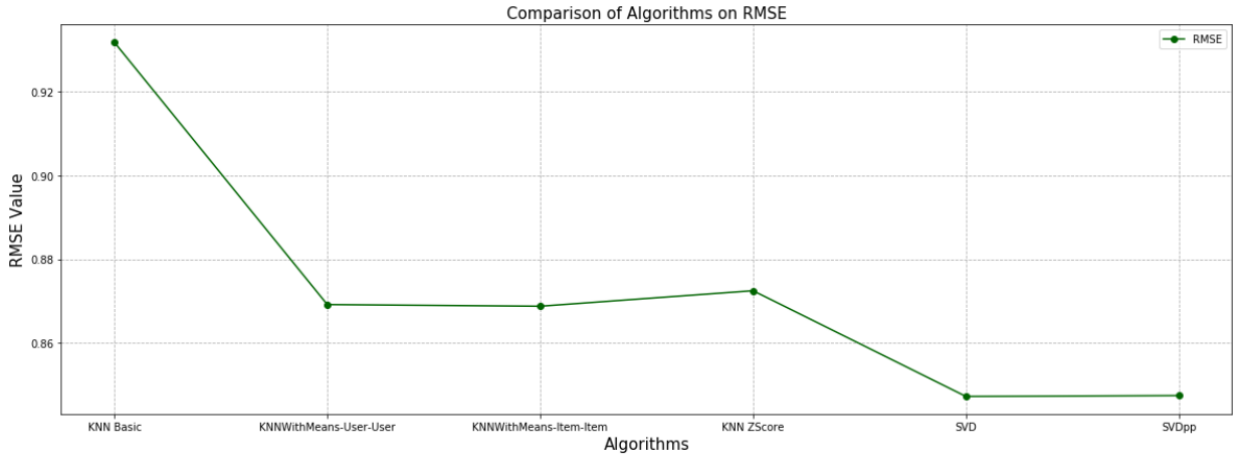

Figure 11: RMSE plot (a)

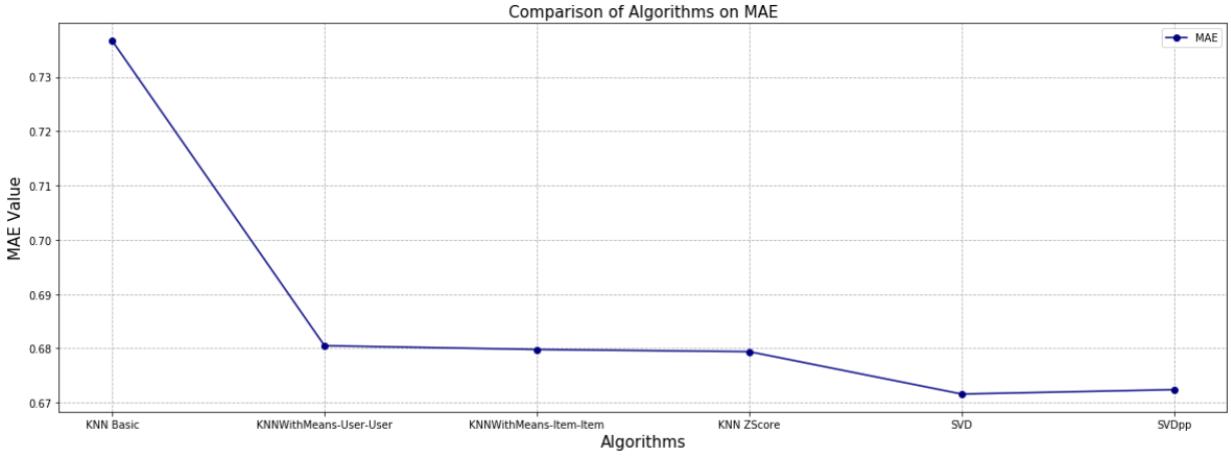

Figure 12: MAE plot (a)

b) sentiment analysis scores

After implementing the sentiment analysis on the user reviews, we repeat the same procedure in order to evaluate the proposed algorithms and conclude to the best model for our predictions. The table below depicts the outcome for the 6 models in terms of RMSE, MAE, fit time and test time.

	Model	RMSE	MAE	Fit Time	Test Time
0	SVD	0.208737	0.153293	3.537732	0.093498
1	SVDpp	0.208349	0.153028	42.204102	0.894864
2	KNNBasic	0.221270	0.165770	0.125818	0.501355
3	KNNWithZScore	0.217343	0.158484	0.244830	0.703047
4	KNNWithMeans User-User	0.217566	0.160155	0.158462	0.640314
5	KNNWithMeans Item-Item	0.217287	0.159953	0.564232	1.684060

Figure 13: 5-fold cross validation results

The overall computational time is presented in the following table. It is obvious that the KNN based algorithms are much faster, but yet as presented in fig. 15-17, they do not have the best performance.

Table 3: Computational time of all algorithms (b)

Model	Computational time (in seconds)
SVD	22.656
SVDpp	289.274
KNN basic	5.205
KNN with Z score	6.568
KNN with Means (user-user)	5.503
KNN with Means (item-item)	10.263

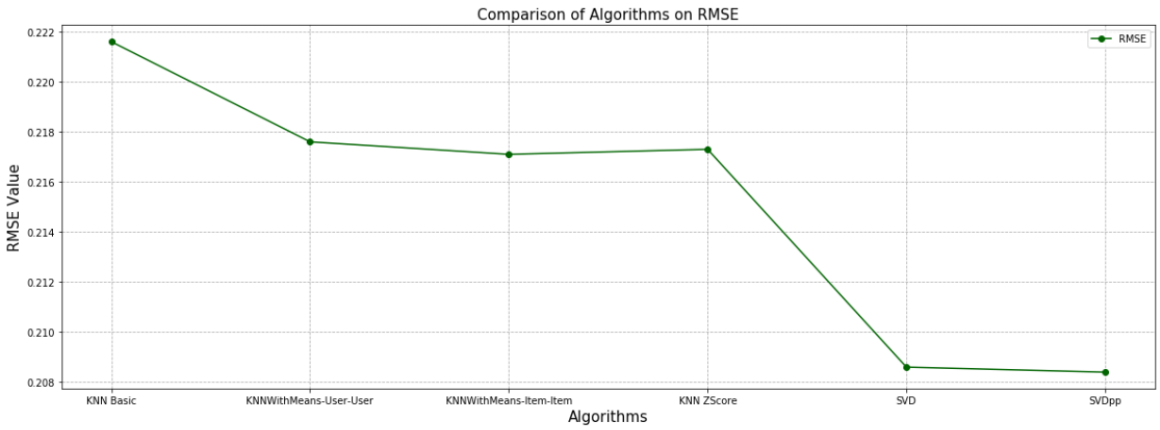

Figure 14: RMSE plot (b)

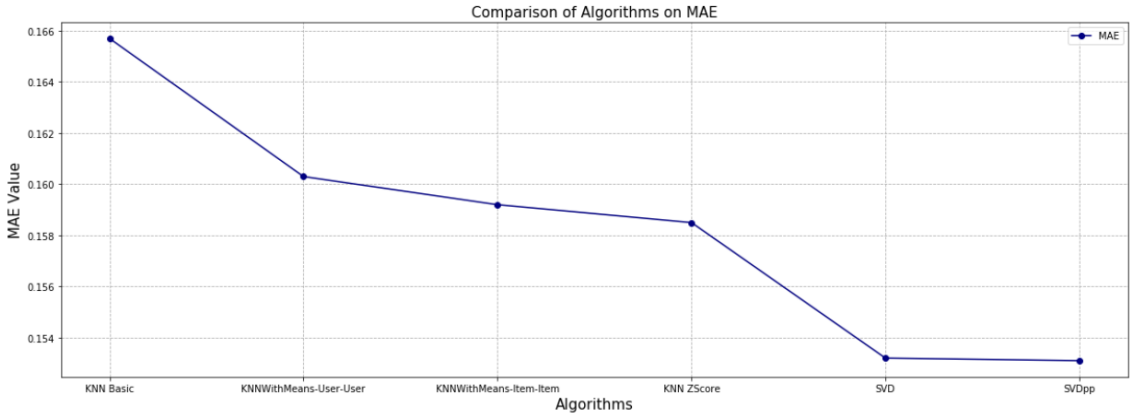

Figure 15: MAE plot (b)

In this case the best performing model is the SVDpp with a very small difference from the basic SVD. The predictions on the user's ratings will be computed using this model and the best combination of parameters.

Chapter 6: Results and Evaluation

The two metrics used for the evaluation are precision@k and recall@k. Precision@k denotes the number of relevant items that were recommended on the top k generated recommendations so in other terms the proportion of correctly recommended items over the total number of items. Recall@k denotes the correctly recommended item over the set of relevant items.

a) user ratings

For the user ratings we tested the precision and recall of the SVD algorithm, for the top 5 and 10 recommendations using 5-fold and 10-fold cross validation. The results of the mean precision and recall along with their standard deviations are presented in table 4 and table 5. We test the precision and recall in the first 5 and the first 10 recommended items to the user and examine the number of the relevant items that are recommended and the total of the proportion of the relevant items included in the recommendations.

Table 4: Precision results (a)

k/ No-fold cross validation	5-fold cross validation	10-fold cross validation
5	80.266% (0.683%)	78.573% (1.039%)
10	80.406% (0.658%)	81.634% (1.152%)

Table 5: Recall results (a)

k/ No-fold cross validation	5-fold cross validation	10-fold cross validation
5	61.679% (1.282%)	55.398% (1.310%)
10	69.445% (0.551%)	74.411% (1.152%)

In terms of precision and recall the best score is achieved with a 10-fold cross validation and by calculating the top 10 recommendations. In particular the score explains that from the first 10 recommended items the 8 are relevant to the user's

preferences and the 74% of the relevant items are recommended in the top 10 suggestions. In general, we see that all results regarding precision are close to 80% and the ones regarding recall are close to 65%. The outcomes generated support the idea that the using the methodology described in chapter 5, we can obtain good recommendations based on the ratings of the users.

In figures 16-17 we can see the plot of precision and recall at top 5 items with a 5-fold cross-validation and then, the plot of the best performing combination. the top 10 recommended items with a 10-fold cross-validation.

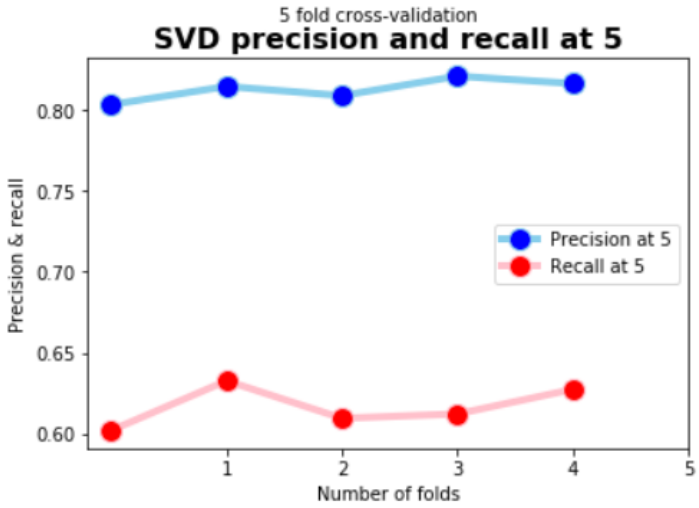

Figure 16: SVD precision and recall plot (5-fold cross validation) (a)

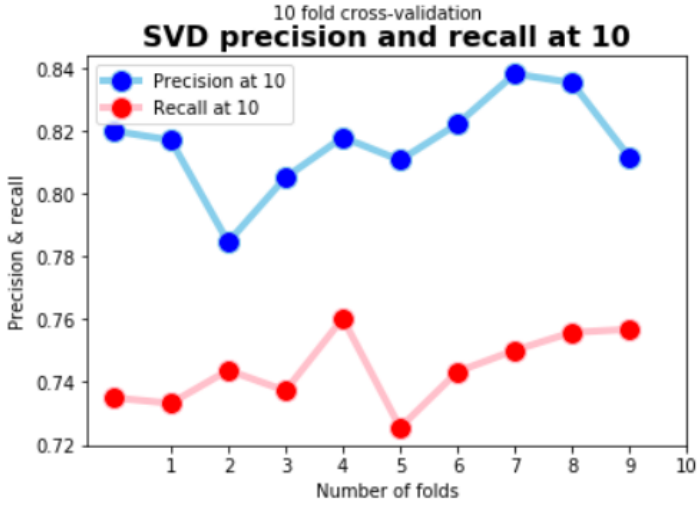

Figure 17: SVD precision and recall plot (10-fold cross validation) (a)

b) ratings of sentimental analysis

Then, we examined the prementioned metrics over the second case which uses the method of sentiment analysis to the comments in order to assign each text to a rating number. In tables 6 and 7, the results of this experiment using a variation of the basic SVD algorithm, are presented. We again examine the cases of the top 5 and top 10 recommended items with a 5-fold and 10-fold cross validation.

Table 6: Precision results (b)

k/ No-fold cross validation	5-fold cross validation	10-fold cross validation
5	88.419% (0.0522%)	89.148% (0.306%)
10	88.333% (0.475%)	89.329% (0.244%)

Table 7: Recall results (b)

k/ No-fold cross validation	5-fold cross validation	10-fold cross validation
5	46.020% (1.700%)	59.416% (2.303%)
10	45.774% (1.460%)	59.591% (1.406%)

In terms of precision and recall the best score is achieved for the top 10 recommendations with a 10-fold cross validation. In particular almost 9 out of the 10 items that are recommended to the user are relevant and around 60% of the relevant items are recommended in the top 10.

To continue with, in figures 20 and 21, we can see the results at k=5, for a 5-fold cross-validation then the best performing combination of variables which is the top 10 recommendations with a 10-fold cross validation. In both cases, users' ratings and the score obtained after the sentiment analysis, the best results are obtained by the same combination of attributes. In general, we see that all results regarding precision are close to 90% and the ones regarding recall are close to 55%.

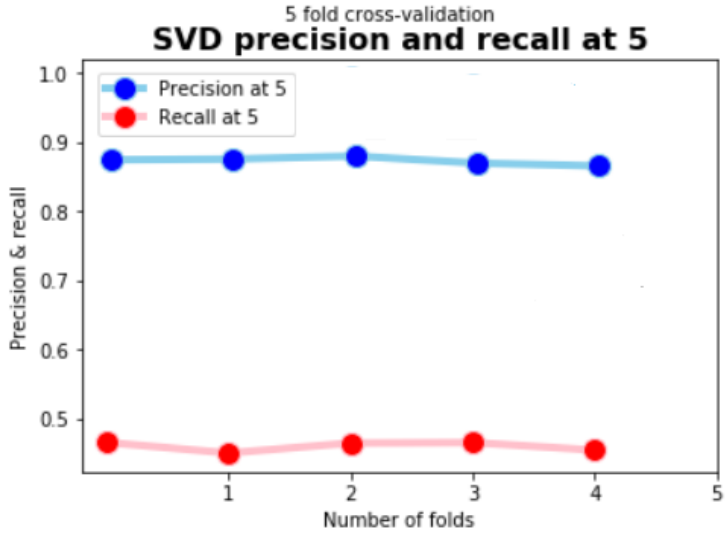

Figure 18: SVDpp precision and recall plot (5-fold cross-validation) (b)

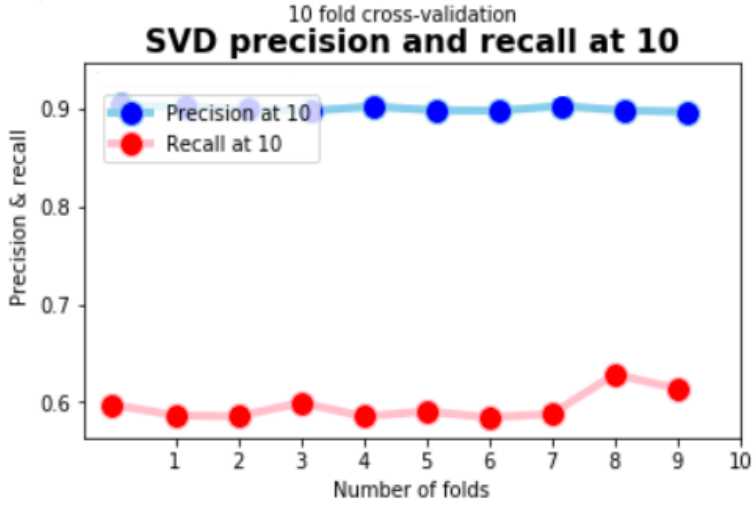

Figure 19: SVDpp precision and recall plot (10-fold cross-validation) (b)

As observed by using the ratings obtained by the sentiment analysis as the indicator for the future ratings we result in an increase in the precision of our model. At the same time, as the precision increases the recall decreases since there is a trade-off between those metrics. So, we cannot improve the one without effecting the other.

To obtain a more general view we can also introduce the F-score for both cases calculated by the formula in equation 6. For the user rating case, we achieve an F-score@10 = 0.78 and for the case of sentiment analysis, we achieve an F-score@10 = 0.72.

The main conclusion that can be drawn by observing the results is that in case (b) we were able to achieve a better score for precision but at the same time, we dropped in the score of recall. This means that less of the relevant items in total are recommended to the users while the final recommendations are more relevant. Both attempts result

in good scores and the fact that in general, the initial experiment resulted in a higher total does not mean that the second approach can be also useful.

For a recommender system, precision can be more important than recall, as it depicts the number of relevant recommendations that the user gets, while recall simply expresses the performance of the system in recommending as less irrelevant items as possible [40]. In the case of creating a book RS as in our project, the decrease of false positives or in other terms the better precision scores from a user point of view is more important. Having more false positives can be costly to the system, as the possibility of the users purchasing a new item decreases. Furthermore, if the recommendations are not relevant enough, it is possible that the user might eventually leave our platform due to the lack of interest. Since the better precision was achieved by the analysis of the textual comments, we can assume that the new scores provide a more detailed insight about the preferences of the users than a simple integer number.

On the other hand, if we select to follow the common approach of using the ratings, we can achieve a higher recall, which means that a higher percentage of relevant items are recommended to the end-user but the top recommendations are not that relevant as in the previous case. An explanation for the higher recall score could be the more limited dataset in the second case, as for the sentiment analysis it is obligatory for the users to have placed a textual comment on the item. It is impossible to be included in the dataset of the second case if the user has not provided a comment, while in the first case we can generate results just by the ratings. This means that maybe some relevant items were eventually excluded and never recommended to the users because of the lack of textual reviews. In the case of a higher recall, the benefit is that there is a bigger variety of items that are suggested so there are bigger chances that all the relevant items appear to the end-users.

It is important to mention that the measures used for the evaluation have the drawback of not being able to also test the positions in which the items are proposed to the users.

It is also important to highlight some of the threats to the validity of the findings and facts that can influence the results. In the case of the user rating, the results can be generalized more easily as the 0-5 rating is an indicator that includes fewer biases than the comments. Since the sentiment analysis was contacted using a pre-set python library, the Textblob library it is practically impossible to test whether the comments are correctly assigned to a score and if this score depicts the actual evaluation of the user. Furthermore, in the second case, we can only use the part of our dataset that includes a user's comment but, in many cases, the information gathered about an item is lacking the textual critic. As a result, it is certainly easier to generalize the results of the first experiment that is based only on the numerical ratings.

Another important point that might influence the results is that the library used to extract the sentiment of the user comments is performing significantly well in English but for other languages, the results are not so good since the library uses Google Translate to detect the language and process it. It is possible that important information is lost during that process or maybe the final outcome is not that valid with respect to the original test.

In addition, the results of these experiments are depicting the best possible outcomes. In each case, the threshold that discriminates between a relevant and an irrelevant recommendation is found through trial and error and by testing different levels. As a consequence, if we try to generalize to different datasets even of the same field or even from the same site, the results might differ.

Finally, the study was conducted only for a specific genre of books and, in particular the comic one, so, the conclusions drawn refer to a specific category of people that read comics and potentially share common characteristics. An experiment contacted in the whole Goodreads dataset would be in a position to provide more general results at least for this type of item but, unfortunately, due to limited resources, this was not possible.

Chapter 7: Conclusion and Future Work

7.1 Conclusion

Based on the recommendation system and the results that were generated after different experiments some interesting results have been extracted. This dissertation project follows the technique of collaborative filtering and examines multiple algorithms. After testing each one of them, taking into consideration the RMSE and MAE scores, the best performing algorithm appears to be the SVD which is based on the principle of singular value decomposition.

The main subject of research in this project was to examine how the recommendations might change, when, instead of using the actual ratings that the user provides regarding an item, we use the available textual information, i.e., the comments. The dataset used in this study was extracted by the goodreads.com and consisted of the book comic collection of the website.

Initially, using the SVD algorithm and the best parameters which were found after a grid search, the best results in terms of precision@k and recall@k were obtained with a 10-fold cross validation and when the predictions were regarding the top 5 relevant items. After processing the textual information of each review, we repeated the procedure using a variation of the SVD algorithm since it achieved better RMSE and MAE scores when tested on the new data. The outcome was an increase in the precision@k and a decrease in the recall@k. The best results were obtained again with a 10-fold cross-validation and the top 10 predictions.

In terms of precision@k after taking into consideration the textual information we were able to achieve an increase of almost 10%. This means that after examining the commends and the system used them as input data, more relevant recommendations to the preferences of the user were generating. Since recall and precision are a trade-off, the achieved increase in precision can only result in a decrease in the recall@k which leds to a smaller percentage of the total relevant items being recommended. In order to test the experiment in a more general way, the F-score@k was calculated as well and the result was a decrease of 0.06%. This can indicate that the first experiment is already a good model but depending on the way we want to design our system and which metric we consider to be more important we can result in meaningful and valuable outcome also with the second experiment.

7.2 Future Work

Future work should examine more datasets to validate whether the same results can be generalized. It could also test different fields of recommendations and expand in the field of book recommendations.

Furthermore, more algorithms could be tested in order to achieve potentially better results. In the field of collaborative filtering, an extenction could be a combination of both the user ratings and also the scores achieved after implementing a sentiment analysis on the user comments.

Another expansion of this work could be the creation of a hybrid recommendation system that will also combine content-based filtering techniques to improve the results of this study.

Finally, the next step should be a deeper and more insightful testing on how the tradeoff between precision@k and recall@k works and whether there is a chance of achieving better results in terms of both metrics or even utilize new ones to test the outcome of this project.

References

1. Adeniyi, D. A., Wei, Z., & Yongquan, Y. (2016). Automated web usage data mining and recommendation system using K-Nearest Neighbor (KNN) classification method. Applied Computing and Informatics, 12(1), 90-108

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

3. Aggarwal, C. C. (2016). Recommender Systems: The Textbook

4. Baatarjav, E. A., Phithakkitnukoon, S., & Dantu, R. (2008, November). Group recommendation system for facebook. In OTM Confederated International Conferences" On the Move to Meaningful Internet Systems" (pp. 211-219). Springer, Berlin, Heidelberg

5. Beleveslis, D., & Tjortjis, C. (2020, June). Promoting Diversity in Content Based Recommendation Using Feature Weighting and LSH. In IFIP International Conference on Artificial Intelligence Applications and Innovations (pp. 452-461). Springer, Cham.

6. Bennett, J., & Lanning, S. (2007, August). The netflix prize. In Proceedings of KDD cup and workshop (Vol. 2007, p. 35).

7. Browne, M. W. (2000). Cross-validation methods. Journal of mathematical psychology, 44(1), 108-132. Koren, Y., Bell, R., & Volinsky, C. (2009).
8. Burke, R. (2000). Knowledge-based recommender systems. Encyclopedia of library and information systems, 69(Supplement 32), 175-186

9. Candillier, L., Jack, K., Fessant, F., & Meyer, F. (2009). State-of-the-art recommender systems. In Collaborative and Social Information Retrieval and Access: Techniques for Improved User Modeling (pp. 1-22). IGI Global

10. Davidson, J., Liebald, B., Liu, J., Nandy, P., Van Vleet, T., Gargi, U., ... & Sampath, D. (2010, September). The YouTube video recommendation system. In Proceedings of the fourth ACM conference on Recommender systems (pp. 293-296).

11. Davies, J. E. What shall I read next? Developing tools for reader support. Proceedings of IFLA General Conference and Council, Vol. 68, 2002

12. Eckhardt, A. (2009). Various aspects of user preference learning and recommender systems. In DATESO (pp. 56-67).

13. Frigui, H., Krishnapuram, R.: Clustering by competitive agglomeration. Pattern Recognition Journal 30(7), 1109–1119 (1997)

14. Garg, P., & Bassi, V. G. (2016). Sentiment analysis of twitter data using NLTK in python (Doctoral dissertation)

15. Gomez-Uribe, C. A., & Hunt, N. (2015). The netflix recommender system: Algorithms, business value, and innovation. ACM Transactions on Management Information Systems (TMIS), 6(4), 1-19.

16. Hariadi, A. I., & Nurjanah, D. (2017, November). Hybrid attribute and personality based recommender system for book recommendation. In 2017 International Conference on Data and Software Engineering (ICoDSE) (pp. 1-5). IEEE.

17. Herlocker, J. L., Konstan, J. A., Terveen, L. G., & Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems (TOIS), 22(1), 5-53

18. Huang, Z., Chung, W., Ong, T. H., & Chen, H. (2002, July). A graph-based recommender system for digital library. In Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries (pp. 65-73).

19. Hug, N. (2020). Surprise: A Python library for recommender systems. Journal of Open Source Software, 5(52), 2174.

20. Kanetkar, S., Nayak, A., Swamy, S., & Bhatia, G. (2014, August). Web-based personalized hybrid book recommendation system. In 2014 International Conference on Advances in Engineering & Technology Research (ICAETR-2014) (pp. 1-5). IEEE.

21. Koren, R. Bell and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," in Computer, vol. 42, no. 8, pp. 30-37, Aug. 2009, doi: 10.1109/MC.2009.263

22. Lee, S. An Implicit Feedback Model for Goodreads Recommendations.

23. Liu, Y., Xie, M., & Lakshmanan, L. V. S. (2014). Recommending user generated item lists. Proceedings of the 8th ACM Conference on Recommender Systems - RecSys ’14. doi:10.1145/2645710.2645750

24. Loria, S. (2018). textblob Documentation. Release 0.15, 2

25. Ma, C., Kang, P., & Liu, X. (2019, July). Hierarchical gating networks for sequential recommendation. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 825-833).

26. Mathew, P., Kuriakose, B., & Hegde, V. (2016, March). Book Recommendation System through content based and collaborative filtering method. In 2016 International conference on data mining and advanced computing (SAPIENCE) (pp. 47-52). IEEE.

27. Mengting Wan, Julian McAuley, "Item Recommendation on Monotonic Behavior Chains", in RecSys'18. [bibtex]

28. Mengting Wan, Rishabh Misra, Ndapa Nakashole, Julian McAuley, "Fine-Grained Spoiler Detection from Large-Scale Review Corpora", in ACL'19. [bibtex]

29. Mohamed, M. H., Khafagy, M. H., & Ibrahim, M. H. (2019, February). Recommender systems challenges and solutions survey. In 2019 International Conference on Innovative Trends in Computer Engineering (ITCE) (pp. 149-155). IEEE

30. Nalmpantis O. and Tjortjis C., 'The 50/50 Recommender: a Method Incorporating Personality into Movie Recommender Systems', CCIS 'Communications in Computer and Information Science, pp. 498-507, 2017, Springer-Verlag

31. Rana, C., & Jain, S. K. (2012). Building a Book Recommender system using time-based content filtering. WSEAS Transactions on Computers, 11(2), 2224-2872

32. Resnick P, Varian HR. Recommender systems. Commun ACM 1997;40(3):56–8. http://dx.doi.org/10.1145/245108.24512

33. Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2002, December). Incremental singular value decomposition algorithms for highly scalable recommender systems. In Fifth international conference on computer and information science (Vol. 1, No. 012002, pp. 27-8)

34. Satyanarayana, K. (2011). A STUDY OF RECOMMENDER SYSTEM ON DIFFERENT PERSPECTIVES

35. Schafer, J.B., Frankowski, D., Herlocker, J., Sen, S.: Collaborative filtering recommender systems. In: The Adaptive Web, pp. 291–324. Springer Berlin / Heidelberg (2007)

36. Schoinas, I., & Tjortjis, C. (2019, May). MuSIF: a product recommendation system based on multi-source implicit feedback. In IFIP International Conference on Artificial Intelligence Applications and Innovations (pp. 660-672). Springer, Cham.

37. Sivapalan, S., Sadeghian, A., Rahnama, H., & Madni, A. M. (2014, August). Recommender systems in e-commerce. In 2014 World Automation Congress (WAC) (pp. 179-184). IEEE

38. Smith, B., & Linden, G. (2017). Two decades of recommender systems at Amazon. com. Ieee internet computing, 21(3), 12-18

39. Sodera, N., & Kumar, A. (2017). Open problems in recommender systems diversity. 2017 International Conference on Computing, Communication and Automation (ICCCA). doi:10.1109/ccaa.2017.8229776

40. Subramaniyaswamy, V., Logesh, R., Chandrashekhar, M., Challa, A., & Vijayakumar, V. (2017). A personalised movie recommendation system based on collaborative filtering. International Journal of High Performance Computing and Networking, 10(1/2), 54. doi:10.1504/ijhpcn.2017.083199

41. Wei, K., Huang, J., & Fu, S. (2007, June). A survey of e-commerce recommender systems. In 2007 international conference on service systems and service management (pp. 1-5). IEEE.

42. Zhou, R., Khemmarat, S., & Gao, L. (2010, November). The impact of YouTube recommendation system on video views. In Proceedings of the 10th ACM SIGCOMM conference on Internet measurement (pp. 404-410)