UNIVERSIDADE FEDERAL DA BAHIA
INSTITUTO DE MATEMÁTICA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

IMPROVING RECOMMENDER SYSTEMS PRECISION WITH MULTIPLE METADATA USING ENSEMBLE METHODS

Bruno Souza Cabral

DISSERTAÇÃO DE MESTRADO

Salvador
December/2015

UNIVERSIDADE FEDERAL DA BAHIA
INSTITUTO DE MATEMÁTICA

Bruno Souza Cabral

IMPROVING RECOMMENDER SYSTEMS PRECISION WITH MULTIPLE METADATA USING ENSEMBLE METHODS

Trabalho apresentado ao PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO do INSTITUTO DE MATEMÁTICA da UNIVERSIDADE FEDERAL DA BAHIA como requisito parcial para obtenção do grau de Mestre em CIÊNCIA DA COMPUTAÇÃO.

Orientador: Frederico Araujo Durão
Co-orientador: Marcelo Garcia Manzato

Salvador
December/2015

Ficha catalográfica.

Cabral, Bruno Souza

Improving Recommender Systems Precision with Multiple Metadata using Ensemble Methods/ Bruno Souza Cabral– Salvador, December/2015.

79p.: il.

Orientador: Frederico Araujo Durão.
Co-orientador: Marcelo Garcia Manzato.
Qualificação de Mestrado– UNIVERSIDADE FEDERAL DA BAHIA, INSTITUTO DE MATEMÁTICA, December/2015.

TOPICOS PARA FICHA CATALOGRAFICA.
I. Durão, Frederico Araujo. II. Manzato, Marcelo Garcia.
III. UNIVERSIDADE FEDERAL DA BAHIA. INSTITUTO DE MATEMÁTICA. IV. Título.

NUMERO CDD

ACKNOWLEDGEMENTS

I would like to thank my beautiful family and friends. Especially my Dad, Charles, who always advised me and took care of all boring stuff in my life, so I could have time to concentrate; My mom, Eneida, who is a super mother, always worked insanely long periods and still have time to take care, cook and groom us; My sister, Debora, who had to handle my hugs and rants every half hour during my academic life. I truly love them and appreciate all the support they have given me. I would also like to express my love and gratitude for my girlfriend, Bruna, for having an endless patience, and offering me a warm shoulder during my entire journey. I would like to thanks Marivaldo and Gabriel, my friends and partners at Potelo, who handled my share of working during my Masters. Finally, I would like to thanks my advisors: Frederico Durão, who is an amazing advisor and teacher, and during the way, he was very supportive, participative and keep incentivizing me to keep going; and Marcelo Manzato, who was a great mentor, who introduced me to this area and during this work guided me to the right path.

RESUMO

Sistemas de Recomendação tornaram-se populares e amplamente adotados por muitos sites e serviços, sendo ferramentas importantes para ajudar os usuários a filtrar o que é relevante para eles neste mundo com tamanha quantidade de informação. Há diversas maneiras de construir Sistemas de Recomendação, como a filtragem baseada em conteúdo, que recomenda itens para o usuário com base em um perfil que contém informações a respeito do conteúdo, tais como gênero, palavras-chave, assunto, etc. Estes metadados são ponderados de acordo com avaliações anteriores, a fim de caracterizar principais interesses do usuário. No entanto, esta abordagem tem problemas, tais como excesso de especialização e desempenho limitado devido à escassez de metadados ou à qualidade. Uma alternativa é a filtragem colaborativa, baseado em clusters de usuários ou itens semelhantes. Uma desvantagem da filtragem colaborativa é o esforço computacional gasto para calcular similaridade entre usuários e/ou itens em um espaço vectorial composto por avaliações do usuário. Sistemas híbridos surgiram para combinar os benefícios de ambas as técnicas. No entanto, a maioria dos sistemas recentes não consideram todos os metadados associados ao conteúdo, o que poderia fornecer informações significativas sobre os interesses do usuário. Esse trabalho propõe uma série de estratégias, como ensembles, para combinação de múltiplos metadados, com o objetivo de melhorar a performance dos atuais algoritmos de recomendação de uma forma computacionalmente viável. Quatro experimentos foram realizados utilizando conjuntos de dados em algoritmos no estado -da-arte e os resultados indicam que os algoritmos propostos alcançaram uma melhoria considerável do MAP de até 21 % quando usando os algoritmos do conjunto. Estes resultados encorajadores indicam que os algoritmos propostos podem ser usados para melhorar os Sistemas de Recomendação com múltiplos metadados.

Palavras-chave: recomendação; ensemble; metadados; filme; filtragem colaborativa

ABSTRACT

Recommender systems have become increasingly popular and widely adopted by many sites and services. They are important tools in assisting users to filter what is relevant for them in this complex information world. There are a number of ways to build recommender systems such as content-based filtering, which recommends multimedia content to the user based on a profile containing information regarding the content, such as genre, keywords, subject, etc. These metadata are weighted according to past ratings, in order to characterize the user's main interests. However, this approach has problems such as over-specialization and limited performance due to metadata scarcity or quality. An alternative to this problem is the collaborative filtering approach, which is based on clusters of similar users or items. The drawback is the computational effort spent to calculate similarity between users and/or items in a vectorial space composed of user ratings in a user-item matrix. Alternatively, hybrid recommenders aim at grouping the benefits of content based and collaborative filtering approaches. The downside of hybrid recommenders which primarily exploit latent factor models are i) do not consider the metadata associated to the content, which could provide significant and meaningful information about the user's interests, and ii) usually process only one item attribute missing the exploitation of combination of the metadata available. This dissertation investigates the problem of using and combining multiple metadata in hybrid Recommender Systems, characterizing it and proposing ensemble strategies to combine different metadata. This work aims at improving the top-performing state-of-art algorithms to leverage the available item metadata with an ensemble of this information in a computationally feasible way. Four experiments were performed using state-of-art datasets and algorithms and the results indicate that we were able to archive a considerable MAP improvement of up to 21% when using the ensemble algorithms. These encouraging results indicate that ensemble algorithms can be used to enhance the recommenders algorithms with multiple metadata.

Keywords: recommendation; ensemble; metadata; movie; collaborative filtering

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

Drop by drop is the water pot filled. Likewise, the wise man, gathering it little by little, fills himself with good.

—BUDDHA (Dhammapada)

The Information Age is characterized by the widespread proliferation of emerging information and communication technologies and the capabilities that those technologies provide to overcome the barriers imposed on communications by time, distance, and location. This new age represents the swift from the traditional industry to an economy based on information and, as a result, we never had been exposed to such a staggering amount of information. According to Gantz and Reinsel (2012), from 2005 to 2020, the information in the digital universe will grow by a factor of 300, from 130 exabyte to 40,000 exabytes, or 40 trillion gigabytes (more than 5,200 gigabytes for every man, woman, and child in 2020). Accordingly, from now until 2020, the digital universe information will double every two years, as illustrated in Figure 1.1. This leads to a problem: there is simply too much information to process and to choose. It is simply not possible to grasp even a small percentage of it in a single lifetime.

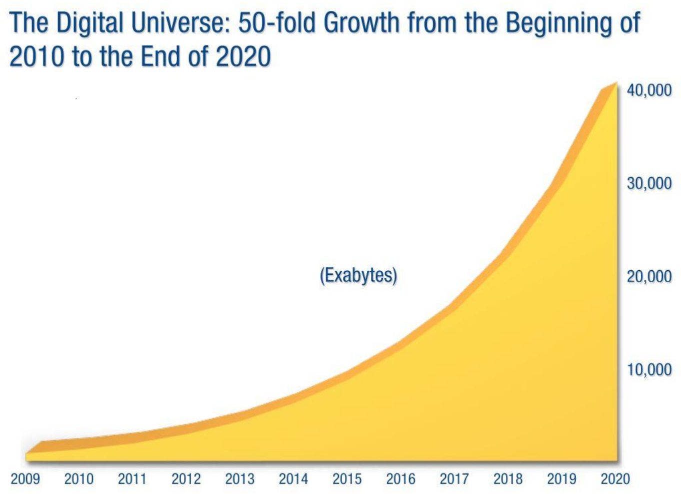


Figure 1.1: Information growth (GANTZ; REINSEL, 2012).

Processing huge amount of information is not a new phenomenon. Our brains are already adapted to cope with a huge amount of data received every second through our senses, and have a very good capacity of filtering and processing signals, images and messages. For some type of signals such as visuals, the brain has a phenomenal ability to minimize the noise and receive only the relevant patterns (MUSTO, 2010). Recent studies (OTT, 2010) showed that human brain is able to absorb 126 bits of information per second from the 393 bits per second we interact every day, totaling 34 billion of bits. However, it is expected that until 2020 the amount of information will increase by almost 30 times. The problem is that nowadays the information is presented in a format that our brain is not prepared and needs to be assisted (OTT, 2010). This is known as *Information Overload*, which is the sensation of fatigue and distress that follows the cognitive surplus required to handle the volume of information we have to deal with everyday (MUSTO, 2010). It is considered as the cause for diseases such attention deficits, anxiety and cybercondria (HALLOWELL, 2005).

As seen, the amount of information we are exposed every day presents itself as a challenge to find relevant and useful information. However, we cannot blame on the abundance of information but rather on the absence of appropriate filters that support our physiological brain deficits and help us to select the most important pieces of information. The key is filtering the information (MUSTO, 2010).

To mitigate this problem, Recommender Systems (RS) appears as a response to the information overload problem by learning from users about their interests from past user actions (ratings, votes, ranked lists, mouse clicks, page views, product purchases, etc.) and suggesting products that are likely to fit their needs (GANTNER, 2012). Organizing and filtering this information can also present as a good opportunity to improve conversion rates in e-commerce, engage customers with your product and improving content discoverability for business (ADOMAVICIUS; TUZHILIN, 2005). Thanks to this, Recommender Systems are increasingly present in a wide variety of areas such as e-commerce, entertainment, education and tourism, making it a very promising area of research. Examples of RS used commercially include Youtube[1], by recommending related videos, Google[2], with personalized searches according to the user interest, Submarino[3], recommending related products, Facebook[4] that recommends friends you might know and Netflix[5], that recommends movies you might enjoy.

To provide recommendations that respect the user interest it is necessary to capture users' preferences and compare with information describing the item to be recommended. However, the available algorithms do not always use all the available information, presenting as an opportunity to improve the Recommendation performance.

The remaining of chapter contextualizes the focus of this dissertation and starts by presenting its motivation in Section 1.1 and a clear definition of the problem in Section 1.2. Section 1.3 presents the main contributions, while Section 1.4 describes the methodology used by this work. Finally, Section 1.5 outlines the structure of this dissertation.

1.1 MOTIVATION

Recommender systems have become increasingly popular and widely adopted by many sites and services. They are important tools in assisting users to filter what is relevant for them in this complex information world. There are a number of ways to build recommender systems and they can be used to predict a rating or to generate a top-n ranking of recommended items. Today, a large portion of the work on recommender systems is based on top-n recommendation or rating prediction. The top-n recommendation requires bi/unary interaction data between users and items, whereas rating prediction requires a dataset with previous ratings (SAID, 2013).

Previously, rating prediction – How much will a user like/rate a given item? – had the majority of research and attention (GANTNER, 2012). Nowadays the task of item recommendation – Which items will a user like/buy? – takes the bulk of attention and it is considered more relevant for practical recommender system applications (GANTNER, 2012). For both tasks, you have three different approaches: content-based filtering, collaborative filtering and the combination of both of them (ADOMAVICIUS; TUZHILIN, 2005; EKSTRAND; RIEDL; KONSTAN, 2011).

Content-based filtering recommends content to the user based on a profile containing information regarding the content, such as genre, keywords, subject, etc. These metadata are weighted according to past ratings, in order to characterize the user's main interests. However, this approach has problems such as over-specialization (ADOMAVICIUS; TUZHILIN, 2005), limited performance due to metadata scarcity or quality and requires knowledge of the domain. An alternative to this problem is the collaborative filtering, which is based on the premise that users who have similar preferences in the past are likely to have similar preferences in the future. However, collaborative filtering

[1]http://www.youtube.com
[2]http://www.google.com
[3]http://www.submarino.com.br
[4]http://www.facebook.com
[5]http://www.netflix.com

alone has some limitations such as problems when the data is too sparse, overfitting and it is not able to recommend new items or to new users.

Considering the limitations and challenges depicted above, one solution is the use of Hybrid Systems, which use collaborative filtering with the addition of information describing the contents of the items. This information is called metadata. For example, a film can be represented by the title, actors, release date, genre, etc. With more information about the user, greater is the chance that a movie recommendation will be accurate. It is known that limitations of content based and collaborative filtering, such as the cold start problem, over-specialization and limited content analysis, can be reduced when combining both strategies into a unified model (ADOMAVICIUS; TUZHILIN, 2005).

However, many recent systems which exploit latent factor models do not consider the metadata associated to the content, that could provide significant and meaningful information about the user's interests, and some metadata aware recommenders only supports one item attribute. An alternative approach for handling multiple metadata is with ensemble methods. An ensemble method combines the recommendations of different algorithms, or the same algorithm with different parameters to obtain a final recommendation. Ensemble methods have been successfully used in the Netflix Prize contest, consisting of the majority of the top performing solutions. (TÖSCHER; JAHRER; BELL, 2009; PIOTTE; CHABBERT, 2009). Most of the related works in the literature point out that ensemble learning has been used in recommender system as a way of combining the prediction of multiple algorithms (heterogeneous ensemble) to create a stronger rank (JAHRER; TöSCHER; LEGENSTEIN, 2010), in a technique known as *blending*. They have been also used with a single collaborative filtering algorithm (single-model or homogeneous ensemble), with methods as *Bagging* and *Boosting* (BAR et al., 2013).

Nonetheless, applying traditional ensemble techniques to metadata are often not feasible to implement in a production scenario because of the computational cost and complexity. In the case of heterogeneous ensemble, it needs to train all models in parallel and treat the ensemble as one big model, but unfortunately training 100+ models in parallel and tuning all parameters simultaneously is computationally not feasible (TÖSCHER; JAHRER; BELL, 2009). In contrast, the homogeneous ensemble demands the same model to be trained multiple times, and some methods such as Boosting requires that the underlying algorithm be modified to handle the weighted samples. The focus of this dissertation is to investigate algorithms for combining multiple metadata available to improve the Recommendation performance.

1.2 PROBLEM STATEMENT

This dissertation investigates how the actual Recommenders Systems utilizes the available items metadata, characterizing it and proposing ensemble strategies to combine multiple metadata in hybrid Recommender Systems. This work aims at improving the top-performing state-of-art algorithms to leverage the available item metadata with an ensemble of this information in a computationally feasible way.

1.3 STATEMENT OF THE CONTRIBUTIONS

As a result of the work presented in this dissertation, the following contributions can be highlighted:

- **A study on existing solutions** , which can provide the research and professional community an overview of the state-of-the-art algorithms in the field that support multiple metadata.
- **A study on the best performant metadata for movies dataset** , specifying what metadata provides the biggest increase in Recommender performance, which metadata provides the most significant value and charactering how the number of metadata and sparsity impacts on the performance. This information is valuable for researchers to further create novel tools and algorithms.
- **A set of ensemble techniques** that can be used to improve the performance of existing Recommender Systems when using multiple metadata (e.g. Movie recommendation).
- **An open-source application implementing the techniques proposed in this work** . We feel that in science the results should be replicable and freely accessible. In this way, we published a fully-featured open-source software implementing the techniques proposed.

In addition to the contributions mentioned, the work proposed in this project has already been published in the form of papers at peer-reviewed workshops and conferences. Moreover, we are currently submitting other papers to report the remaining results.

1.4 METHODOLOGY

The methodology of this project was based on a multi-method approach, which consists of a combination of primary studies (proposed solutions, applications development, experiments, etc.) and secondary studies (literature review, mapping studies, etc.) to increase the body of knowledge in a particular area based on the findings of such research. Followed by the specification and development of a tool implement the techniques proposed by this dissertation.

The activities realized by this work were:

- **Literature review:** in this activity was be identified all relevant work carried out in Recommender Systems in order to collect information necessary for the dissertation development. It was necessary theoretical background on the following topics: recommender systems, matrix factorization, collaborative filtering, ensemble methods and recommender systems evaluation. Including the study of its features, algorithms, data structures, involved concepts and aspects of the related technology.
- **Specification and design of the tool architecture:** oriented by the requirements and opportunities identified in the literature review, several of ensemble

strategies were proposed. Based on those requirements of the strategies, an architecture was designed to fulfill their needs.

- **Feasibility study:** to verify shortcomings and benefits of the proposed architecture and strategies;
- **Development of strategies and tool:** The development and codification of strategies and tool to achieve the proposed objectives. It included adapting existing algorithms to make use of metadata, develop several ensemble strategies, and the combination of various metadata using the proposed ensemble strategies.
- **Experiments and Evaluation:** experiments was be conducted to validate the effectiveness of the proposed strategies in quantitative terms. For each proposed approach, multiple evaluation methods will be applied to the developed technique. The technical performance was evaluated according to their efficiency to return items according to the user's needs. We used a number of metrics that are available in the literature in our evaluation of performance, such as precision, recall, MAP and AUC.
- **Dissertation writing:** after the completion of the validation and evaluation studies, the entire body of knowledge gained in this study was documented.

1.5 DISSERTATION STRUCTURE

The remainder of this dissertation is organized as follows:

- **Chapter 2** reviews the essential topics used throughout this work: Recommender systems and ensemble techniques. It also contains a comprehensive revision of Recommendation Models.
- **Chapter 3** describe the Movie Recommendation problem, our motivation and what is the problem this work is trying to solve. Additionally, it has a overview about related works, citing important papers in industry and literature, and an detailed description of ensemble techniques.
- **Chapter 4** describes the proposed solution. We introduce four ensemble strategies to combine multi-model interactions.
- **Chapter 5** describes some experiments conducted to evaluate the proposed solution.
- **Chapter 6** provides the concluding remarks. It discusses our contributions, limitations, threats to validity, and outline directions for future work.

Chapter 2

RECOMMENDER SYSTEMS

Don't let schooling interfere with your education.

—MARK TWAIN

This chapter provides an overview of the Recommender Systems field. After introducing the Recommendation problem, it is explored the most prominent prediction tasks for recommender systems, describing different approaches for accomplishing these tasks, and introduce the concept of ensemble to combine multiple results. Finally, we discuss some state-of-art Hybrid Recommender Systems that are used through this work.

2.1 THE RECOMMENDATION PROBLEM

The concept of Recommender Systems was introduced (RESNICK; VARIAN, 1997) to formally define tools and techniques able to provide personalized information access to large collections of structured and unstructured data, providing users with advices about items they might be interested in. Those items represent things to be recommended, such as pages on the web, news, articles, jokes, movies, products of any kind, music albums, individual songs, etc.

Later, several others definitions of what a RS is have been proposed in literature. For example, Burke (BURKE, 2002) says that recommender systems have the effect of guiding the user in a personalized way to interesting or useful objects in a large space of possible. Ganter (GANTNER, 2012), define Recommender System as information systems that learn user preferences from past user actions (ratings, votes, ranked lists, mouse clicks, page views, product purchases, etc.) and suggest items according to those user preferences.

A formal definition to the recommendation problem can be formulated as follows (ADOMAVICIUS; TUZHILIN, 2005; MUSTO, 2010):

Let U be the set of all users and let I be the set of all possible items that can be recommended to this user. Each element of the user space U can be defined as a profile that can include various user features, such as gender, age or city. Likewise, each element of the item space I is defined through a set of features. For example, in a movie recommendation scenario each movie can have as features its genre, director and year of production. The size of the sets I and U can be very large in a real use case.

Let f be a utility function that measures the usefulness of item i to user u, $f : UI \rightarrow R$, where R is a totally ordered set (nonnegative integers or real numbers within a certain range). The recommendation problem consists in choosing such item $i'_u \in I$ that maximizes user's utility for each user $u \in U$.
More formally:

$$\forall u \in U, i'_u = argmax_{i \in I} f(u, i) \tag{2.1}$$

Usually the utility function represented by a rating, which indicates how a particular user liked a specific item. The central problem of recommender systems lies in that utility is usually not defined on the whole U x I space, but initially defined only on the items previously rated by the users. For example, in a movie scenario, users initially rate some subset of movies they have already seen. Therefore, the recommendation engine should be able to predict the ratings of the non-rated movie/user combinations and issue appropriate recommendations based on these predictions (MUSTO, 2010).

With rate prediction, recommendations of an item to a user are made by selecting the highest rating among all the estimated ratings for that user (best recommendation problem), or select the top N ratings (Top-N recommendation problem) (ADOMAVICIUS; TUZHILIN, 2005).

2.2 THE RECOMMENDATION PIPELINE

There are multiple approaches to generate recommendations. Musto (MUSTO, 2010) came up with a very thorough and generic Pipeline that can represent the steps used in a recommendation pipeline. It was enhanced to also consider from the items point of view, and consists of the following steps:

1. **Training:** first, the recommender needs to gather information about the target users and the items that will be recommended. For items it maybe be extracted a vector of features composed of information describing them. For the users the system may also collect information about what one knows and likes, demographical or contextual information. This step could be accomplished in an explicit or implicit way. In the first case the user explicitly expresses her preferences, by giving a rating items or marking as positive, while in the latter user's preferences are gathered by analyzing their transactional or behavioral data (for example, clicking a link or reading a news article could be considered as a clue of user interest in that item).

2. **User Modeling:** to personalization effectively happen, it implies the presence of something describing and identifying the preferences of the user interacting with the recommender. Therefore, the information extracted are modeled and stored in a user profile. Modeling the user profile is the core of the pipeline since it is the component that triggers the whole recommendation process (MUSTO, 2010). Depending on the filtering model, different information's are stored about the user.

For a pure collaborative filtering, only past actions are recorded, while in a content-based recommender architecture the component for User Modeling can be split in a Content Analyzer, whose goal is to analyze content in order to pop up relevant concepts from unstructured text, and a Profile Learner that stores these concepts into user profiles (RICCI; ROKACH; SHAPIRA, 2011).

3. **Filtering:** lastly, based on the user profile the recommender system will filter the information and recommend it to the user. It can be based on a profile, predicting how the user would rate an unknown item (Rating Prediction) (GANTNER, 2012) or rank the items according to a relevance criterion, providing the user with an ordered list of the most relevant items (MUSTO, 2010).

The main differences among different recommendation approaches lie in the way profiles are built and how it recommend items. In next section those different approaches will be detailed.

2.3 RECOMMENDATION APPROACHES

There are multiple recommendation approaches, and authors in literature classify them in different classes. Burke (BURKE, 2007) split recommendation models into six classes (Content-based; Collaborative; Demographic; Knowledge-based; Community-based and Hybrid). Masthoff (MASTHOFF, 2011) divides in two broad categories: content-based and collaborative filtering approaches. In this work we use Masthoff definition with the addition of a third category, Hybrids, a combination of content-based and collaborative filtering.

In summary, with content-based recommenders the concept of similarity identifies the items that share common features with those the user already considered as relevant or interesting in the past. On the other hand, collaborative filtering tries to bring out hidden connections between users that belong to the same community and share similar tastes. In the next section, a description of the content-based approach and an analysis its strengths and weaknesses will be provided.

2.3.1 Content-based Recommender Systems

Content-based Recommenders generates recommendations by analyzing a set of descriptions of items previously rated or viewed by a user, and building a profile of user interests based on the features of the objects rated by that user, as can be seen on Figure 2.1

The profile is an organized representation of user interests built based on previous items observed by him. The recommendation of new interesting items consists in matching up the attributes of the user profile against the attributes of a content object (RICCI; ROKACH; SHAPIRA, 2011). In a movie recommendation scenario, for example, a user profile may store the preferred genres and directors. The movie would also be composed of a set of those features, as illustrated on Figure 2.2. With this information, the recommendation step consists in matching up the features of an unseen item with a user profile.

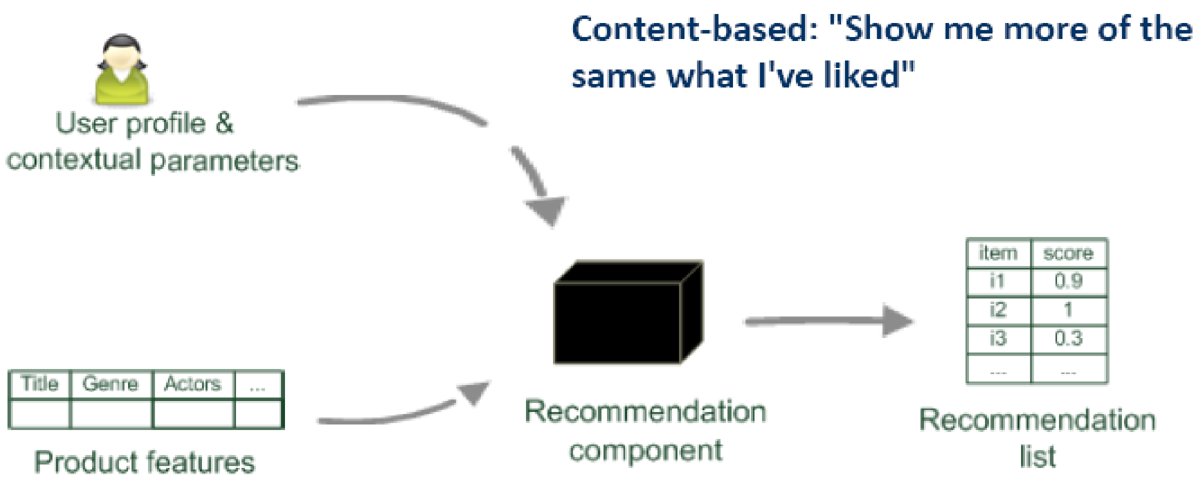
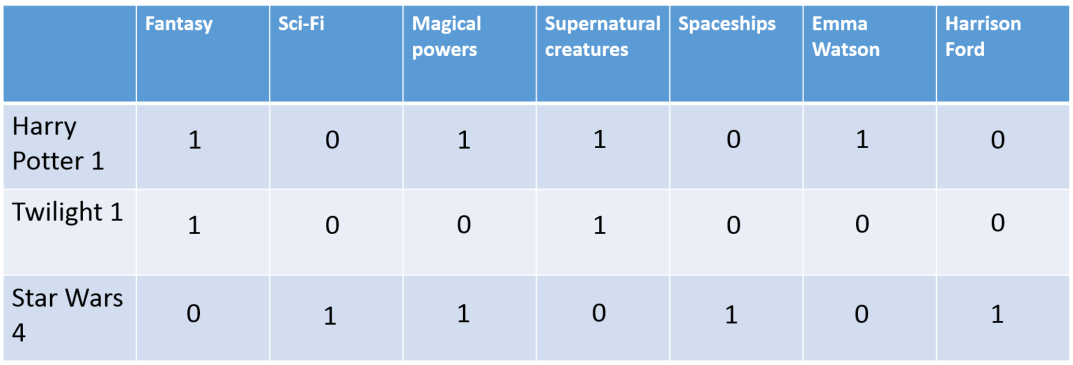

Content-based: "Show me more of the same what I've liked"
User profile & contextual parameters
Title | Genre | Actors | ...
Product features
Recommendation component
item | score
i1 | 0.9
i2 | 1
i3	0.3
Recommendation list

Figure 2.1: Content-based recommender overview (SEAGATE, 2015).



	Fantasy	Sci-Fi	Magical powers	Supernatural creatures	Spaceships	Emma Watson	Harrison Ford
Harry Potter 1	1	0	1	1	0	1	0
Twilight 1	1	0	0	1	0	0	0
Star Wars 4	0	1	1	0	1	0	1

Figure 2.2: Example of item features.

Ricci (RICCI; ROKACH; SHAPIRA, 2011) proposes a high level architecture of a content- based recommender system, where each step of the Figure 2.1 is extracted by a separate component:

- **Content Analyzer -** It responsible for extracting the vector of features that describe the item. Usually the item has textual information, and a pre-processing step is needed to extract structured relevant information. The information is analyzed by feature extraction techniques in order to shift item representation from the original information space to the target one (RICCI; ROKACH; SHAPIRA, 2011). For example, in a movie recommendation scenario, if the information about the movie is textual, it needs to use Natural Language Processing (NLP) techniques to extract information such as director, genre and so. This representation is the input to the Profile Learner and Filtering component;

- **Profile Learner -** This component collects as much as possible information about user preferences or tastes based on previous observed items and tries to generalize this data, in order to build a user profile. Usually, the generalization strategy is realized through machine learning techniques (RICCI; ROKACH; SHAPIRA, 2011), which are able to infer a model of user interests starting from items liked or disliked in the past. In a movie recommendation scenario, a simple user profile could be composed of multiple movie genres, such as horror, comedy, and how much the user likes it, from a scale 0 to 1.

- **Filtering -** This component suggest relevant items by matching the similarity between the user profile against available items. The result is a binary or a ranked list of potentially interesting items (RICCI; ROKACH; SHAPIRA, 2011).

  The matching can be determined using two different techniques: heuristic-based or model-based (MUSTO, 2010). The former calculate the score using information retrieval methods, such as cosine similarity. The latter predict the relevance relying on models learned from the underlying data through statistical or machine learning-based techniques.

2.3.1.1 Limitations of Content-based Recommender Systems .

Content-based recommender systems have several weak points:

1. **Limited content analysis.** Content-based recommendations need to have a very rich amount of features describing the items to be recommended to be effective (RICCI; ROKACH; SHAPIRA, 2011). However, in many cases, those extraction requirements are very difficult to fulfill. There are some domains where automatic feature extraction is complicated (MUSTO, 2010)(such as graphical images, video streams, and audio streams), and extracting content from Natural Language is a complicated problem (BANGALORE; RAMBOW, 2000). Finally, assigning features by hand is often not practical.

2. **Over-specialization.** Content-based recommendation system retrieves items that match against a specific user profile. In this way, it cannot recommend items that are different from anything the user has seen before, and this is not a desirable characteristic, as it will always recommend some of the same with a limited degree of novelty. This is known as the serendipity problem. A desirable goal is that the recommender system increase the serendipity of the recommendation lists by including “unexpected” items in which the user might be interested in (MUSTO, 2010).

3. **Cold-start.** When a new user is added to the system, it does not have a previous history of items. Therefore, his user profile will be poor and the recommender system will not be able to understand user preferences and provide accurate recommendations.

2.3.2 Collaborative Filtering Recommender Systems

Another way of recommending is using a Collaborative Filtering approach. It is considered to be the most popular and widely implemented technique in RS (RICCI; ROKACH; SHAPIRA, 2011). Differently from content-based approaches, Collaborative Filtering do not use the item content directly, but allows users to give ratings or observations about the items in such a way that when enough information is stored on the system, it can make recommendations user based on information provided by what those users consider to have the most in common with them (ORTEGA et al., 2013).

This approach overcomes some of the limitations of content-based (RICCI; ROKACH; SHAPIRA, 2011). For example, items with non-existent or poor descriptions can still be recommended to users through the feedback of other users. Furthermore, collaborative recommendations are based on the quality of items as evaluated by peers, instead of relying on content that may be unreliable. Finally, unlike content-based systems, collaborative filtering can recommend items with very different content, as long as other users have already shown interest for these different items (RICCI; ROKACH; SHAPIRA, 2011).

Collaborative filtering methods can be grouped in the two general classes (RICCI; ROKACH; SHAPIRA, 2011):

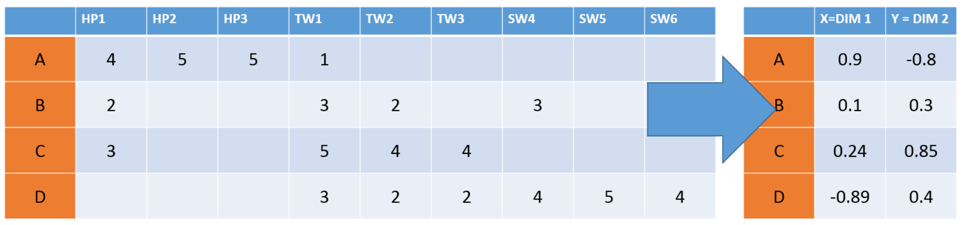
- **Memory-based:** Here, the user-item ratings stored in the system are directly used to predict ratings for new items. It can also be subdivided in two types, User-based and Item-based.

  In User-Based Collaborative Filtering, correlations are identified between users based on past preferences that are similar in order to make predictions on what each user will like in the future. If two users have rated many items similarly in the past, they may be considered in the same neighborhood (CASINELLI, 2013).

  In an Item-Based Collaborative Filtering, the item similarities are used in order to make recommendations. Rather than building a neighborhood and making recommendations based on similar users, correlations are made between items' preferences. The intuition is that the user will be recommended items that are most similar to items he has already rated in the past (CASINELLI, 2013).

- **Model-based:** Differently from neighborhood-based systems, which use the ratings matrix directly, model-based approaches use those ratings to learn a predictive model. The general idea is to model the user-item interactions with factors representing latent (or hidden) characteristics of the users and items matrix and discover general classes. This model needs to be trained firstly using the existing data, and only after training, it can be used to generate recommendations (RICCI; ROKACH; SHAPIRA, 2011).

  Matrix factorization it one of the most popular model-based method (RICCI; ROKACH; SHAPIRA, 2011). It works by decomposing the user vs item matrix of preference data into a more compact, denser representation that can be used to extrapolate the expected preference of items to a user. One of the most common techniques for this is Singular Value Decomposition (SVD). In Figure 2.3, illustrates the reduction of a 9-dimensional movie rating matrix to a more compact 2-dimensional matrix. In Figure 2.4 we made an assumption of what those two dimensions represent in the movie recommendation context.



	HP1	HP2	HP3	TW1	TW2	TW3	SW4	SW5	SW6
A	4	5	5	1					
B	2			3	2		3		
C	3			5	4	4			
D				3	2	2	4	5	4

	X=DIM 1	Y = DIM 2
A	0.9	-0.8
B	0.1	0.3
C	0.24	0.85
D	-0.89	0.4

Figure 2.3: Example Singular Value Decomposition.

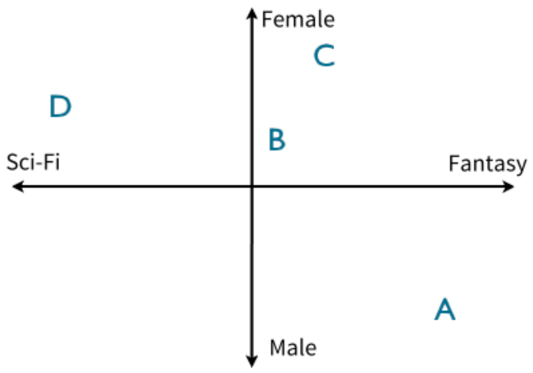


Figure 2.4: Interpretation of the reduced matrix.

2.3.2.1 Limitations of Collaborative Filtering Recommender Systems .

Collaborative Filtering address some issues compared to Content-based recommender systems, however it also has several weak points:

1. **Sparsity.** The effectiveness of a Collaborative Filtering algorithm is closely related to the availability of a dense and complete matrix of users and ratings. However, a typical issue is that the number of ratings provided by the users is often very small compared to the number of items that need to be predicted (MUSTO, 2010). In this way, the matrix user x item becomes sparse (with empty spaces). It cannot make valuable recommendations with the sparsity problem. Dimensionality reduction techniques, such as Singular Value Decomposition (SVD) alleviates this problem (RICCI; ROKACH; SHAPIRA, 2011).

2. **Cold Start.** When a new item is added in a recommender system there is no way to recommend it before more users rate those items. You can address this problem by encouraging users to vote in novel items in order to trigger the similarity calculations that feed CF algorithms (MUSTO, 2010). Similarly, when a new user is added to

the system, we do not have any information to determine his preferences. Usually this is address by recommending popular items firstly (STECK, 2011).

2.3.3 Hybrid recommender systems

Hybrid recommender systems combine two or more different recommender algorithms to create a stronger recommender. A hybrid recommender could for example, alleviate the cold-start problem in Collaborative Filtering by using the item description to match the new item with existing items based on metadata, when the users did not rate the item.

Burke (BURKE, 2002), classified hybrid recommender systems into seven classes:

- **Weighted** recommenders take the scores produced by several recommenders and combine them to generate a recommendation list (or prediction) for the user.
- **Switching** recommenders switch between different algorithms and use the algorithm expected to have the best result in a particular context.
- **Mixed** recommenders present the results of several recommenders together. This is similar to weighting, but the results are not necessarily combined into a single list.
- **Feature-combining** recommenders use multiple recommendation data sources as inputs to a single meta-recommender algorithm.
- **Cascading** recommenders chain the output of one algorithm into the input of another.
- **Feature-augmenting** recommenders use the output of one algorithm as one of the input features for another.
- **Meta-level** recommenders train a model using one algorithm and use that model as input to another algorithm.

Hybrid recommenders proved to be quite powerful in the Netflix Prize contest consisting of the majority of the top performing solutions. (TÖSCHER; JAHRER; BELL, 2009; PIOTTE; CHABBERT, 2009).

Considering the limitations and challenges depicted in previous sections, hybrid recommenders play an important role because they group together the benefits of content based and collaborative filtering. It is known that limitations of both approaches, such as the cold start problem, overspecialization and limited content analysis, can be reduced when combining both strategies into a unified model (ADOMAVICIUS; TUZHILIN, 2005).

2.4 PERSONALIZED RECOMMENDATION MODELS

Recommender Systems are often used for two tasks. Rating prediction or item recommendation. While rating prediction (How much a user will rate an unknown item?) was very popular in the recommender systems literature in the past, the task of item recommendation (Which items will a user like?) got much more popular lately.

In the next subsections, we present a set of metadata aware algorithms, which use the Bayesian Personalized Ranking (BPR) framework (GANTNER et al., 2010) to personalize a ranking of items using only implicit feedback, they are considered state-of-art for the top-n recommendation problem, and were used through this project.

2.4.1 Notation

Following the same notation in (KOREN, 2010; MANZATO, 2013), we use special indexing letters to distinguish users, items and attributes: a user is indicated as u, an item is referred as i, j, k and an item's attribute as g. The notation r_{ui} is used to refer to explicit or implicit feedback from a user u to an item i. In the first case, it is an integer provided by the user indicating how much he liked the content; in the second, it is just a boolean indicating whether the user consumed or visited the content or not. The prediction of the system about the preference of user u to item i is represented by $\hat{r}_{ui}$, which is a floating point value calculated by the recommender algorithm. The set of pairs (u, i) for which r_{ui} is known is represented by the set $K = \{(u, i) | r_{ui} \text{ is known}\}$.

Additional sets used in this section are: $N(u)$ to indicate the set of items for which user u provided an implicit feedback, and $\bar{N}(u)$ to indicate the set of items that is unknown to user u.

2.5 BAYESIAN PERSONALIZED RANKING

Bayesian Personalized Ranking (BPR) is a generic framework for optimizing different kinds of models based on training data containing only implicit feedback information. It was proposed by Rendle et al. (RENDLE et al., 2009) to address the issue that happens when training an item recommendation model using implicit feedback based only on positive/negative data. The model will be fitted to provide positive scores to the observed items, while considering unvisited items as negative. However, such assumption is inaccurate because a not observed item may be due to the fact it was unknown to the user.

Considering this problem, instead of training the model using only the user-item pairs, Rendle et al. proposed considering the relative order between a pair of items, according to the user's preferences. It is inferred that if an item i has been viewed by user u and j has not ($i \in N(u)$ and $j \in \bar{N}(u)$), then $i >_u j$, which means that he prefers i over j. Figure 2.5 presents an example of this method.

The key idea is to consider entity pairs instead of single entities in its loss function, allowing the interpretation of positive-only data as partial ranking data. The user-item preference estimation is based on a Bayesian analysis using the likelihood function for

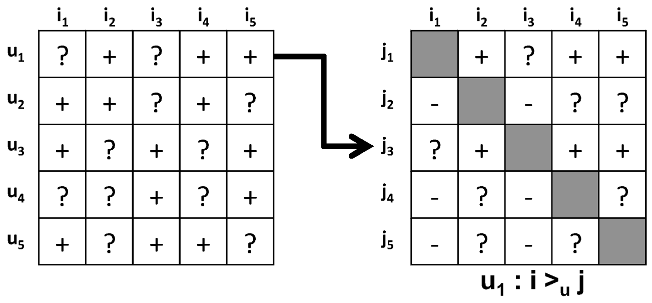


Figure 2.5: Adapted from Rendle et al., the left-hand side table represents the observed data K. On right-hand side, after applying a user-specific pairwise relation $i >_u j$, the plus signal indicates that user u has more interest in item i than j; the minus signal indicates he prefers item j over i; and the interrogation mark indicates that no conclusion can be inferred between the items.

$p(i >_u j|\Theta)$ and the prior probability for the model parameter $p(\Theta)$. The final optimization criterion, BPR-Opt, is defined as:

$$\text{BPR-Opt} := \sum_{(u,i,j)\in D_K} \ln \sigma(\hat{s}_{uij}) - \Lambda_\Theta ||\Theta||^2 \quad , \tag{2.2}$$

where $\hat{s}_{uij} := \hat{r}_{ui} - \hat{r}_{uj}$ and $D_K = \{(u,i,j)|i \in N(u) \ \& \ j \in \bar{N}(u)\}$. The symbol Θ represents the parameters of the model, Λ_Θ is a regularization constant, and σ is the logistic function, defined as: $\sigma(x) = 1/(1+e^{-x})$.

For learning the model, the authors use a variation of the stochastic gradient descent technique, denominated LearnBPR, which randomly samples from D_K to adjust Θ. Algorithm 1 shows an overview of the algorithm, where α is the learning rate.

```
Input: $D_K$
Output: Learned parameters $\Theta$
Initialize $\Theta$ with random values
for count = 1,...,#Iter do
    draw $(u,i,j)$ from $D_K$
    $\hat{s}_{uij} \leftarrow \hat{r}_{ui} - \hat{r}_{uj}$
    $\Theta \leftarrow \Theta + \alpha \left( \frac{e^{-\hat{s}_{uij}}}{1+e^{-\hat{s}_{uij}}} \cdot \frac{\partial}{\partial\Theta} \hat{s}_{uij} - \Lambda_\Theta \Theta \right)$
end
```

Algorithm 1: Learning through LearnBPR.

The BPR framework can be used with different prediction rules, where the involved parameters generate the set Θ which will be learned according to Algorithm 1. In the next three subsections, we present a set of metadata aware algorithms which use the BPR framework to personalize a ranking of items using only implicit feedback. These techniques will be considered in our evaluation in the context of movies recommendation.

2.5.1 BPR-Linear

The BPR-Linear (GANTNER et al., 2010) is an algorithm based on the Bayesian Personalized Ranking (BPR) framework, which uses item attributes in a linear mapping for score estimation. The prediction rule is defined as:

$$\hat{r}_{ui} = \phi_f(\vec{a}_i) = \sum_{g=1}^{n} w_{ug} a_{ig} \quad , \tag{2.3}$$

where $\phi_f : \mathbb{R}^n \to \mathbb{R}$ is a function that maps the item attributes to the general preferences $\hat{r}_{ui}$ and $\vec{a}_i$ is a boolean vector of size n where each element a_{ig} represents the occurrence or not of an attribute, and w_{ug} is a weight matrix learned using LearnBPR, which is variation of the stochastic gradient descent technique (GANTNER et al., 2011). This way, we first compute the relative importance between two items:

$$\begin{aligned} \hat{s}_{uij} &= \hat{r}_{ui} - \hat{r}_{uj} \\ &= \sum_{g=1}^{n} w_{ug} a_{ig} - \sum_{g=1}^{n} w_{ug} a_{jg} \\ &= \sum_{g=1}^{n} w_{ug} (a_{ig} - a_{jg}) \quad . \end{aligned} \tag{2.4}$$

Finally, the partial derivative with respect to w_{ug} is taken:

$$\frac{\partial}{\partial w_{ug}} \hat{s}_{uij} = (a_{ig} - a_{jg}) \quad , \tag{2.5}$$

which is applied to the LearnBPR Algorithm considering that $\Theta = (w_*)$ for all set of users and descriptions.

2.5.2 BPR-Mapping

The BPR-Mapping was also proposed by Gantner et al. (GANTNER et al., 2010); the key difference is that it uses the linear mapping depicted in Subsection 2.5.1 to enhance the item factors which will be later used in an extended matrix factorization prediction rule. Such an extension of matrix factorization is optimized for Bayesian Personalized Ranking (BPR-MF) (RENDLE et al., 2009) that can deal with the cold-start problem, yielding accurate and fast attribute-aware item recommendation. Gantner et al. (GANTNER et al., 2010) address the case where new users and items are added by first computing the latent feature vectors from attributes like the user's age or movie's genres, and then using those estimated latent feature vectors to compute the score from the underlying matrix factorization (MF) model.

Gantner et al.(GANTNER et al., 2010) explained that one way to learn suitable parameters for the linear mapping functions is optimizing the model for the (regularized) squared error on the latent features, and a ridge regression was used. In addition, a stochastic gradient descent was used for training because of the enormous number of input variables. Nevertheless, this approach leads to a sub-optimal performance. Thereafter, a linear mapping optimized for BPT-Opt was proposed and is what is used in BPR-Mapping.

The model considers the matrix factorization prediction rule:

$$\hat{r}_{ui} = b_{ui} + p_u^T q_i = b_{ui} + \sum_{f=1}^{k} p_{uf} q_{if} \quad , \tag{2.6}$$

where each user u is associated with a user-factors vector $p_u \in \mathbb{R}^f$, and each item i with an item-factors vector $q_i \in \mathbb{R}^f$. The baseline b_{ui} is defined as $b_{ui} = \mu + b_u + b_i$ and indicates the distinct estimates of users and items in comparison to the overall rating average μ.

From this model, the item factors are mapped according to their attributes as:

$$\hat{r}_{ui} = b_{ui} + \sum_{f=1}^{k} p_{uf} \phi_f(\vec{a}_i) \quad , \tag{2.7}$$

where $\phi_f(\vec{a}_i)$ has the same definition as in Equation 2.3.

2.5.3 MABPR

One disadvantage of the previous BPR algorithms is that they are not able to infer any conclusion when the items i and j are known (or both are unknown). In other words, if an item has been viewed by the user, it is possible to conclude that this content is preferred over all other unknown items, as it aroused a particular interest to him than the others. On the other hand, when both items are known (or both are unknown), it is not possible to infer which one is preferred over the other because the system only has the positive/negative feedback from the user. Consequently, those pairs which belong to the same class (positive or negative) will not be able to be ranked accordingly, as the model will be learned only by using the specific case where one item is known and the other is not.

To overcome this limitation, Manzato et al. (MANZATO; DOMINGUES; REZENDE, 2014) proposed an extension to the BPR technique which also considers metadata from items in order to infer the relative importance of two items.

It starts by redefining the set D_K which contains the data used during training to $D'_K := \{(u,i,j) | i \in N(u) \ \& \ j \in \bar{N}(u) \ \textbf{or} \ i \in N(u) \ \& \ j \in N(u) \cup \bar{N}(u) \ \& \ |G(i)| > 0 \ \& \ |G(j)| > 0\}$ to consider the metadata available in the specified case, while also considering items without descriptions.

Figure 2.6 shows how the proposed extension affects the relationship between items i and j with respect to the preferences of user u. Because items i_2, i_4 and i_5 are known, the system has to analyze their metadata to infer which one is preferred over the other. This is the role of function $\delta(i,j)$, which is defined as:

$$\delta(i,j) = \begin{cases} + & \text{if} \quad \varphi(u,i) > \varphi(u,j), \\ - & \text{if} \quad \varphi(u,i) < \varphi(u,j), \\ ? & \text{otherwise,} \end{cases} \tag{2.8}$$

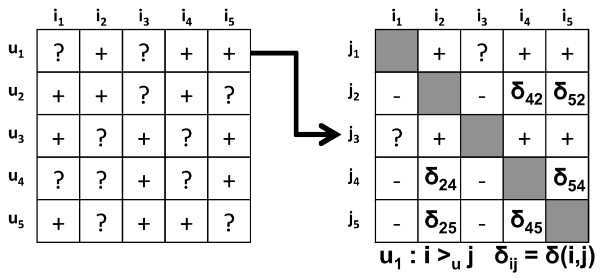


Figure 2.6: As an extension to Rendle et al. approach, Manzato et al. also consider the metadata describing items i and j when both are known ($i \in N(u)$ & $j \in N(u)$). The function $\delta(i,j)$ returns positive whether user u prefers the description of item i over the description of item j, and negative otherwise.

where $\varphi(u,.)$ is defined as:

$$\varphi(u,.) = \frac{1}{|G(.)|} \sum_{g \in G(.)} w_{ug} \quad , \tag{2.9}$$

and w_{ug} is a weight indicating how much u likes a description $g \in G(.)$.

This approach enhances the BPR algorithm with further insight about the user's preferences by considering his personal opinions about particular descriptions of items. Such metadata can be of any type: genres of movies/music, keywords, list of actors, authors, etc. The mechanism used to infer such opinions w_{ug} by analyzing only the training data is accomplished by adopting the same linear attribute-to-feature mapping described in Subsection 2.5.1.

In this section, was presented a background about Recommender Systems and detailed a generic framework for optimizing different kinds of models based on training data containing only implicit feedback information.

2.6 SUMMARY

In this chapter, we discussed about important concepts to this work. We stated by contextualizing the Recommendation problem, discussing the main concepts and its importance. Then, we provided a detailed overview about the existing Recommendation Approaches in the literature and defined the Bayesian Personalized Ranking, a popular framework for optimizing different kinds of models based on training data containing only implicit feedback information.

Next chapter presents an overview about the problem of using multiple metadata in actual recommenders, discussing the main concepts, techniques, roles, approaches and so on, in order to define the bases for this dissertation. It also presents an overview about ensembles and related works.

Chapter 3

USING MULTIPLE METADATA TO IMPROVE RECOMMENDATIONS

Success is not final, failure is not fatal: it is the courage to continue that counts.

—WINSTON CHURCHILL

In this chapter describes the Movie Recommendation problem, why it is popular in Recommender Systems, and what is the problem this work is trying to solve. Right after we make an overview about related works, citing important papers in industry and literature, and a detailed description of ensemble techniques.

This chapter is organized as follows: Section 3.1 defines the Movie Recommendation problem and the existing issue with actual recommenders algorithms, the problem of using metadata with actual Recommenders, and a analyze of relevant works in the area; and, Section 3.2 details existing ensemble algorithms in literature. Finally, Section 3.3 presents the findings and summarizes this chapter.

3.1 MOVIE RECOMMENDATION

Recommendation of Movies is a very popular area for Recommender Systems, for multiple reasons. One important reason is that many papers in the literature uses it, making easier to compare results. Another reason points to the “Netflix Prize”, a challenge started in 2007 by Netflix[1] where with a provided dataset, the participants needed to reduce the *Root Mean Squared Error* (RMSE) as much as possible. The winner won one million of dollars (TÖSCHER; JAHRER; BELL, 2009). This challenge made a boost in recommendation research where lots of quality papers with novel algorithms were published (PIOTTE; CHABBERT, 2009).

The availability of public available datasets is also a important factor. In the recommender systems literature, offline algorithmic evaluations frequently play a major role,

[1]http://www.netflix.com

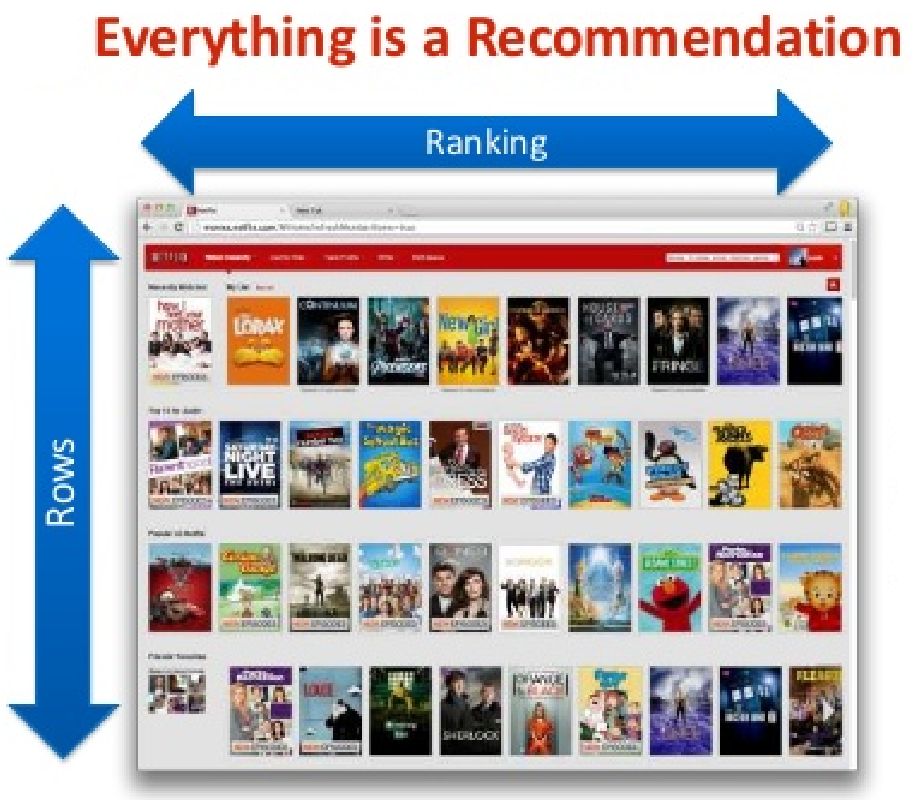


Figure 3.1: In NetFlix Everything is Recommendation.

because may be not possible to test with real users(EKSTRAND; RIEDL; KONSTAN, 2011). However, using one of the several datasets that are publicly available, we can form a body of knowledge on which the raw numeric performance of new algorithms can be compared against known performance of existing systems in a consistent environment. Those results can serve as a preliminary testing domain in building a system for which no directly relevant data is available (EKSTRAND; RIEDL; KONSTAN, 2011). In movie recommendation we have important and popular public available datasets such as MovieLens[2], NetFlix[3] and Yahoo! Movies[4].

Finally, movie recommendation is an appealing problem with immediate applications in many real world popular applications, used by millions of users everyday such as Youtube [5], NetFlix [6] and IMDB [7], which uses Recommender systems to improve users experience and content discoverability.

[2]http://grouplens.org/datasets/movielens/
[3]http://www.netflix.com
[4]http://webscope.sandbox.yahoo.com/catalog.php?datatype=r
[5]http://www.youtube.com
[6]http://www.netflix.com
[7]http://www.imdb.org

3.1.1 Metadata

Commercial movies, usually have a rich metadata. Metadata can be defined as structured information that describes, explains, locates, or otherwise makes it easier to retrieve, use, or manage an information resource. Metadata is often called data about data or information about information (GUENTHER; RADEBAUGH, 2004).

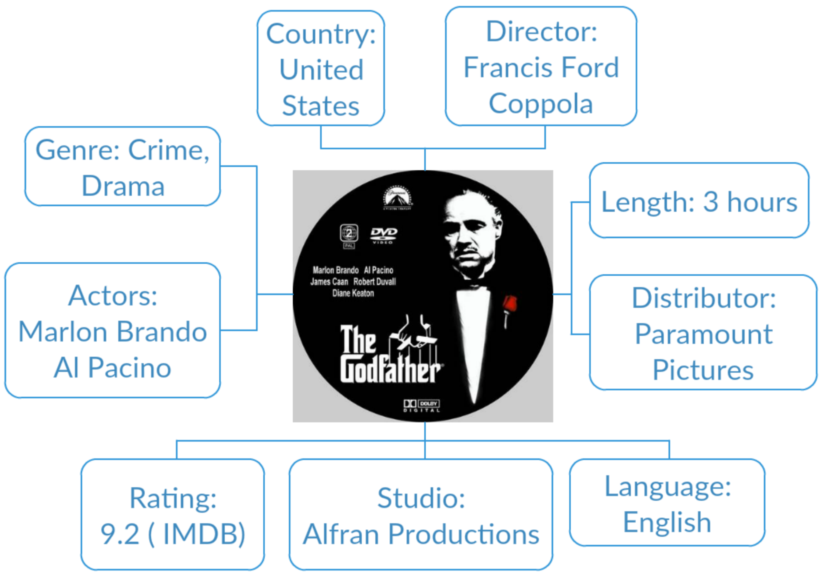

Country: United States
Director: Francis Ford Coppola
Genre: Crime, Drama
Length: 3 hours
Actors: Marlon Brando Al Pacino
Distributor: Paramount Pictures
Rating: 9.2 (IMDB)
Studio: Alfran Productions
Language: English
The Godfather
Marlon Brando Al Pacino James Caan Robert Duvall Diane Keaton

Figure 3.2: Example of possible metadata for the movie *The Godfather* (1972).

In the Figure 3.2, we can see an example of available metadata for the movie *The Godfather* (1972). In this figure, we listed only 9 metadatum (Genre; Director; Length; Distributor; Language; Studio; Rating; Actors and Country). It is possible to obtain this metadata and much more from the original media or from Web of Data sources as DBPedia[8].

3.1.2 Actual Recommender System Problem

As stated, metadata provides an important insight about the item to be Recommended. However, Recommender Systems that uses solely this metadata to recommend (Content-based) have severe shortcomings in a real world implementation, as stated in Chapter 2. One of those limitation is that the metadata is not always available, or complete enough. On the other hand, Collaborative Filtering became a hugely popular recommendation technique, prevalent for its high performance and simple requirements (KOREN, 2010). It does not use any metadata, using only historic data. However, it has shortcomings such as sparsity, overfitting and data distortion caused by imputation methods (KOREN, 2010).

Hybrid recommenders came as an answer to mitigate those problems by combining the benefits of content based and collaborative filtering. It is known that limitations of

[8]http://dbpedia.org

both approaches, such as the cold start problem, overspecialization and limited content analysis, can be reduced when combining both strategies into a unified model (ADOMAVICIUS; TUZHILIN, 2005). However, most recent systems which exploit latent factor models do not consider the metadata associated to the content, which could provide significant and meaningful information about the user's interests, and current metadata aware recommenders only supports one item attribute.

As an example, the Bayesian Personalized Ranking is one of the most used and top performant framework for the top-n problem but can only be trained with up to one item metadata. The question is, which of the nine metadatum of *The Godfather* movie should be used ? Our hypothesis in this dissertation is that if we utilize all available metadata, we can improve the performance of Hybrid Recommender Systems. In the next section, we analyze what solutions had been proposed in the past related works.

3.1.3 Related Work

In recent years, the way of users interact with computer systems changed considerably. In the dawn of the Internet, users were consumers of information. With the Web 2.0, users started to actively contribute and became providers of information. The quantity of information present in the Internet exploded and ways to mitigate the Information Overload was needed. Recommender Systems was the answer to this, and evolved closely with the Internet.

Nowadays popular websites such as Google[9] and Facebook[10]offers personalized results for users. Users now expect that computer systems to be smart enough and to provide answers personalized to them. The Microsoft CEO affirmed that digital assistants that personalized answer to a user are the future of computing [11].

In order to present this evolution and a literature review of recommendation systems area, Table 3.1 presents a research about relevant papers in the area who contributed to this dissertation, discussing the features and offering examples over time.

Furthermore, regarding the combination of metadata, recommender systems can be extended in several ways aiming at improving the understanding of users and items, incorporating new types of metadata or interactions in the recommendation process and making the combination of them. One of these improvements is the support for multi-criteria interactions, so as to provide greater flexibility and less obtrusive types of recommendations (RICCI; ROKACH; SHAPIRA, 2011). In this context, with more studies in the area of recommender systems, various algorithms enabled the usage of more than one type of user interaction.

These studies resulted in works such as Johansson (JOHANSSON, 2003), responsible for developing the MADFILM, a movie recommendation system that addresses the integration of prediction and organization of content, through explicit and implicit user's feedback. The work proposed by (YANG et al., 2007) developed a recommendation system for online video based on explicit and implicit feedback, plus feedback from relevant

[9]http://www.google.com
[10]http://www.facebook.com
[11]http://microsoft-news.com/microsofts-ceo-says-cortana-personal-assistants-replace-browser/

Table 3.1: Relevant works in Recommender Systems.

Year	**Remarkable examples**	**Description**
1992	**Papers:** Tapestry (GOLDBERG et al., 1992)	It was the first mention for the term "collaborative filtering". It began to arise as a solution for dealing with overload in online information spaces. The recommendation was not personalized and manual.
1994 - 2000	**Papers**: A series of systems for various domains, such as GroupLens for articles, Ringo (SHARDANAND; MAES, 1995) for music, the BellCore Video Recommender (HILL et al., 1995) for movies, and Jester (GOLDBERG et al., 2001) for jokes. **Industry examples:** Amazon.com	Collaborative filtering systems, based on ratings or other observable actions from the user, automatically combined them with the ratings or actions of other users to provide personalized results.
2002	**Papers**: Hybrid recommender system (BURKE, 2002)	Hybrid recommender systems emerged as various recommender strategies have matured, combining multiple algorithms into composite systems that ideally build on the strengths of their component algorithm (EKSTRAND; RIEDL; KONSTAN, 2011).
2003 - 2007	**Papers**: Social matching Recommender systems (TERVEEN; MCDONALD, 2005), Social recommender systems for web 2.0 folksonomies (SIERSDORFER; SIZOV, 2009), Tag-aware recommender systems (TSO-SUTTER; MARINHO; SCHMIDT-THIEME, 2008). **Industry examples:** Last.FM[12] and Del.icio.us[13]	With the creation of Web 2.0 sites, an enormous quantity of user created data became available. Recommenders started to use information's as tags to recommend items. In this period, also happened the ascension of social networks, and news algorithms where created to use this data.
2008-2014	**Papers:** Matrix Factorization (KOREN; BELL; VOLINSKY, 2009) ; Bayesian personalized ranking from implicit feedback (RENDLE et al., 2009); **Industry examples:**: Netflix, Facebook, Google, Youtube	In this period, a huge leap in Recommender systems research happened. Fueled by a greater number of internet users and initiatives such as the Netflix Prize competition a good number of innovate algorithms appeared. ThThiscompetition has demonstrated that matrix factorization models are superior to classic nearest-neighbor techniques for producing product recommendations (KOREN; BELL; VOLINSKY, 2009).
Tendencies	**Industry examples:** Facebook, Google Now, Siri, Cortana	Combining as many possible sources of data to produce a better recommendation, such as semantic data, social graph, item content and collaborative filtering.

information provided by the user. The used video was composed of multimedia content and related information (such as query, title, tags, etc.). The project aimed to combine these types of interactions with the information provided by users in order to generate a more precise rank of relevant items. In order to automatically adjust the system, it was implemented a set of adjustment heuristics given new user interactions.

The SVD++ algorithm proposed by (KOREN; BELL; VOLINSKY, 2009) uses explicit and implicit information from users to improve the prediction of ratings. As explicit information, the algorithm uses the ratings assigned by users to items, and as implicit information, it simulates the rental history by considering which items users rated, regardless of how they rated these items. However, it uses a stochastic gradient descent to train the model, which requires the observed ratings from users. Thus, it is impossible to infer preferences for those users who provided only implicit feedback.

Ensemble is a machine learning approach that uses a combination of similar models in order to improve the results obtained by a single model, and can be used to combine multiple metadata. In fact, several recent studies, such as (JAHRER; TöSCHER; LEGENSTEIN, 2010), demonstrate the effectiveness of an ensemble of several individual and simpler techniques, and show that ensemble-based methods outperform any single, more complex algorithm. Ensemble algorithms have been successfully used, for instance, in the Netflix Prize contest consisting of the majority of the top performing solutions (TÖSCHER; JAHRER; BELL, 2009; PIOTTE; CHABBERT, 2009).

Most of the related works in the literature point out that ensemble learning has been used in recommender system as a way of combining the prediction of multiple algorithms (heterogeneous ensemble) to create a stronger rank (JAHRER; TöSCHER; LEGENSTEIN, 2010), in a technique known as *blending*. They have been also used with a single collaborative filtering algorithm (single-model or homogeneous ensemble), with methods as *Bagging* and *Boosting* (BAR et al., 2013). We going to provide a more detailed analysis of the existing ensemble algorithms in the next section.

In the work of Randle et al. (RENDLE, 2010), the developers propose a technique called factorization Machines (FM), responsible for combining the advantages of Support Vector Machines (SVM) with factoring models. This technique can consider both information from items such as user information to generate the recommendation. However, the calculation of similarity between the information is done by pairs of comparison, which causes not as accurate results, as it does not take into consideration the semantics of data.

However, those solutions do not consider the multiple metadata present in the items, and are often not practical to implement in a production scenario because of the computational cost and complexity. In the case of heterogeneous ensemble, it needs to train all models in parallel and treat the ensemble as one big model, but unfortunately training 100+ models in parallel and tuning all parameters simultaneously is computationally not feasible (TÖSCHER; JAHRER; BELL, 2009). In contrast, the homogeneous ensemble demands the same model to be trained multiple times, and some methods such as Boosting requires that the underlying algorithm be modified to handle the weighted samples. Beltrão et al. (BELTAO et al., 2014) tried a different approach and combined multiple metadata by concatenating them, with a modest performance increase.

Our proposed solution involves three ensemble strategies that combine predictions from a recommender trained with distinct item metadata into a unified rank of recommended items. In comparison, da Costa at al. (FORTES; MANZATO, 2014), proposed a similar ensemble strategy based on machine learning in order to combine different types of interactions generated by multiple recommenders. Those strategies differ from the aforementioned works because they adopt a post-processing step to analyze the rankings created separately by different algorithms. This is because our method uses the user prediction (which is the least possible information in any Recommender System).

Our approach involves two voting strategies and a weighted strategy where the parameters are optimized using a Genetic Algorithm approach. The advantage of this approach is that it does not require the algorithm to be modified, or to be trained multiple times with the same dataset, and therefore, it is easier to extend the models to other types of interactions and recommenders.

In the next subsection, we describe in details some types of Ensemble that exists in literature, and how they are related to this work.

3.2 ENSEMBLE LEARNING FOR RECOMMENDER SYSTEMS

An ensemble method combines the predictions of different algorithms to obtain a stronger final prediction. Experimental results have shown that ensemble-based methods outperform any single, more complex algorithm (JAHRER; TöSCHER; LEGENSTEIN, 2010).

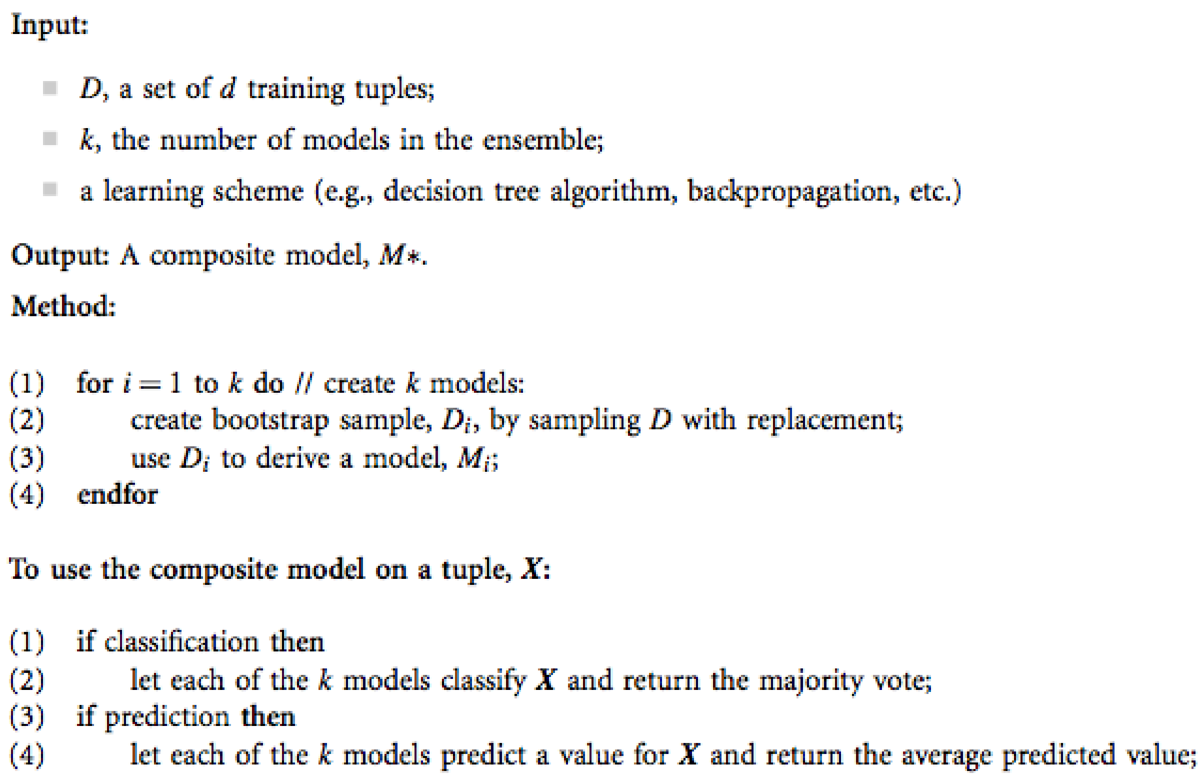
Constructing good ensembles of classifier has been a very active area of research in supervised (DITTERRICH, 1997), and can have two different ways of generating ensembles. One way is using a single learning algorithm (DIETTERICH, 2000) , such as decision tree learning or neural network training. Different classifiers are generated by manipulating the training set (as done in boosting or bagging), manipulating the input features, manipulating the output targets or injecting randomness in the learning algorithm. The generated classifiers are then typically combined by majority or weighted voting (DŽEROSKI; ŽENKO, 2004).

Another approach is to generate classifiers by applying different learning algorithms (heterogeneous models) to a single dataset. One popular way is Stacking (WOLPERT, 1992), a technique used to learn a combining method in addition to the ensemble of classifiers. Voting is then used as a baseline method for combining classifiers against which the learned combiners are compared (DŽEROSKI; ŽENKO, 2004). The two top-performers in the Netflix competition utilized blending, which may be considered to be a form of stacking (JAHRER; TöSCHER; LEGENSTEIN, 2010).

The following sections describe the ensemble approaches used in the literature:

3.2.1 Bagging

Bagging, or Bootstrap Aggregation combines multiple outputs of a learning algorithm by taking a plurality vote to get an aggregated single prediction. It decreases the variance of your prediction by generating additional data for training from your original dataset using combinations with repetitions to produce multisets of the same cardinality/size as your original data. Many experimental results show that bagging can improve accuracy substantially. The vital element in whether bagging will improve accuracy is the instability of the predictor (BREIMAN, 1996). For an unstable predictor, a small change in the training dataset may cause large changes in predictions(BREIMAN et al., 1996). For a stable predictor, however, bagging may slightly degrade the performance(BREIMAN, 1996).



Input:

- D, a set of d training tuples;
- k, the number of models in the ensemble;
- a learning scheme (e.g., decision tree algorithm, backpropagation, etc.)

Output: A composite model, $M*$.

Method:

```
(1) for $i = 1$ to $k$ do // create $k$ models:
(2)     create bootstrap sample, $D_i$, by sampling $D$ with replacement;
(3)     use $D_i$ to derive a model, $M_i$;
(4) endfor
```

To use the composite model on a tuple, X:

```
(1) if classification then
(2)     let each of the $k$ models classify $X$ and return the majority vote;
(3) if prediction then
(4)     let each of the $k$ models predict a value for $X$ and return the average predicted value;
```

Figure 3.3: The bagging algorithm (HAN; KAMBER; PEI, 2011) creates an ensemble of models (classifiers or predictors) for a learning scheme where each model gives an equally-weighted prediction.

The algorithm is described in Figure 3.3. Given a set, D, of d tuples, bagging works as follows (HAN; KAMBER; PEI, 2011). For iteration $i(i = 1, 2, 3, ..., k)$, a training set, D_i of d tuples is sampled with replacement from the original set of tuples, D. Because sampling with replacement is used, some of the original tuples of D may not be included in D_i, whereas others may occur more than once. A classifier model M_i is learned for each training set, D_i. To classify an unknown tuple, X, each classifier, M_i, returns its class prediction, which counts as one vote. The bagged classifier, M*, counts the votes and assigns the class with the most vote to X. If we use Bagging for prediction of continuous values, we can take the average value of each prediction for a given test tuple.

3.2.2 Boosting

Boosting combines the different decisions of a learning algorithm to produce an aggregated prediction. It calculates the output using several different models and then average the result using a weighted average approach. The weights of training instances change at each iteration to force learning algorithms to put more emphasis on instances that were predicted incorrectly previously and less emphasis on instances that were predicted correctly previously (DIETTERICH, 2000).

Intuitively, combining multiple models only helps when these models are significantly different from one another and when each one treats a reasonable percentage of the data correctly. Ideally, the models complement one another, each being a specialist in a part of the domain where the other models do not perform very well (HAN; KAMBER; PEI, 2011). The boosting method for combining multiple models exploits this insight by explicitly seeking models that complement one another. As bagging, boosting uses voting (for classification) or averaging (for numeric prediction) to combine the output of individual models. Again like bagging, it combines models of the same type, such as decision trees (HAN; KAMBER; PEI, 2011).

However, the boosting technique is iterative, and each new generated model is influenced by the performance of the models generated previously. The purpose of Boosting is to create new models, fixing the bad examples classified by previous models, so its strategy is to focus on the examples classified wrongly. For example, assuming there are eight training examples and that Example 1 is difficult to classify. Then, in each training set along the iterations of the algorithm, the Example 1 will be presented more times (as iterations of Table 3.2).

Table 3.2: Boosting example.

Training dataset	**Examples**
Training dataset (original)	1, 2, 3, 4, 5, 6, 7, 8
Training dataset 1 (boosting)	2, 7, 8, 3, 7, 6, 3, 1
Training dataset 2 (boosting)	1, 4, 5, 4, 1, 5, 6, 4
Training dataset 3 (boosting)	7, 1, 5, 8, 1, 8, 1, 4
Training dataset 4 (boosting)	1, 1, 6, 1, 1, 3, 1, 5

In addition, this technique gives a weight to each of the models generated according to how well they classified the training data. Thus, models with less mistakes will have a greater weight in classification of new examples while models with more mistakes will have a lower weight.

In a visual example, considering the dataset shown in Figure 3.4 (a), the samples misclassified by the first classifier h_1 increases their weight in the next distribution (represented by the sample size), as shown in Figure 3.4 (b). Then, a new classifier h_2 is generated considering the distribution of the new weights, as shown in Figure 3.4 (c), gives an even greater weight to examples difficult to classify. Finally, in Figure 3.4 (d) shows the combination of the final classifier models (h_1, h_2, h_3 and h_4) generated considering their weights

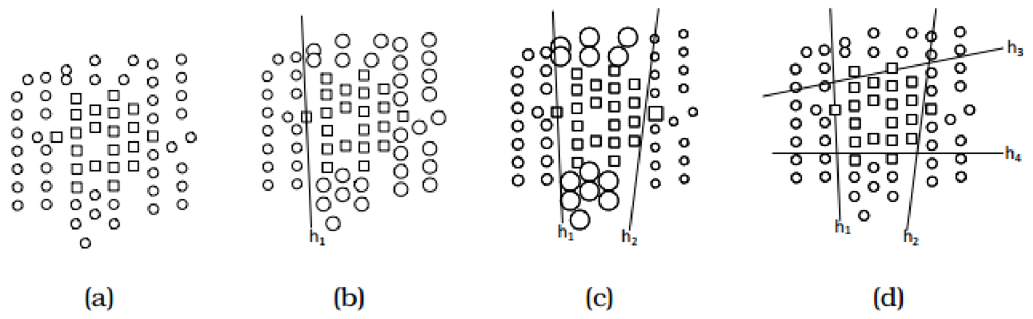
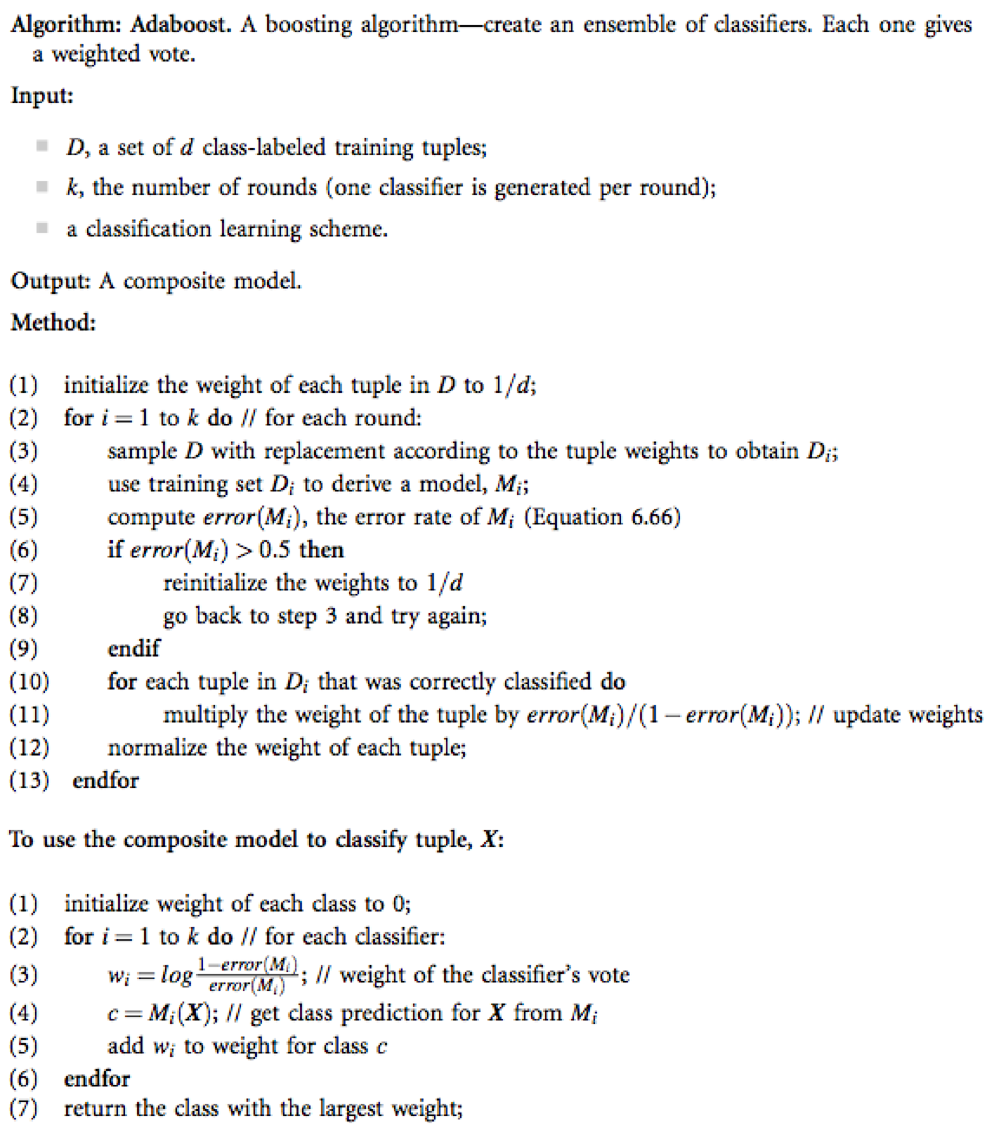


Figure 3.4: Example from LLerena (LLERENA, 2011) illustrating the Boosting technique: (a) initial dataset, (b) the dataset classified with the “weak” classifier (First Run), (c) the dataset classified with the “weak” classifier (Second Run) and (d) construction of the final classifier combining “weak” classifiers. The item size represents its weight.

A very popular boosting algorithm is AdaBoost (FREUND; SCHAPIRE, 1997). It is the abbreviation for adaptive boosting algorithm because it adjusts adaptively to the errors returned by classifiers from previous iterations. In AdaBoost, the input includes a dataset D of d class-labeled tuples, an integer k specifying the number of classifiers in the ensemble and a classification-learning scheme.



Algorithm: Adaboost. A boosting algorithm—create an ensemble of classifiers. Each one gives a weighted vote.

Input:

- D, a set of d class-labeled training tuples;
- k, the number of rounds (one classifier is generated per round);
- a classification learning scheme.

Output: A composite model.

Method:

```
(1)  initialize the weight of each tuple in $D$ to $1/d$;
(2)  for $i = 1$ to $k$ do // for each round:
(3)      sample $D$ with replacement according to the tuple weights to obtain $D_i$;
(4)      use training set $D_i$ to derive a model, $M_i$;
(5)      compute $error(M_i)$, the error rate of $M_i$ (Equation 6.66)
(6)      if $error(M_i) > 0.5$ then
(7)          reinitialize the weights to $1/d$
(8)          go back to step 3 and try again;
(9)      endif
(10)     for each tuple in $D_i$ that was correctly classified do
(11)         multiply the weight of the tuple by $error(M_i)/(1 - error(M_i))$; // update weights
(12)     normalize the weight of each tuple;
(13) endfor
```

To use the composite model to classify tuple, X:

```
(1)  initialize weight of each class to 0;
(2)  for $i = 1$ to $k$ do // for each classifier:
(3)      $w_i = log\frac{1-error(M_i)}{error(M_i)}$; // weight of the classifier's vote
(4)      $c = M_i(X)$; // get class prediction for $X$ from $M_i$
(5)      add $w_i$ to weight for class $c$
(6)  endfor
(7)  return the class with the largest weight;
```

Figure 3.5: AdaBoost Algorithm framework (HAN; KAMBER; PEI, 2011). The weights are assigned to each training tuple. A series of k classifiers is iteratively learned. After a classifier M_i is learned, the weights are updated to allow the subsequent classifier, $M_i + 1$, to "pay more attention" to the training tuples that were misclassified by M_i.

Each tuple in the dataset is assigned a weight. The higher the weight is the more it influences the learned theory. Initially, all weights are assigned a same value of $1/d$. The algorithm repeats k times. At each time, a model M_i is built on current dataset D_i which is obtained by sampling with replacement on original training dataset D. The framework (HAN; KAMBER; PEI, 2011) of this algorithm is detailed in Figure 3.5

3.2.3 Bayes optimal classifier

The Bayes Optimal Classifier creates an ensemble consisting of all the hypotheses in the hypothesis space. Each hypothesis is given a vote proportional to the likelihood that the training dataset would be sampled from a system if that hypothesis was true. The Bayes Optimal Classifier can be expressed with the following equation:

$$y = argmax_{c_j \in C} \sum_{h_i \in H} P\left(c_j|h_i\right) P\left(T|h_i\right) P\left(h_i\right) \tag{3.1}$$

Where y is the predicted class, C is the set of all possible classes, H is the hypothesis space, P refers to a probability, and T is the training data. Although, theoretically, on average, no other ensemble can outperform it (JAHRER; TöSCHER; LEGENSTEIN, 2010). It has only has been utilized in simple and small problems. There are several reasons why the Bayes Optimal Classifier are not used in practice:

- The argmax utility function requires a small number of hypothesis spaces to iterate over.
- It cannot be used if the hypotheses yield only a predicted class, rather than a probability for each class as required by the term $P(c_j|h_i)$.
- Estimating the prior probability for each hypothesis $P(h_i)$ is rarely feasible.
- Computing an unbiased estimate of the probability of the training set given a hypothesis $P(T|h_i)$ is non-trivial.

3.2.4 Stacking

Stacking generalization (WOLPERT, 1992), or stacking is a technique in which the predictions of a collection of models are given as inputs to a second-level learning algorithm. This second-level algorithm is trained to combine the model predictions optimally to form a final set of predictions. Stacking typically yields performance better than any single one of the trained models. Unlike bagging and boosting, stacking is not normally used to combine models of the same type. Instead it is applied to models built by different learning algorithms. As an example of the power of stacking, the team BellKor's Pragmatic Chaos won the $1 million prize offered by the Netflix Prize using a blend of hundreds of different models (TÖSCHER; JAHRER; BELL, 2009; PIOTTE; CHABBERT, 2009).

Consider that you have an ensemble learning, with multiple classifiers trying to fit to a training set to approximate the target function. Since each classifier will have its own output, we will need to find a combining mechanism to combine the results. One way to combine outputs is by voting—the same mechanism used in bagging. However, unweighted voting only makes sense if the learning schemes perform comparably well. If at least one of classifiers make predictions that are grossly incorrect, the results will get much worse.

Stacking introduces the concept of a meta learner, which replaces the voting procedure. The problem with voting you cannot guarantee that all classifiers are trustable.

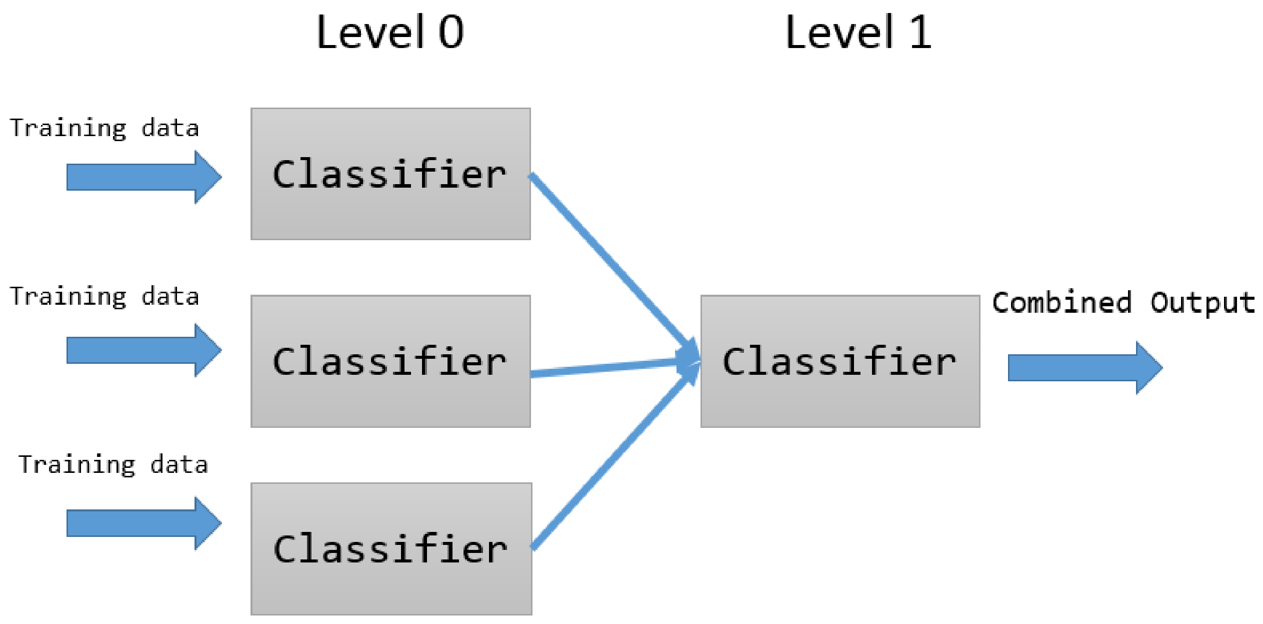


Figure 3.6: Stacking generalization overview.

Stacking tries to learn which classifiers are the reliable ones, using another learning algorithm - the meta learner — to discover how best to combine the output of the base learners (HAN; KAMBER; PEI, 2011)

As illustrated by the Figure 3.6 , the input to the level-1 model, or meta-model are the predictions of the base models, or level-0 models. A level-1 instance has as many attributes as there are level-0 learners, and the attribute values give the predictions of these learners on the corresponding level-0 instance (HAN; KAMBER; PEI, 2011).

Although some authors consider the same thing and use "stacked ensembling" and "blending" interchangeably, blending was a word introduced by the Netflix winners(TÖSCHER; JAHRER; BELL, 2009; PIOTTE; CHABBERT, 2009). Its similar o stacked generalization, but a bit simpler and less risk of an information leak. Performance-wise, both techniques are able to give similar results. In Figure 3.7 is depicted the solution, composed of approximately 500 predictors.

With blending, instead of creating out-of-fold predictions for the train set, you create a small holdout set of say 10% of the train set. The stacker model then trains on this holdout set only. It has some benefits: It's simpler than stacking; The generalizers and stackers use different data, preventing an information leak; and use can use as many models as you want and just throw models in the 'blender'. It decides if it wants to keep that model or not. However, it also has some issues. You use less data overall and there's a chance that the final model is overfit, as the holdout set is limited.

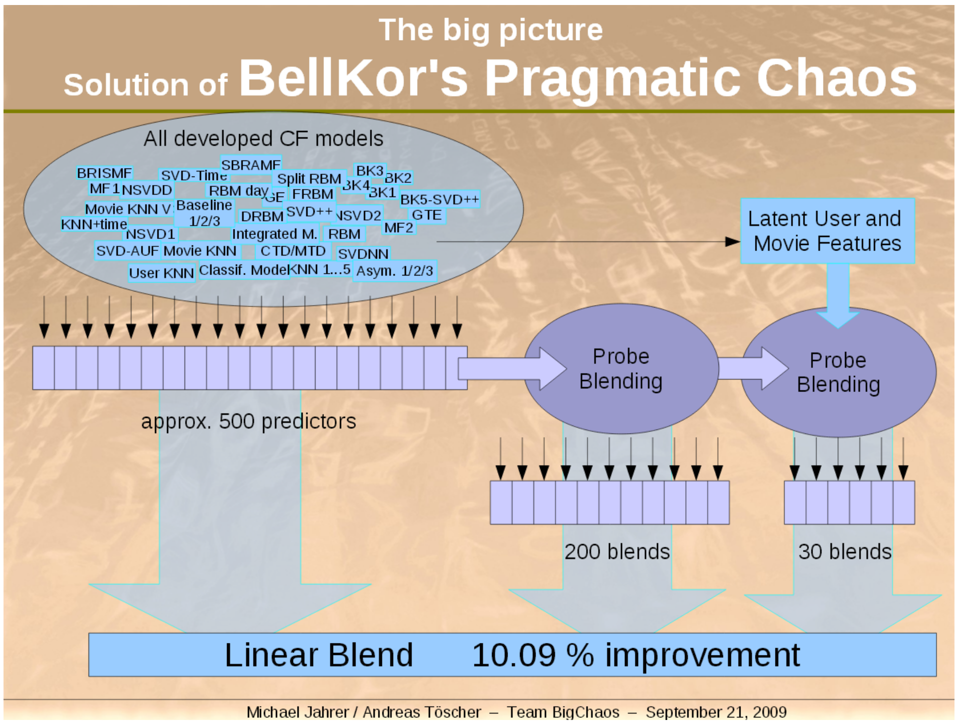


Figure 3.7: The BigChaos Solution to the Netflix Grand Prize (TÖSCHER; JAHRER; BELL, 2009).

3.3 SUMMARY

This chapter discussed the main concepts related to movies recommendation. It was characterized the problem including motivations, benefits and definitions and what are possible approaches that can be used to overcome the limitations. Then it was discussed the evolution of Recommender Systems, and how they relate to this work, highlighting and discussing related work and future trends. Finally, we made a thoughtful analysis of ensemble techniques, an alternative for combining together multiple metadata.

From what was presented in this chapter, you can see the importance of ensemble methods for recommendation systems and the need to further explore these techniques, given that the more information its available, the greater the probability of generating good recommendations to the user. The combination of metadata using the ensemble algorithms cited in this chapter, can greatly enhance the performance of recommender systems. However, the studies presented do not propose a general model for the creation of a recommendation system that can be used specifically for metadata combination. In this context, the next chapter presents a proposal to improve the recommendation performance by all available metadata.

Chapter 4

COMBINING METADATA USING ENSEMBLE METHODS

Believe nothing, no matter where you read it, or who said it, no matter if I have said it, unless it agrees with your own reason and your own common sense

—DREAMS COME DUE (John Calt)

As discussed in the previous chapters, using available metadata is important, and hybrid recommenders play an important role because they group together the benefits of content based and collaborative filtering. However, some recent systems which exploit latent factor models do not consider the metadata associated to the content, which could provide significant and meaningful information about the user's interests. Another issue is that the current metadata aware recommenders usually supports only one type of item attribute at a time. To overcome this issue, this work proposes a different approach for handling multiple metadata, using ensemble algorithms. In this chapter, we introduce multiple ensemble strategies to combine different metadata, but with the advantage that it does not require the algorithm to be modified, or to be trained multiple times with the same dataset, and therefore, it can be used in many current Recommender Systems.

In this chapter, is presented an open source recommendation tool for combining metadata, using ensemble algorithms developed during the research. This tool is based on the *MyMediaLite* tool (GANTNER et al., 2011), and implements multiple ensemble strategies, which support the idea that the more types of metadata the recommender system can handle, the more accurate the generated recommendations will be.

This chapter is organized as follows: Section 4.1 specifies the notation used; Section 4.2 describes the proposed ensemble algorithms; Section 4.3 presents the general architecture and details of the implementation; and, finally, Section 4.4 presents the summary of this chapter.

4.1 NOTATION

This chapter extends the notation established in Chapter 2. We need to recall that our recommenders produce a ranking of items. For generating the recommendations, this *Ranking-Oriented Recommender* receives as an input a dataset of ratings as a tuple $\langle u, i, r \rangle$, and outputs a matrix M_{UI}, where U is the set of all users and I is the set of all items known by the recommender system. Each row of the matrix M is composed of a vector of tuples $\langle i, \hat{r}_{ui} \rangle$, ordered by the item score prediction $\hat{r}_{ui}$ for the user u. The ensemble algorithms proposed in this paper can be formally defined as a function $f : M^K \rightarrow M$, where the input M^K is a vector of k-predictions and the output is a matrix of the combined predictions M.

4.2 ENSEMBLE STRATEGIES

The strategies elicited were inspired by group decision-making strategies that combine several users' preferences to aggregate item-ranking lists. According to Senot et al (SENOT et al., 2010) there are three categories of strategies, namely majority-based, which strenghten the "most popular" choice among the group, e.g. *Borda Count* and *Plurality Voting* strategies; *Consensus-based* strategies, which average somehow all the available choices, e.g. *Additive Utilitarian* and *Average without Misery*; and borderline strategies, also known as role-based strategies, which only consider a subset of choices based on user roles or any other relevant criterion, e.g. *Dictatorship*, *Least Misery* and *Most Pleasure* strategies.

In (BELTAO et al., 2014), it was investigated the problem of using multiple metadata and it was evaluated the performance of various metadata in the context of movie recommendation. It also experimented a simple strategy for combining different types of attributes by combining them in a unified metadata x item list. The results were promising, however, the performance improvement was moderate. Following this work in (CABRAL et al., 2014), we proposed an ensemble framework that consisted of training the recommender system for each different item metadata and combining them with one of the ensemble strategies presented next. The strategies can be defined as a type of *stacking* ensemble.

In the following sections we present proposed strategies: Most Pleasure, the simplest ensemble strategy, that combines predictions based on score; Best of All strategy, that determines a preferred metadata for a user and uses it to create the ensemble; and the Weighting strategy, that uses multiple metadata and weights them with a Genetic Algorithm optimizing the Mean Average Precision (MAP).

4.2.1 Naïve Combination Strategy

This is a very straightforward way of combining metadata, and was used as a baseline for our proposed strategies. It consists of getting pairs of attributes combining them, creating a linearly combined in pairs by concatenating the attributes.

The Naïve combination Strategy, proposed in (BELTAO et al., 2014), investigates the problem of using multiple metadata and it this solution consists of concatenating the different types of attributes as a single metadata x item list. Figure 4.1 illustrates this strategy.

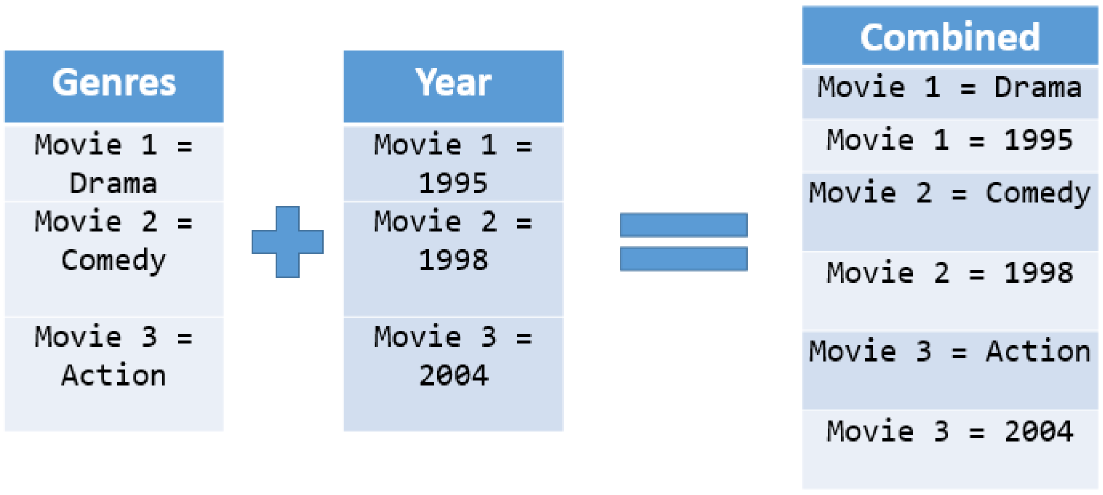


Figure 4.1: Naïve Combination Strategy. It concatenates two lists of attributes into one.

The results were positive when compared to not utilizing the metadata, however, the performance improvement was modest and more advanced strategies were proposed below.

4.2.2 Most Pleasure Strategy

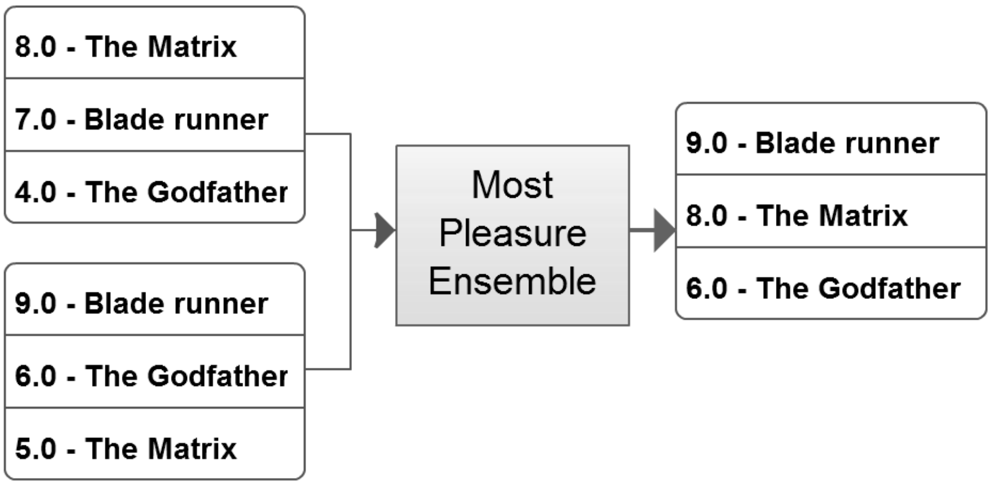


Figure 4.2: Most Pleasure Strategy.

The *Most Pleasure* strategy is a classic aggregation method, often used for combining individual ratings for group rating (MASTHOFF, 2011). It takes the maximum of individual ratings for a specific item and creates a unified rank. Figure 4.2 illustrates the *Most Pleasure* strategy, in which the output comprehends a ranked list of movies with highest ratings from two distinct input sets.

```
Input: Vector of predictions, $P$
Output: Predictions ensemble $M$
for $u$ = 1,...,#Users do
    for $i$ = 1,...,#Items do
        Select highest $\hat{r}_{ui}$ for the item $i$ among the K-predictions for the user $u$
        $M_{ui} \leftarrow (i, \hat{r}_{ui})$ //Store the highest score
    end
    Sort $M_u$ by $\hat{r}_{ui}$
end
```

Algorithm 2: Most Pleasure algorithm.

Algorithm 2 shows that it only needs the generated prediction vector as an input. This vector is composed of the predictions from the recommender algorithm trained with one of the item metadata. For each user, a new prediction is created, selecting the highest score of an item among all the individually-trained algorithms.

The idea behind this strategy is that differently trained algorithms have a distinct knowledge about the user's preferences, and the predicted score can be considered as an indicator of the algorithm's confidence. So the created ensemble is a list of items whose distinct algorithms have more confidence to recommend.

4.2.3 Best of All Strategy

The *Most Pleasure* strategy gives the same weight for different types of metadata. However, it is natural to assume that different types of metadata can affect users differently. In contrast, the *Best of All* strategy considers the recommendation algorithm that provides the best results for a specific user, and uses this algorithm to provide future predictions as illustrated in Figure 4.3.

The *Best of All* Strategy, as detailed in Algorithm 3 requires as an input: i) the recommendation algorithm, ii) a training dataset, iii) a probe dataset, and iv) the set of item's metadata. Unlike the *Most Pleasure* strategy, this one requires a probe run to determine which is the best performing algorithm. Therefore, the dataset is divided in training and probe. The recommender algorithm is firstly trained using each of item metadata individually. Then, for each user, a probe run is made to determine the metadata with the highest performance. This performance is indicated by the Mean Average Precision (MAP) metric (GOODRUM, 2000), often used for ranked recommendations. Finally, the algorithms are retrained using all data (including the probe set), and the final ensemble is the result of the combination of predictions using, for each user, the prediction from the algorithm with the highest performance in the probe test.

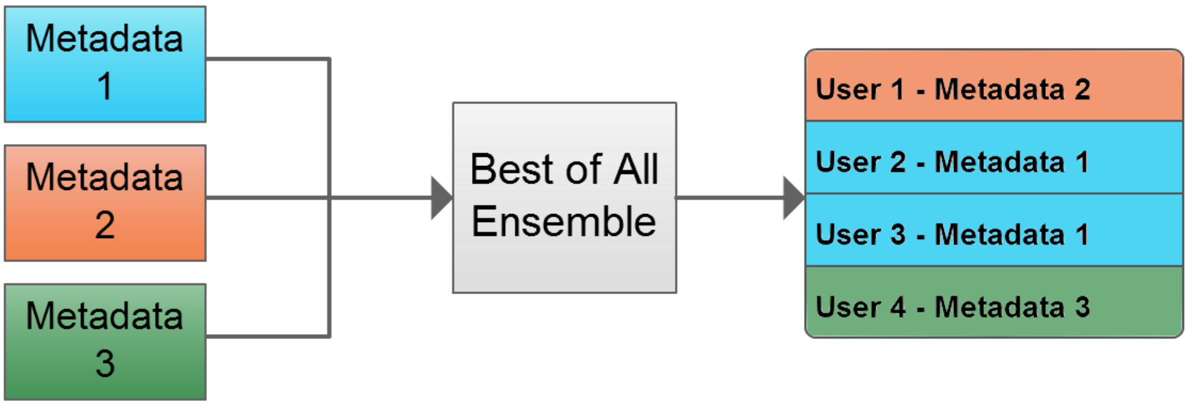


Figure 4.3: Best of All Strategy.

The idea behind this strategy is that a single metadatum can greatly influence the user's preferences, and this should be used for future predictions. For instance, if a User A enjoys films from a particular genre such as "horror", and other User B enjoys films of some specific theme such as "bloody films", the ensemble will contain predictions from the recommendation algorithm trained with both: the genre metadata for User A, i.e. "horror", and a keyword metadata for user B, i.e. "bloody".

4.2.4 GA Weighting Strategy

One drawback of the *Best of All* strategy is that it considers that only one type of metadata influences the user preference. However, the *GA Weighting* strategy assumes that the interests of a user may be influenced by more than one metadatum, and with different levels. The *GA Weighting* strategy considers all available metadata assigning different weights for each prediction as illustrated in Figure 4.4.

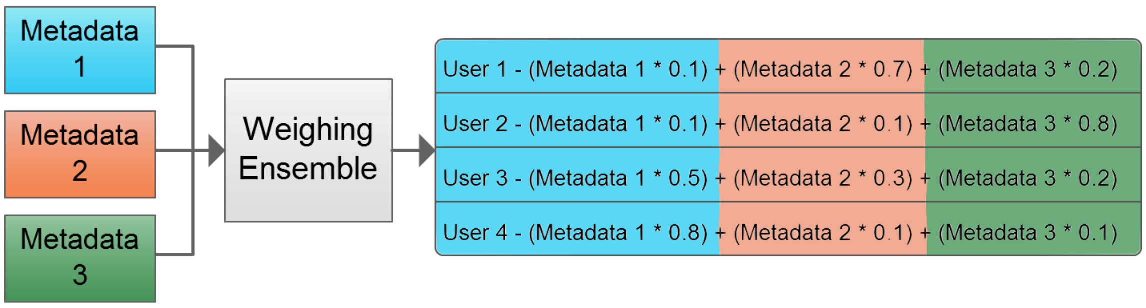

Metadata 1
Metadata 2
Metadata 3
Weighing Ensemble
User 1 - (Metadata 1 * 0.1) + (Metadata 2 * 0.7) + (Metadata 3 * 0.2)
User 2 - (Metadata 1 * 0.1) + (Metadata 2 * 0.1) + (Metadata 3 * 0.8)
User 3 - (Metadata 1 * 0.5) + (Metadata 2 * 0.3) + (Metadata 3 * 0.2)
User 4 - (Metadata 1 * 0.8) + (Metadata 2 * 0.1) + (Metadata 3 * 0.1)

Figure 4.4: Weighting Strategy.

Similarly to the previous strategy, the Algorithm 4 requires as an input: i) the recommendation algorithm, ii) a training and probe dataset, and iii) the set of item metadata.

```
Input: T - Training dataset of rating $< U, I, R >$
Input: P - Probe dataset of rating $U, I, R$
Input: $A$ - Vector of Metadata
Input: $PredAlg$ - the Base prediction algorithm
Output: Predictions ensemble $M$
for $m$ = 1,...,#Metadata do
    $K_m \leftarrow PredAlg$ Trained with T dataset and $A_u$
end
for $u$ = 1,...,#Users do
    Evaluate all $K$ models against the $P$ dataset and select the one with highest
      MAP for the user $u$ as $highest_u$
end
for $m$ = 1,...,#Metadata do
    $K_m \leftarrow PredAlg$ Trained with T+P dataset and $A_u$
end
for $u$ = 1,...,#Users do
    $\hat{r}_u \leftarrow K_{highest_u} u$
    $M_u \leftarrow \hat{r}_u$
end
```

Algorithm 3: Best of All algorithm.

After training the algorithm using each of item metadata individually, a probe run is also needed; however, the objective is to determine the optimal weights for each user. This is an optimization problem and that can be solved using a Genetic Algorithm (GA). GA is particularly appealing for this type of problem due to its ability to handle multi-objective problems. In addition, the parallelism of GA allows the search space to be covered with less likelihood of returning local extremes (NEWCOMBE, 2013).

The probe part consists of running the GA to find out the optimal weights. Our algorithm was implemented using the GA Framework proposed by Newcombe (NEWCOMBE, 2013), where the weights are the chromosomes, and the fitness function is the MAP score against the probe dataset. Other GA characteristics include the use of 5% of Elitism, Double Point crossing-over, and Binary Mutations. Finally, the algorithms are retrained using all data (including the probe set), and the final ensemble uses, as the item score, the sum of individual predictions multiplied by the weights found in the probe phase and divided by the total number of metadata.

The idea behind this strategy it that the different types of metadata influence differently the user preference. Still in the context of movies, let us consider two users: User A, that enjoys films from a determinate set of genres, but do not care about the production country and User B, that does not care about film genre or country of production. For the User A, the ensemble should give a higher weight for the film genre, and a lower weight for the production country. In contrast, to the User B, the ensemble should equally distribute the weights between those metadata.

```
Input: T - Training dataset of rating $< U, I, R >$
Input: P - Probe dataset of rating $< U, I, R >$
Input: $A$ - Vector of Metadata
Input: $PredAlg$ - the Base prediction algorithm
Output: Predictions ensemble $M$
for $m$ = 1,...,#Metadata do
    $K_m \leftarrow PredAlg$ Trained with T dataset and $A_u$
end
for $u$ = 1,...,#Users do
    Get weights $w_u$ for all $K$ models against the $P_u$ dataset for the user $u$ using a
      Genetic Algorithm, where the MAP is the Fitness function.
end
for $m$ = 1,...,#Metadata do
    $K_m \leftarrow PredAlg$ Trained with T+P dataset and $A_u$
end
for $u$ = 1,...,#Users do
    $\hat{r}_{ui} \leftarrow \sum_{i=1}^{Metadata} w_{ui} K_i / Metadata$
    $M_{ui} \leftarrow \hat{r}_{ui}$
end
```

Algorithm 4: Weighting algorithm.

4.2.5 BPR Learning Strategy

This strategy was not proposed by this work, it was originally proposed by (FORTES; MANZATO, 2014), for combining different kinds of interactions. We use it in our evaluation as a comparative. It is similar to our Weighting strategy. The main difference, is that it uses a Gradient Decent as optimization technique and tries to maximize the ROC curve, while the Weighting optimize the MAP metric.

In order to combine the output generated by each recommendation technique trained with a different kind of interaction, this ensemble strategy is based on a machine learning algorithm (FORTES; MANZATO, 2014). Firstly, it extracts information about users' interactions from the database, such as sets of tags, ratings and browsing history. With these interactions available, it runs the recommendation algorithms, which receive as input the users' interactions. In this step, each algorithm runs with a particular set of feedback, resulting in a feedback-specific personalized ranking (individual ranking) for each user. Thus, a feedback-specific ranking contains the items and their associated scores, which represent how much a user likes an item described by the considered set of attributes. The final step consists of combining all considered rankings into a final list of recommendations. To do that, it assigns weights according to the relevance of each type/set of attributes. This combination is performed according to a linear function, represented by $\hat{r}_{u,i}^{final}$:

$$\hat{r}_{ui}^{final} = \beta_a r_{ui}^a + \beta_b r_{ui}^b + ... + \beta_n . r_{ui}^n. \tag{4.1}$$

where r_{ui}^a, r_{ui}^b, ..., r_{ui}^n indicate the scores computed previously by each individual recommendation algorithm for a (u, i) pair, and β_a, β_b, ..., β_n are the weights of each individual score for the final prediction, learned using Learn BPR Algorithm 5. This is possible because of the natural strategy of BPR, which in a each interaction, select randomly a couple of items i and j for a user u, a known item i and one unknown item j.

```
Input: $D_K$
Output: Learned parameters $\Theta$
Initialize $\Theta$ with random values
for $count = 1,...,\#Iter$ do
    draw $(u, i, j)$ from $D_K$
    $\hat{s}_{uij} \leftarrow \hat{r}_{ui} - \hat{r}_{uj}$
    $\Theta \leftarrow \Theta + \alpha \left( \frac{e^{-\hat{s}_{uij}}}{1+e^{-\hat{s}_{uij}}} . \frac{\partial}{\partial \Theta} \hat{s}_{uij} - \Lambda_\Theta \Theta \right)$
end
```

Algorithm 5: Learning through LearnBPR.

Finally, the algorithm predicts scores for items not seen by each user and sorts these scores in descending order resulting in the final ranking, which will be recommended in a top N ranking list.

The underlying characteristic of this algorithm is the ability to learn the users' preferences and employ this information to match the recommendations generated individually for each type of interaction.

4.3 ALGORITHMS IMPLEMENTATION

For implementing the ensemble methods described in the previous section it was chosen not to develop each strategy individually, where the solution would be used only in this work and let it to bit rot. Instead we decided to build an extensible platform where new ensemble algorithms can be built on and with enough flexibility so other developers can utilize it.

MyEnsembleLite was the result of this idea. It is an open-source, publicly available[1]. All four ensemble strategies presented previously were implemented on it. It was built using the *MyMediaLite* library (GANTNER et al., 2011), which is a fast and scalable, multi-purpose library of recommender system algorithms, aimed both at recommender system researchers and practitioners. It has been chosen for a number of reasons:

- **Stable and Established tool.** *MyMediaLite* initial release was on October 21, 2010, with the latest version 3.11, released in February, 2015. It has been used in many industrial projects and research (GANTNER et al., 2011). Thus, we have good indicators of stability and efficiency.

[1]https://github.com/wendelad/RecSys

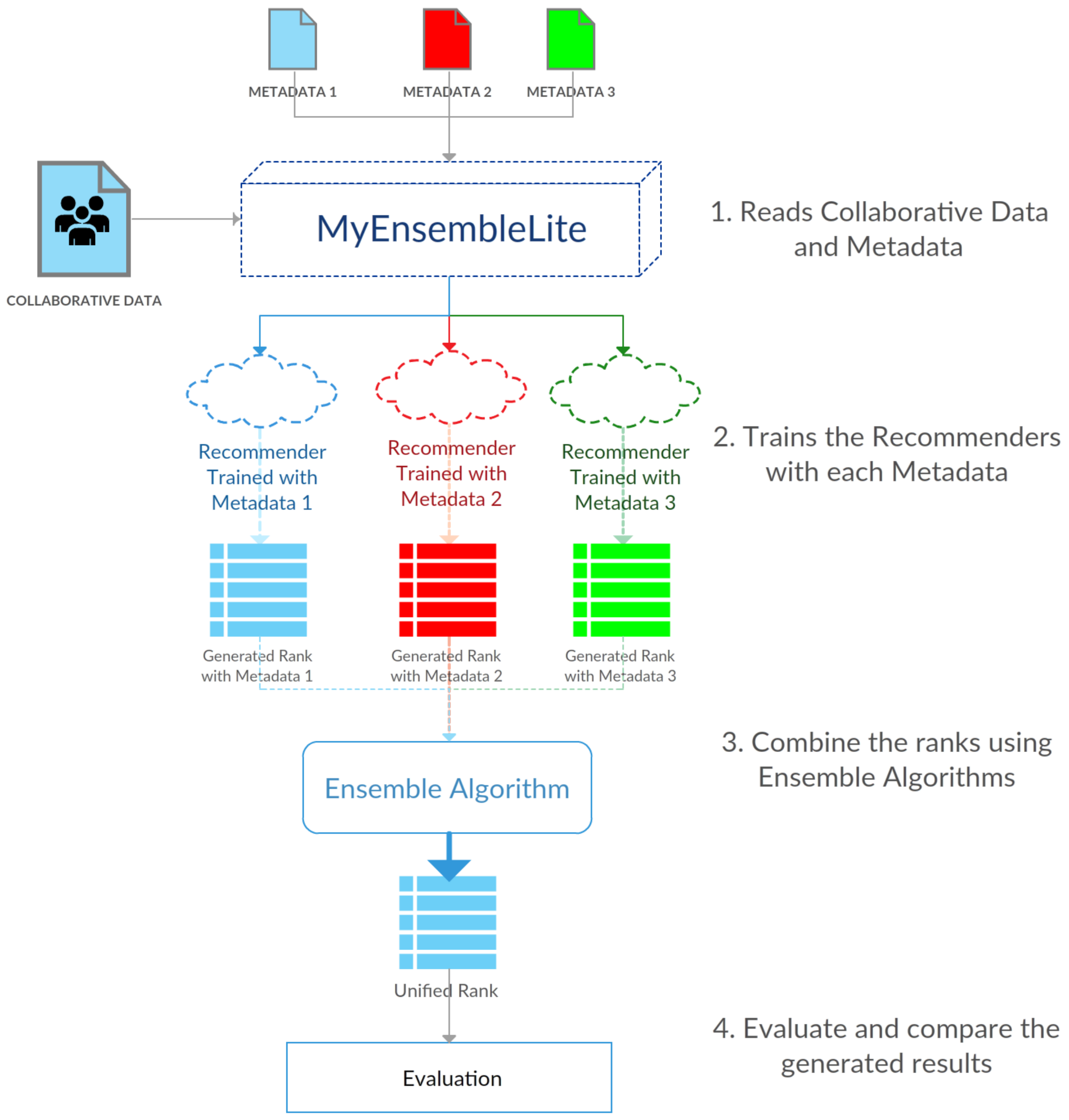
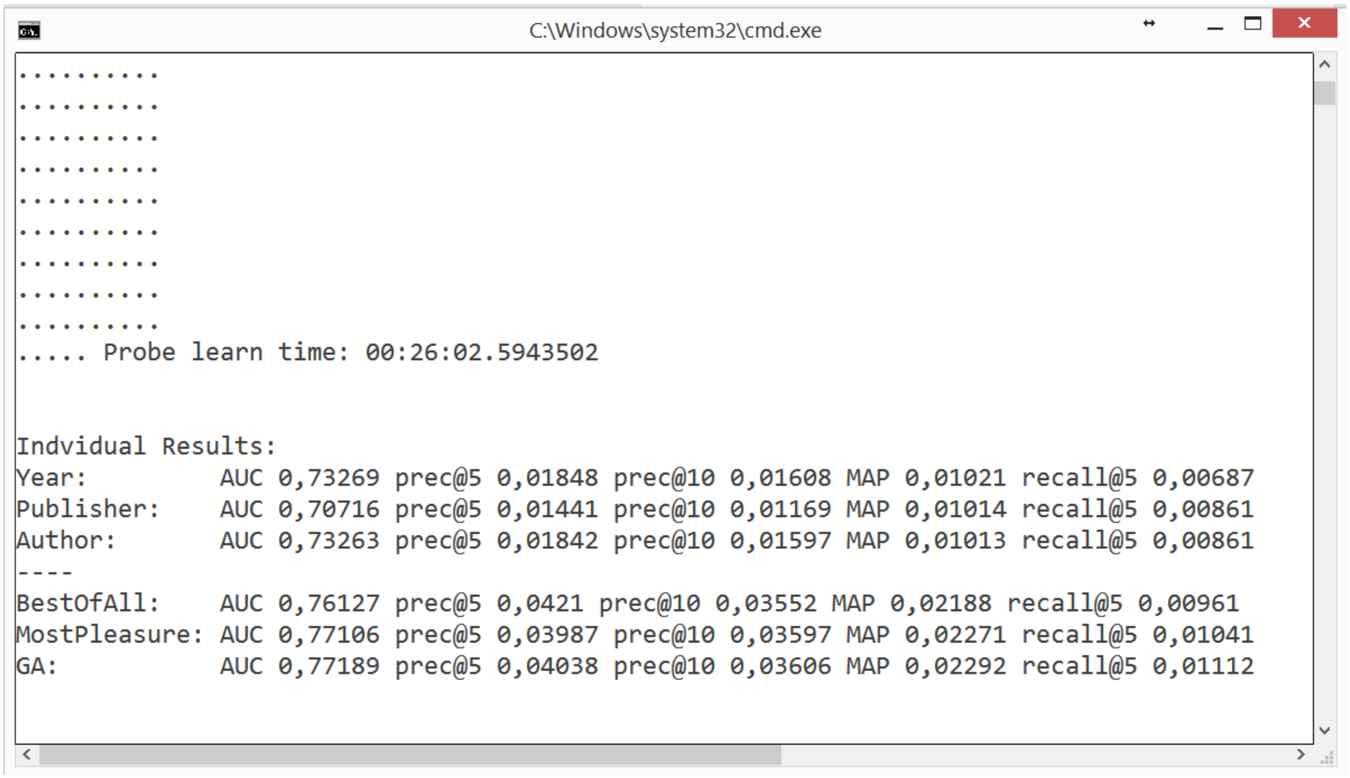

METADATA 1
METADATA 2
METADATA 3
COLLABORATIVE DATA
MyEnsembleLite
1. Reads Collaborative Data and Metadata
Recommender Trained with Metadata 1
Recommender Trained with Metadata 2
Recommender Trained with Metadata 3
2. Trains the Recommenders with each Metadata
Generated Rank with Metadata 1
Generated Rank with Metadata 2
Generated Rank with Metadata 3
Ensemble Algorithm
3. Combine the ranks using Ensemble Algorithms
Unified Rank
Evaluation
4. Evaluate and compare the generated results

Figure 4.5: Overview of *MyEnsembleLite* Architecture.

- **Set of features.** After examination of other open-source solutions, we found that *MyMediaLite* had a good number of features needed to build our tool. Because, it supports the two most common scenarios in collaborative filtering: rating prediction and item prediction from positive-only implicit feedback, offering multiple state-of the-art algorithms for those two tasks, with dozens of different recommendation methods. It also supports online-updates, serialization of computed models, and a rich set of routines for evaluation (AUC, MAP, precision@N, recall@N and more).

- **Portable.** It is implemented in C#, and runs on the .NET platform. With the free .NET implementation Mono, it can be used on multiple operating systems such as Linux, Windows and MacOS. Using the library is not limited to C#, though: it has wrappers from many other languages such as Ruby, Clojure[2] and Python;

- **Liberal license.** *MyMediaLite* uses the GNU General Public License (GPL).[3], a permissive and very popular free software license, that enables us to modify and distribute the code while accepting contributions from external developers.



```
..........
..........
..........
..........
..........
..........
..........
..........
..........
..... Probe learn time: 00:26:02.5943502


Indvidual Results:
Year:         AUC 0,73269 prec@5 0,01848 prec@10 0,01608 MAP 0,01021 recall@5 0,00687
Publisher:    AUC 0,70716 prec@5 0,01441 prec@10 0,01169 MAP 0,01014 recall@5 0,00861
Author:       AUC 0,73263 prec@5 0,01842 prec@10 0,01597 MAP 0,01013 recall@5 0,00861
----
BestOfAll:    AUC 0,76127 prec@5 0,0421 prec@10 0,03552 MAP 0,02188 recall@5 0,00961
MostPleasure: AUC 0,77106 prec@5 0,03987 prec@10 0,03597 MAP 0,02271 recall@5 0,01041
GA:           AUC 0,77189 prec@5 0,04038 prec@10 0,03606 MAP 0,02292 recall@5 0,01112
```

Figure 4.6: Example output of the MyEnsembleLite Tool.

In Figure 4.5, the architecture is detailed, illustrating how *MyEnsembleLite* implements the ensemble techniques in this work. It is composed of four big modules:

1. **File pre-processing.** This module is responsible to do any sort of file-preprocessing, for example, the *MyEnsembleLite* needs a specific input format, often different than the available datasets. The developer had to convert to the needed format to be able to run experiments. The official answer to this, was to create Perl scripts for each dataset. However, for beginners this can be an annoyance. This module have

[2]https://github.com/timgluz/clj-mml
[3]http://opensource.org/licenses/GPL-2.0

some boilerplate and examples so a dataset can be easily converted to the right format.

This module is also responsible for splitting the files in multiple k-folds, for a cross validation evaluation. Although the *MyMediaLite* already supports splitting a file for cross-validation. It has a non-deterministic way of splitting the k-folds and generating the files, causing subsequently runs to produce slightly different results. For some experiments it was required to have a fixed set of k-folds for reproducibility. So this module enables splitting files and saving the output.

2. **Training.** This module is responsible for training the recommenders. It can use any algorithm available in *MyMediaLite*. For example, in an ensemble of three meta-data's, it is responsible for training each recommender with the specific metadata to later provide the Ensemble module with the results for ensembling.

3. **Ensemble.** Here is the core of *MyEnsembleLite*, where the ensemble strategies were implemented. It provides a clear interface, where the ensemble strategies receive the results of previous probe run (if it requires it), and the set of results from previous trained recommenders. Actually, it also utilizes a Genetic Algorithm library for weighting optimization.

4. **Evaluation.** This module extends the evaluation framework available in MyMediaLite, enabling it to comprehends the output generated from the Ensemble Module. Actually it supports: AUC; precision@; MAP; recall; NDCG and MRR.

In the Figure 4.6 it can be seen the output of a run. It shows the evaluation in various metrics (MAP, AUC, Precision, Recall, etc...) for each recommender trained with a single metadata, and the result for each ensemble. It can use any Recommender Algorithm implemented in *MyMediaLite*, however in this context of combining multiple metadata make no sense to use any algorithm that is not attribute/metadata aware. Actually the following recommenders supports attributes:

- **ItemAttributeKNN**, a k-nearest neighbor (kNN) item-based collaborative filtering using the correlation of the item attributes.

- **UserAttributeKNN**, a k-nearest neighbor (kNN) user-based collaborative filtering using the correlation of the user attributes.

- **ItemAttributeSVM**, a content-based filtering using one support-vector machine (SVM) per user.

- **MostPopularByAttributes**, a simple algorithm that always recommends the most popular items by attribute.

- **BPRMF-Mapping, BPRMFAttr , BPR-GSVDPlusPlus and BPRLinear**. A set of matrix factorization algorithms that implements the Bayesian Personalized Ranking that also takes into account what users have rated and its attributes.

The *MyEnsembleLite* tool is public available in a GitHub Repository[4]. It is still in development, however it does have a working prototype. The use instructions are available in the embedded help as some working examples. It has been tested in Windows and Linux (using Mono) environments. Contributions are welcome and expected.

4.4 SUMMARY

This chapter introduced the *MyEnsembleLite* tool, and presented four ensemble strategies capable of processing multiple metadata to generate a more accurate recommendation. Recommender algorithms may not be able to take advantage of multiple types of metadata and the proposed ensemble algorithms enable those recommenders to take advantage of all available metadata.

Initially, the presented techniques were based on heuristics, gathering the recommendations generated individually and combining them, namely *Most Pleasure*, *Best of All*. *Most Pleasure*, the simplest strategy, consisted of combining the predictions based on score, while *Best of All* determined a single metadata that was more preferred for a user. However, despite the results of these techniques have been positive when compared to not utilizing the metadata, as will be seen in Chapter 5, the strategies where very naïve and something more sophisticated could be used. Consequently, two new approaches were developed in order to combine multiple metadata. Those strategies also process the results generated individually for each type of metadata, but use a machine learning technique to examine each type of metadata when combining the ranks. Specifically, *GA Weighting* strategy uses multiple interactions and weights them with a Genetic Algorithm that optimizes the MAP and finally, *BPR Learning* uses LearnBPR to optimize the weights related to AUC.

In the next chapter, the experiments and results of the evaluation of proposed ensemble strategies will be presented.

[4]https://github.com/wendelad/RecSys

Chapter 5

EVALUATION

If we're facing in the right direction, all we have to do is keep on walking.

—JOSEPH GOLDSTEIN (The Experience of Insight)

This chapter will present multiple evaluations performed to verify that the objectives specified in this work have been achieved with the development of the ensemble techniques. It is expected that with the combination of multiple metadata, there will be an improvement in the quality of recommendations in terms of precision. The evaluation consists in comparing the ensemble strategies presented in the previous chapter, using standard datasets available in the literature. The experiments comprehend four experiments, where three are in the movie domain: One with a Naïve combination of metadata; Another using all the proposed ensemble techniques; and, finally one with a different objective (user interaction) in the movies domain. A final experiment in the Books domain was performed to guarantee that the proposed techniques in this master thesis generalizes to another domain.

This chapter is organized as follows: in Section 5.1 we present details of the methodology used for the development and evaluation of this work, tools and datasets used during the experiments; from Section 5.2 to Section 5.5 all experiments are depicted;Section 5.6 discuss the results and finally, Section 5.7 presents the final remarks.

5.1 METHODOLOGY

The objective of the evaluation presented in this work is to validate the proposed ensemble strategies in Chapter 4. It is used the algorithms presented in Chapter 2 with real-world datasets to verify the performance of Recommender Systems performance by using standard evaluations metrics such as MAP and AUC. To this end, four experiment were performed. The goals of each experiment are described as following:

- **Experiment 1.** This was an initial evaluation, to validate the hypothesis that metadata combination could improve the performance of Recommender Systems.

It was used the MovieLens 100k [1] dataset with some additional data extracted from IMDB. The *Naïve Combination Strategy* was used for combining five different types of metadata. This experiment also gave insights of the most performant metadata for this particular dataset.

- **Experiment 2.** Following the previous experiment, in this evaluation, it was used more advanced ensemble strategies proposed in Chapter 4 to combine metadata. It was used the HetRec 2011 MovieLens 2k dataset (CANTADOR; BRUSILOVSKY; KUFLIK, 2011).

- **Experiment 3.** This was the last experiment in the movie domain. In this experiment it was compared the proposed ensemble strategies for a different objective. The difference was that the ensemble strategies were applied not to combine metadata but user interactions (historic, tags and explicit rating). It has also used the HetRec 2011 2k dataset (CANTADOR; BRUSILOVSKY; KUFLIK, 2011).

- **Experiment 4.** Because all previous experiments were performed in the movie recommendation domain, another experiment in a different domain was needed in order to verify whether the proposed ensemble techniques generalize and not were specific to the movies domain. This experiment was executed with the Book-Crossing dataset, of Books evaluations.

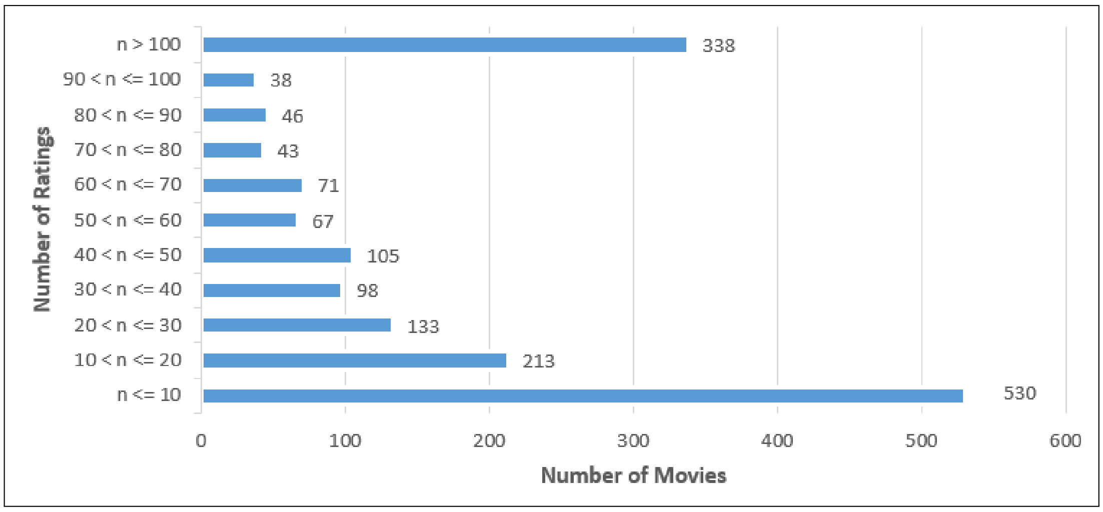
The following subsections presents other aspects of the methodology used in this work. Section 5.1.1 details the datasets used where the experiments were conducted; and, in Section 5.1.2 shows how the work was evaluated.

5.1.1 Datasets

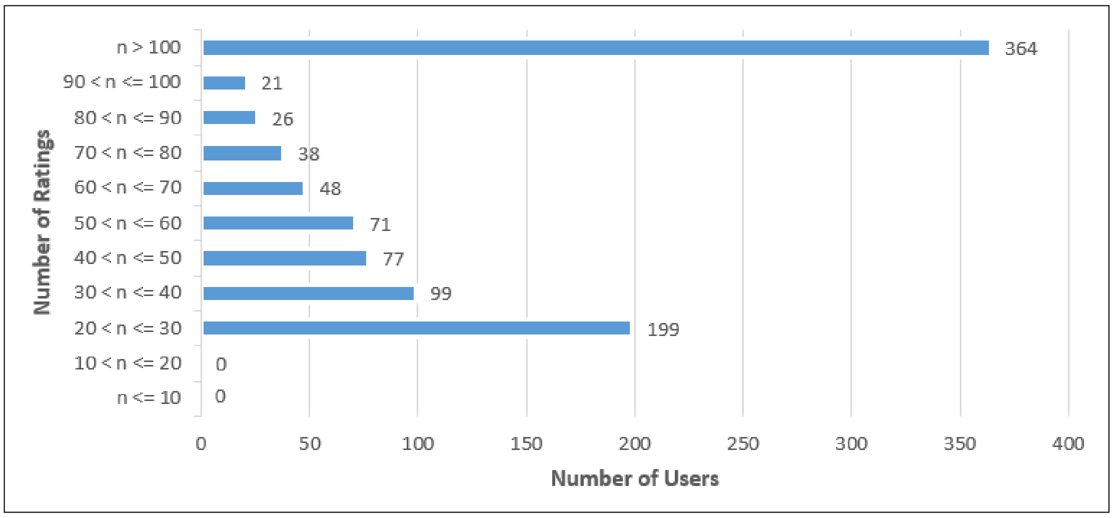
To evaluate the proposed ensemble techniques, we used public available datasets. The experiments were performed in four different datasets. Three in the movie domain and one in the book domain. In this section, we present information about those datasets.

5.1.1.1 Extended MovieLens 100k. In this work, it was used the 100k MovieLens database[2] combined with Internet Movie Database(IMDB)[3] in order to infer which is the best algorithm in movie recommendation. The MovieLens[4] is a Website that provides movie recommendations. The GroupLens[5] research group provides three databases generated from the base of MovieLens called MovieLens 100k (100k-ML), MovieLens 1M (ML-1M) and MovieLens 10M (ML-10M). Such bases vary in number of items, users' and reviews. It was utilized the 100k version. Because the MovieLens dataset has few metadata about the movies (only genres and release date), it was extracted additional

[1]http://www.grouplens.org/node/73
[2]http://www.grouplens.org/node/73
[3]http://www.imdb.com/interfaces
[4]http://www.movielens.org
[5]http://www.grouplens.org



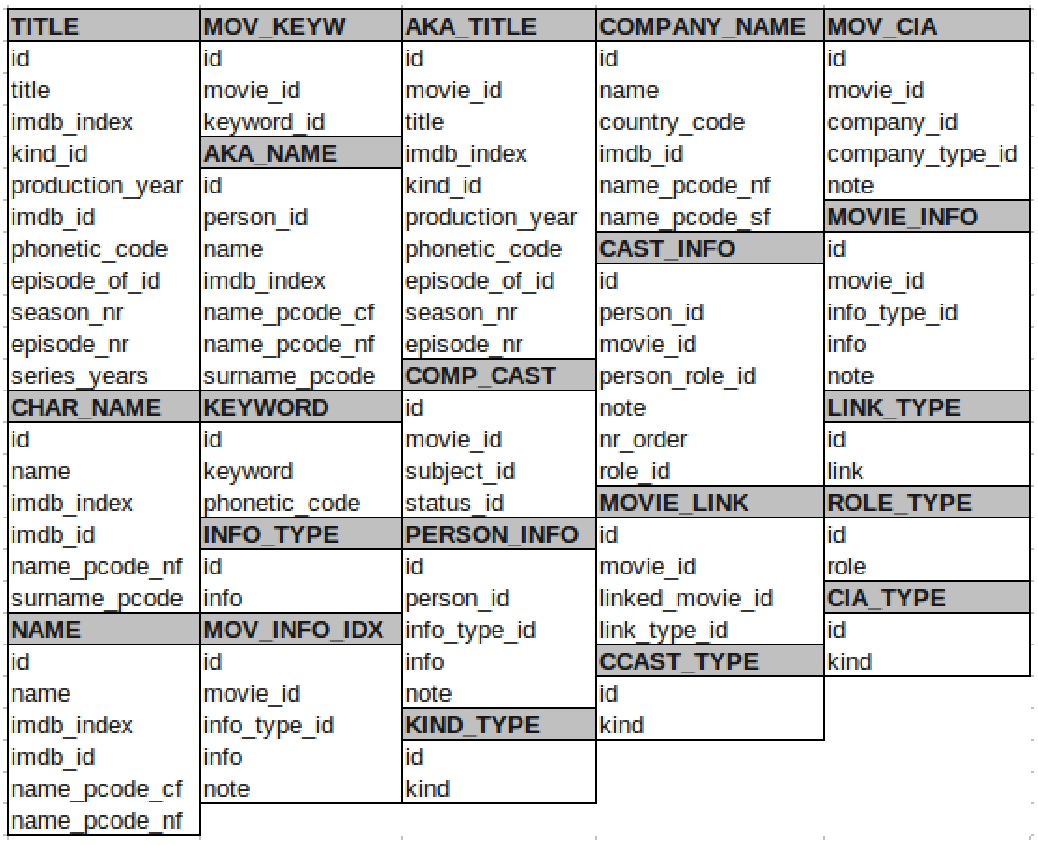
(a) Distributions of movie ratings.



(b) Distributions of user ratings.

Figure 5.1: Distributions of Movie ratings and User ratings in MovieLens 100k dataset.

information from IMDB database, thus enriching the movie dataset information. Figures 5.2 and 5.3 illustrate the items present in each dataset.



TITLE
id
title
imdb_index
kind_id
production_year
imdb_id
phonetic_code
episode_of_id
season_nr
episode_nr
series_years

CHAR_NAME
id
name
imdb_index
imdb_id
name_pcode_nf
surname_pcode

NAME
id
name
imdb_index
imdb_id
name_pcode_cf
name_pcode_nf

MOV_KEYW
id
movie_id
keyword_id

AKA_NAME
id
person_id
name
imdb_index
name_pcode_cf
name_pcode_nf
surname_pcode

KEYWORD
id
keyword
phonetic_code

INFO_TYPE
id
info

MOV_INFO_IDX
id
movie_id
info_type_id
info
note

AKA_TITLE
id
movie_id
title
imdb_index
kind_id
production_year
phonetic_code
episode_of_id
season_nr
episode_nr

COMP_CAST
id
movie_id
subject_id
status_id

PERSON_INFO
id
person_id
info_type_id
info
note

KIND_TYPE
id
kind

COMPANY_NAME
id
name
country_code
imdb_id
name_pcode_nf
name_pcode_sf

CAST_INFO
id
person_id
movie_id
person_role_id
note
nr_order
role_id

MOVIE_LINK
id
movie_id
linked_movie_id
link_type_id

CCAST_TYPE
id
kind

MOV_CIA
id
movie_id
company_id
company_type_id
note

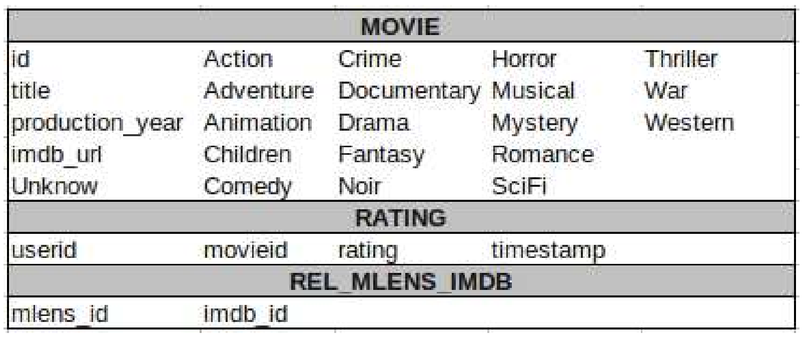
MOVIE_INFO
id
movie_id
info_type_id
info
note

LINK_TYPE
id
link

ROLE_TYPE
id
role

CIA_TYPE
id
kind

Figure 5.2: IMDB database.



MOVIE				
id	Action	Crime	Horror	Thriller
title	Adventure	Documentary	Musical	War
production_year	Animation	Drama	Mystery	Western
imdb_url	Children	Fantasy	Romance	
Unknow	Comedy	Noir	SciFi	

RATING			
userid	movieid	rating	timestamp

REL_MLENS_IMDB	
mlens_id	imdb_id

Figure 5.3: MovieLens database.

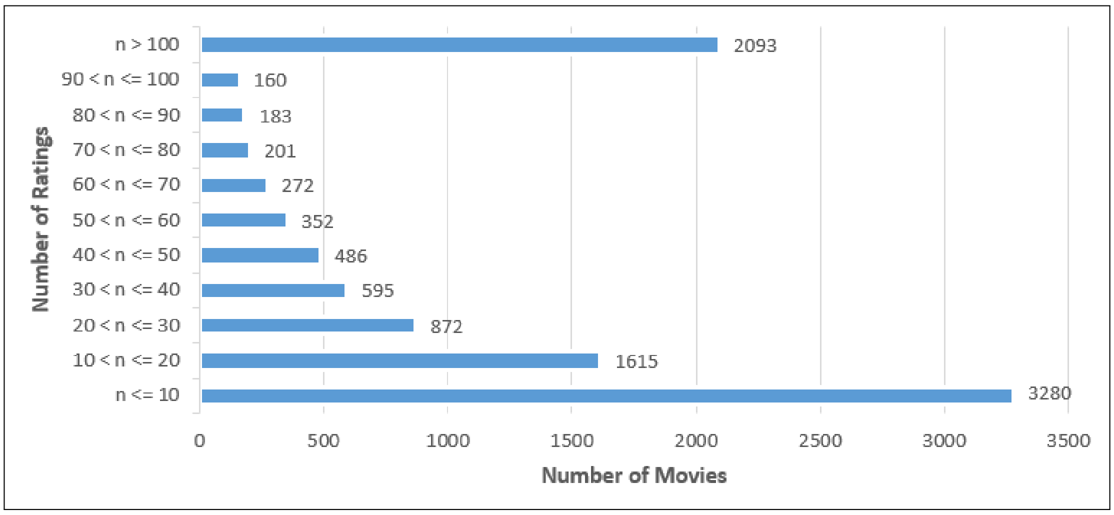
Firstly, we tried to align the information of both datasets using the indexes to generate a unified dataset. However, since the indexes from IMDB and Movielens are not always similar, it was used then a search using the title and year present in the MovieLens dataset to match the movies index in IMDB and recover the information we wanted. It was necessary to modify the data in MovieLens because the movie titles were written in English form (e.g. Godfather, The). Therefore, we changed the names to the form used in IMDB (e.g. The Godfather). The discovery of these indexes enabled us to extract the information we needed, i.e. genre, actor, writer, director and keyword. With this metadata, we created a unified dataset, connecting the movies with their metadata. As we only used the movies from MovieLens dataset, the additional information extracted from IMDB was incorporated to the MovieLens dataset. Worth mentioning that only three movies did not have additional information extracted from IMDB, which did not impact the results.

The final augmented database of MovieLens 100k, contains 100,000 ratings of 943 users on 1682 movies, with 5 different metadata : actors; directors; genres; keywords; and, writer. In the Figure 5.1 we present details on the distribution of users and movies ratings. It can be seen that all users had rated at least 20 movies, and 39% rated more than 100 movies. From the user perspective, cold-start would not be a problem. In the other hand, looking at the Distribution of Movie Ratings, can be seen that 31% of movies had not received more than 10 ratings, with 20% of receiving more than a 100 ratings. This is probably because few movies are blockbusters, attracting a big number of ratings. In fact, the most rated movie in this dataset was Star Wars (1977), with 583 ratings.

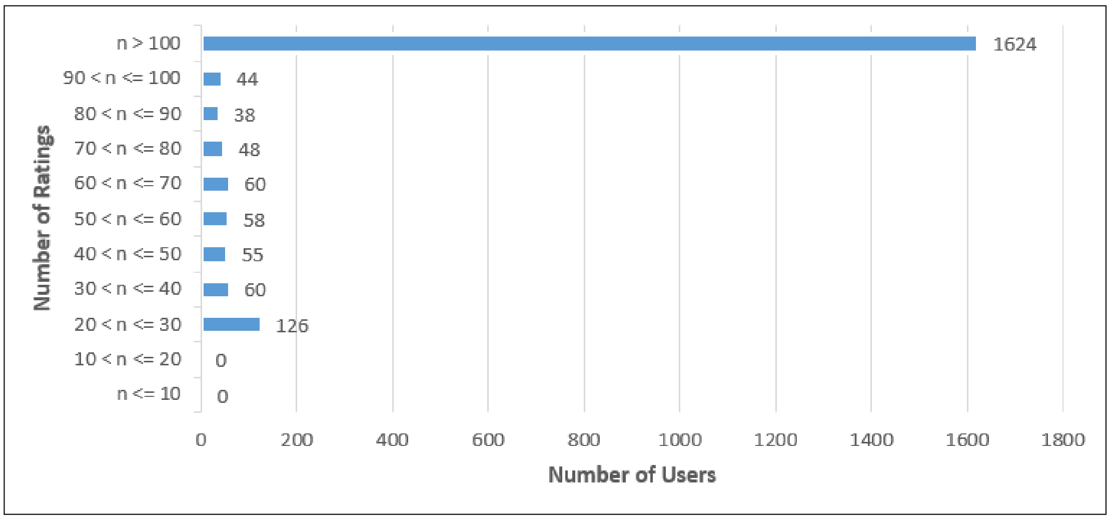
5.1.1.2 HetRec2011 MovieLens 2k. Later, we found out that an extended MovieLens database had been released. HetRec 2011 MovieLens 2k(CANTADOR; BRUSILOVSKY; KUFLIK, 2011) is an extension of MovieLens10M dataset, released for the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec 2011 [6]), which contains personal ratings and tags about movies. In the dataset, MovieLens movies are linked to the Internet Movie Database (IMDb) [7] and RottenTomatoes (RT) [8] movie review systems. Each movie has its IMDb and RT identifiers, English and Spanish titles, picture URLs, genres, directors, actors (ordered by "popularity"), countries, filming locations, and RT audience' and experts' ratings and scores. The dataset was composed of 2113 users with 855598 ratings on 10197 movies, including the relation between 20 movie genres, 4060 directors, 95321 actors, 72 countries and 13222 tags.

In the Figure 5.4 depicts the distribution the distribution of users and movies ratings. It can be seen that all users had rated at least 20 movies, and the majority (77%) rated more than 100 movies. This is a good number of ratings per user, providing a rich amount of information for a collaborative filtering recommender, avoiding the cold-start problem. Similarly, to the previous dataset, looking at the Distribution of Movie Ratings, can be seen that 32 % of movies had not received more than 10 ratings, with 21 % of receiving more than a 100 ratings. The most rated movie in this dataset was The Matrix (1999), with 1670 ratings.

[6]http://ir.ii.uam.es/hetrec2011
[7]Internet Movie Database, http://www.imdb.com
[8]Rotten Tomatoes, movie critic reviews, http://www.rottentomatoes.com



(a) Distributions of movie ratings.



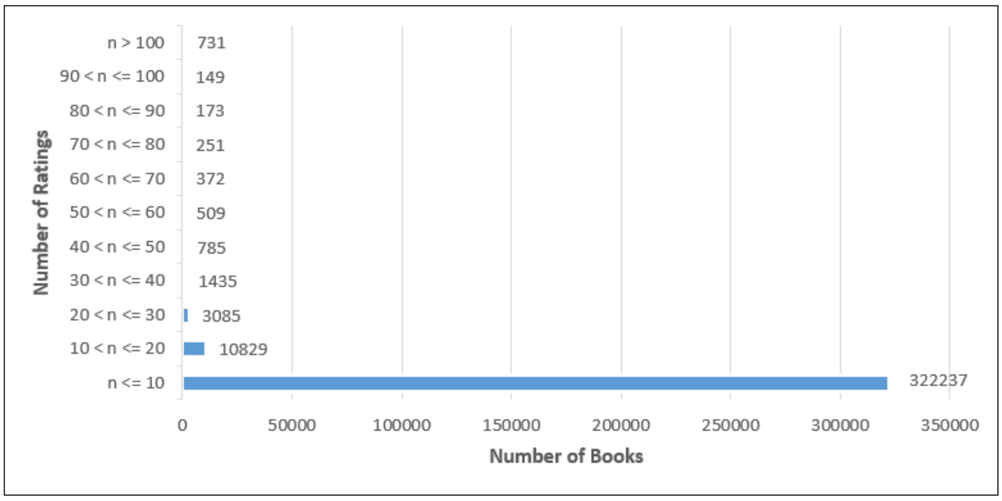
(b) Distributions of user ratings.

Figure 5.4: Distributions of Movie ratings and User ratings in HetRec2011 dataset.

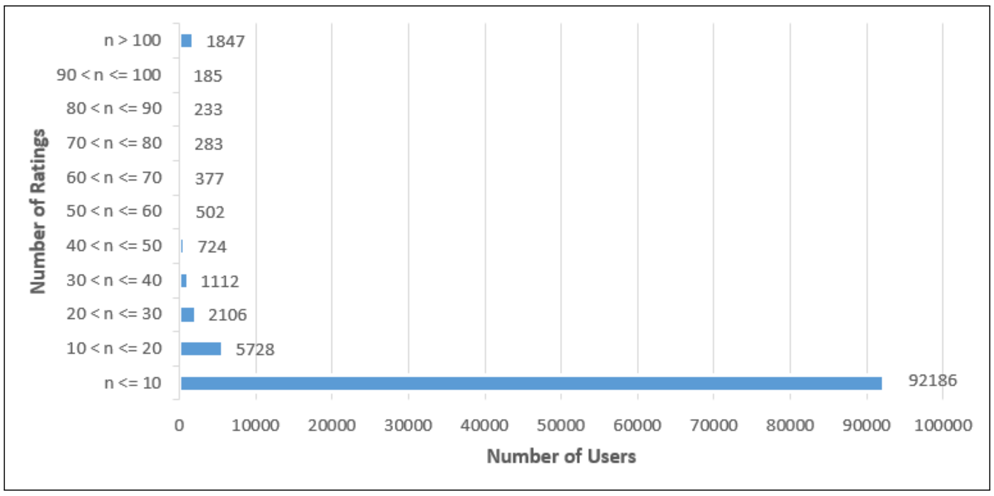
5.1.1.3 Book-Crossing. The Book-Crossing (BX) dataset is a result of a 4-week crawl (August / September 2004) from the Book-Crossing community[9]. It contains 278,858 users (anonymized but with demographic information) providing 1,149,780 ratings (explicit / implicit) about 271,379 books. This dataset was important because it utilizes another domain from the previous datasets. Each Book has the ISBN, Title, Author, Year of Publication, Publisher and cover image.

In the Figure 5.5 presents the details on the distribution of users and books ratings. This dataset exhibits a huge problem of sparsity, with the majority of users (88%) rating less than 10 books. If we applied the same constraint of the previous, where the users

[9]http://www.informatik.uni-freiburg.de/ cziegler/BX/



(a) Distributions of movie ratings.



(b) Distributions of user ratings.

Figure 5.5: Distributions of Movie ratings and User ratings in Book-Crossing dataset.

rated at least 20 items, only 7% of the dataset would be usable. The distribution of ratings in books exhibits the same problem, with 95% of books being rated by less than 10 users.

5.1.2 Experimental Setup and Evaluation Metrics

During this work we use two types of evaluation protocol. For all tests, we did a 10-fold-cross-validation. Given the data set, we randomly divided it into n subsets of the same size, with n as 10, and for each sample we use $n-1$ of these subsets of data for training and the rest for testing. The training set t_r was used to test the proposed assembly and test system T_e randomly split an item for each user to create the truth set H. The remaining items form the set of observable O is used to test the unimodal algorithms.

In the Experiments 1, 2 and 4 we performed the evaluation using the standard protocol, where all items are considered. This is a classical methodology used by the research community with regard to recommender systems evaluation (RICCI; ROKACH; SHAPIRA, 2011). For the evaluation of the Experiment 3, we used the *All But One* (BREESE; HECKERMAN; KADIE, 1998) protocol for the construction of the ground truth. The *All But One* protocol works in the follow way:

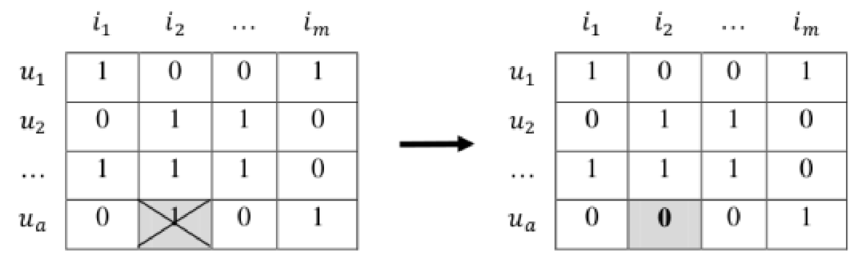


Figure 5.6: Example of hiding items in the *All but One* protocol (MORAIS, 2012).

The total number of users is divided into two subsets: training and testing. Initially, it is selected all users that contain at least two items evaluated. In this set, pairs are then removed for testing and the remainder comprises the training set. This step is required to ensure that all users who are in the test set also belong to the training. Now, from the test set of n items belonging to the active user, one item previously rated by the user is randomly selected to be hidden (set to 0) , causing that item pass to be unknown to the user, as can be seen in Figure 5.6. The relevant items are then hidden and marked as unknown by the protocol. For this reason, for a user to be active, it must have at least two initial items: one for hiding and another to train the model.

To assess the outcomes of the Recommender System we use the evaluation metrics, AUC (Area Under the ROC curve) Precision and Mean Average Precision (MAP) (VOORHEES; HARMAN, 2005). Then, we compute Precision and Mean Average Precision as follows:

Precision calculates the percentage of recommended items that are relevant. This metric is calculated by comparing, for each user in the test set T_e, the set of recommendations R that the system makes, given the set of observables O, against the set H:

$$Precision(T_e) = \frac{1}{|T_e|}\sum_{j=1}^{|T_e|}\frac{|R_j \cap H_j|}{|R_j|}. \tag{5.1}$$

Mean Average Precision computes the precision considering the respective position in the ordered list of recommended items. With this metric, we obtain a single value accuracy score for a set of test users T_e:

$$MAP(T_e) = \frac{1}{|T_e|} \sum_{j=1}^{|T_e|} AveP(R_j, H_j), \tag{5.2}$$

where the average precision (AveP) is given by

$$AveP(R_j, H_j) = \frac{1}{|H_j|} \sum_{r=1}^{|H_j|} [Prec(R_j, r) \times \delta(R_j(r), H_j)], \tag{5.3}$$

where $Prec(R_j, r)$ is the precision for all recommended items up to ranking r and $\delta(R_j(r), H_j) = 1$, iff the predicted item at ranking r is a relevant item ($R_j(r) \in H_j$) or zero otherwise.

AUC, or area under the ROC curve (Receiver Operating Characteristic curve), is a metric for binary classification. A ROC curve is a plot of true positive rate vs. false positive rate as the prediction threshold sweeps through all the possible values. The area under this curve has a property that specifies the probability that, when we draw one positive and one negative example at random, the decision function assigns a higher value to the positive than to the negative example (BICKEL, 2006). The best possible value is 1, and any non-random ranking that makes sense would have an AUC > 0.5. 1 means that for each item, the classifier was able to correctly identify the class. 0.5 means that the classifier get as many true positives as false positives, exactly as a random classifier. Similarly, a 0 value represents that the classifier incorrectly classifies every item.

For the experiments that uses the GA Weighting Strategy, which utilizes a Genetic Algorithm (GA), the following parameters were used: A population of size 40 with 90 generations; a crossover probability of 80% ;and, a mutation probability of 8%. This is a very poorly optimized Genetic Algorithm, as usually a much higher number of generations is needed for the GA converge to the optimal; however, we were to the size of our dataset, and our preference to utilize a real world scenario, where the computations needed to be done in a timely manner, these moderated parameters were used.

In this work we used Precision@N and MAP@N, where N took values of $1, 3, 5$ and 10 in the rankings returned by the system. For each configuration and measure, the 10-fold values are summarized by using mean and standard deviation. In order to compare the results in statistical form in Experiment 3 and 4, we apply the two-sided paired t-test with a 95% confidence level (MITCHELL, 1997).

5.2 EXPERIMENT 1: NAÏVE COMBINATION IN MOVIELENS

This was an initial evaluation, to validate the hypothesis that metadata combination could improve the performance of Recommender Systems. It was used the MovieLens 100k dataset with some dataset extensions from IMDB. The *Naïve Combination Strategy* was used for combining five different types of metadata. We compared the combination of five different types of metadata: actors, directors, genres, keywords and writers using the recommendation algorithms previously described in Section 2. These algorithms were implemented in the *MyEnsembleLite*. To measure the accuracy of recommendations, we used the Mean Average Precision (MAP).

The tests were executed with our augmented database of MovieLens 100k. Each user rated at least 20 movies freeing us from the cold start problem. The metadata were linearly combined in pairs by concatenating the attributes in the final matrix. As a result, a total of 10 combinations was generated and compared.

Table 5.1: Algorithms MAP scores using the Naïve Combination strategy.

Metadata	**BPR-Linear**	**MABPR**	**BPR-Mapping**	**MostPopular**
ACTOR	0.04221	0.25314	0.2552	0.04115
ACTOR-DIRECTOR	0.04352	0.25154	0.25429	0.04337
ACTOR-GENRE	0.0406	0.25335	0.25438	0.01727
ACTOR-KEYWORD	0.04369	0.25187	0.25605	0.02409
ACTOR-WRITER	0.04433	0.25312	**0.25705**	0.04392
DIRECTOR	0.03959	0.252	0.25489	**0.06123**
DIRECTOR-GENRE	0.04476	**0.25531**	0.25121	0.02714
DIRECTOR-KEYWORD	0.05096	0.25388	0.25093	0.02387
DIRECTOR-WRITER	0.04877	0.25354	0.25315	**0.05441**
KEYWORD	**0.0554**	0.25153	0.25122	0.0213
GENRE	0.03915	0.25494	0.25083	0.03401
GENRE-KEYWORD	0.05295	**0.2553**	0.252	0.01686
WRITER	0.04476	0.25171	0.25118	0.0213
WRITER-GENRE	0.04896	0.2519	0.25254	0.02155
WRITER-KEYWORD	0.05106	0.25378	0.25388	0.02439

After executing the algorithms for each possible metadata combination and with different numbers of latent factors in the range [10..100], we compared the best MAP scores in each algorithm and each metadata. The goal was to infer the most suitable in each case. The obtained results are listed in the Table 5.1. The results of the algorithms are also shown in the Figure 5.7.

In Experiment 1 we also utilized a different Recommender algorithm as a baseline called *MostPopularByAttributes*. This is a simple algorithm similar to the "Same artist -greatest hits" baseline presented on McFee et al. (MCFEE et al., 2012). It recommends a ranked item list ordered by popularity, considering attributes that the user had seen previously, followed by the remaining items also ordered by popularity. For instance, if the user had listened only to Rock music, it will recommend first the most popular Rock songs, followed by other genres.

The algorithms MABPR and BPR-Mapping achieved better MAP results than the others algorithms, due to the fact they are based on matrix factorization. These two algorithms generated a MAP score greater than 0.250 in all tested cases, while the others reached a maximum of 0.06. Further, we compared those two best performing algorithms with both the individual and combined metadata as seen in Figure 5.8.

In particular, the best results were achieved when the BPR-Mapping algorithm was combined with the *actor-writer* metadata, followed by *actor-keyword* or when the MABPR

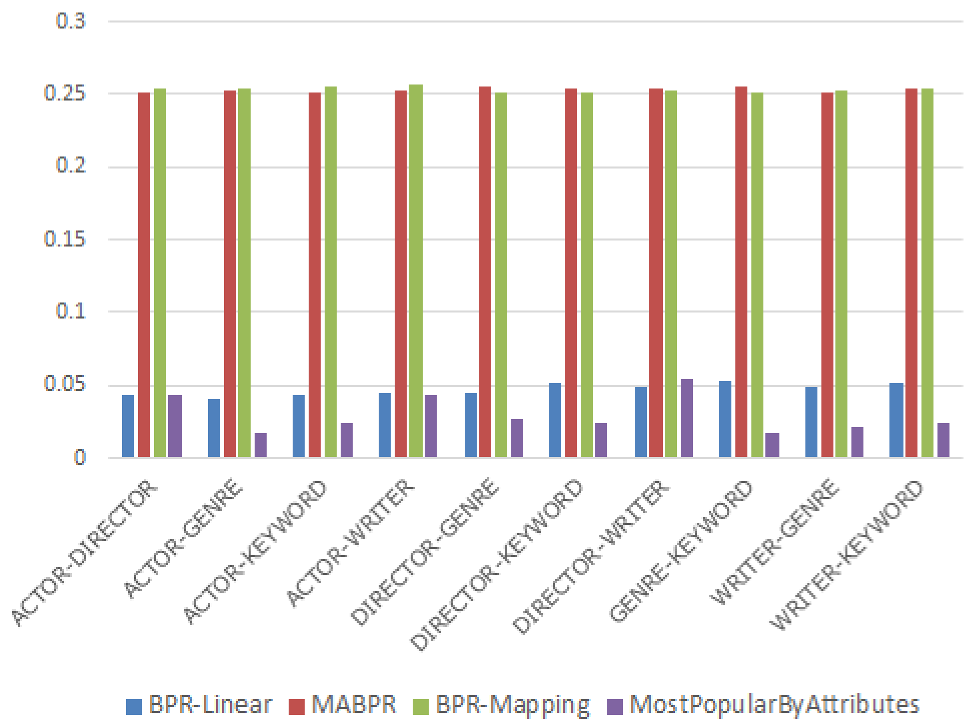


Figure 5.7: Comparison among algorithms.

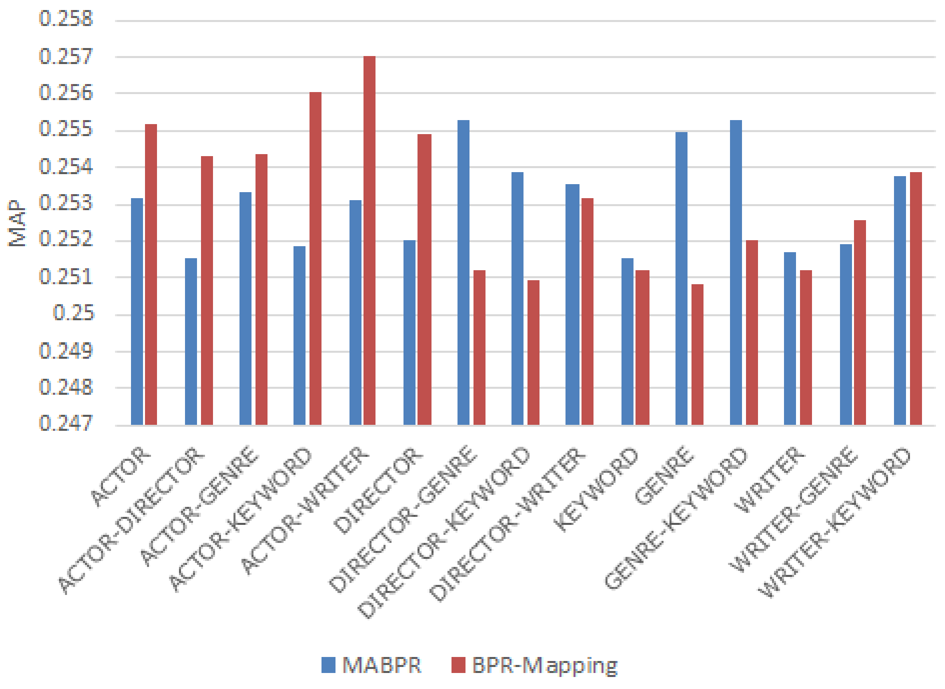


Figure 5.8: Comparison between MABPR and BPR-Mapping.

algorithm was combined with the *genre-keyword* metadata. It is noted that combined metadata performs better with MAP score of 0.25705, 0.25605 and 0.2553 respectively.

Regarding the analyzed metadata, none of the algorithms returned the best recommendation for all tested cases. As shown, the results are balanced and every algorithm has a specific metadata that produces a better score. This occurs because each method has its own purposes. For example, the MostPopularByAttributes was originally proposed for recommending popular songs from an artist that the user already liked (MCFEE et al., 2012). Thus, we expect that the entities *directors* and *actors* to produce a better result over other metadata types in this algorithm. In both BPR-Linear and MostPopularByAttributes combining multiple metadata produced a worse performance.

The results also indicate that for each single metadata it is possible to combine with another metadata and produce a higher score compared to using it individually. However, combining metadata do not always improves the performance. *Actor* is the top performing metadata using the BPR-Mapping, and combining with *writer* or *keyword* metadata lead to a better MAP score. On the other hand, combining with *director* or *genre* impacts negatively the score. Another interesting aspect is that combining the two best individual metadata do not produce the best-combined recommendation score. In fact, the top performing MABPR combined metadata is composed of the best individual metadata (*genre*) and the worse (*keyword*).

In addition, the metadata with the best recommendations for one algorithm is not equivalent in other algorithm. This behavior is observed by analyzing the MAP scores among the tested algorithms. An example is the fact that *actor-keyword* is the metadata which returned the highest MAP in the algorithm BPR-Mapping with MAP 0.25605, and the *genre-keyword* is the metadata which returned the highest MAP score in the algorithm MABPR with MAP 0.2553. Thus, it is possible to note that some algorithms work better when using more general descriptions (e.g. *genres/keywords*), whereas other produce better results when using more specific descriptions (e.g. *actor/writer*).

Nevertheless, although different metadata vary differently in each analyzed algorithm, we understand that the *genre* metadata has a bigger relevance than the single *keyword*, as it describes the whole content in general, and not a single subject of the movie. Thus, in cases where the MAP score is too similar, instead of searching a metadata that prevails over all algorithms, we suggest to search for better recommendations.

Finally, we conclude that best recommendations are achieved when we combine multiple metadata. The top performing is the algorithm BPR-Mapping using the *actor-writer* metadata, followed by the algorithm MABPR using the *genre-keyword* metadata.

5.3 EXPERIMENT 2: ENSEMBLE STRATEGIES USING HETREC 2011 2K DATASET

Following the previous experiment, in this evaluation, it was used the more advanced ensemble strategies proposed in Chapter 4 to combine metadata. They were evaluated with combination of five different types of metadata: actors, directors, genres, tags and countries using the recommendation algorithms and the ensemble algorithms previously described. All algorithms were also available in the *MyEnsembleLite*, thanks to the

MyMediaLite library (GANTNER et al., 2011), which provides the needed infrastructure such as matrix factorization algorithms and error measure methods. To measure the accuracy of recommendations, we used the Mean Average Precision (MAP).

All tests were executed with the HetRec 2011 MovieLens 2k dataset (CANTADOR; BRUSILOVSKY; KUFLIK, 2011), composed of 2113 users with 855598 ratings on 10197 movies, including the relation between 20 movie genres, 4060 directors, 95321 actors, 72 countries and 13222 tags.

The three matrix factorization algorithms were evaluated using a fixed latent factor of 10, and as a preliminary run, they achieved the highest MAP score for the majority of cases. The Genetic Algorithm (GA) uses a population of size 40 with 90 generations, a crossover probability of 80% and a mutation probability of 8%. Usually a higher number of generations is used for convergence; however, due to the size of our dataset, a moderated number was used.

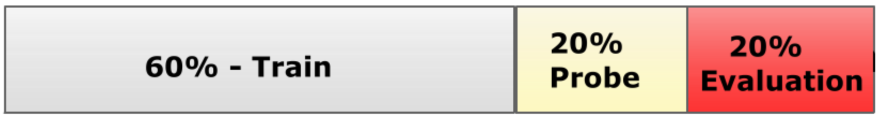


Figure 5.9: Dataset Split.

We split the dataset randomly in an 80:20 proportion and used as training and evaluation respectively. However, due to the need of a probe run in some of the ensemble strategies presented in Section 5, 25% of training dataset was split again to the probe run, resulting in a 60:20:20 split as illustrated in Figure 5.9. It is important to note that during the evaluation the algorithm is trained with the full training dataset. To summarize, the ensemble was created with an algorithm trained with the 60% dataset and evaluated with the 20% probe dataset, later with the ensemble created, the algorithm was trained again, this time with the full 80% training dataset and evaluated with the evaluation dataset.

Finally, we executed for each algorithm, eight different runs, resulting in total 32 runs. The first five are runs where the algorithm is trained with one of the metadata individually, and are used as baseline for performance evaluations of three ensemble strategies. Thus, we compared the best MAP scores in each algorithm and each metadata. The obtained results are listed in the Table 5.2.

The results indicate the following: With ensembles strategies, it was possible to significantly improve the baseline results of using a single metadata. The improvement level was between 1.5% and 7.2%. These improvements were significant as increasing the MAP is a difficult problem, and every increment in MAP is difficult to achieve. Surprisingly, the improvement level was similar among simpler and the complex models, with approximately 7% of improvement discarding the *Tags* metadata outlier in BPR-Linear algorithm as shown in Figure 5.12. The GA Weighting strategy generated the best recommendation for three of the four algorithms, and had the MABPR as the best algorithm to use. The values returned by the algorithms MABPR (Figure 5.10) and BPR-Mapping

Table 5.2: Algorithms MAP scores of Ensemble strategies using the HetRec 2011 2k dataset

Metadata	**MABPR**	**BPR-Mapping**	**BPR-Linear**	**MostPopular**
Genre	0.1671	0.1662	0.0190	0.0186
Tags	0.1704	0.1682	0.1486	0.0155
Directors	0.1687	0.1670	0.0303	0.0504
Actors	0.1675	0.1646	0.0254	0.0202
Countries	0.1671	0.1662	0.0250	0.1051
Most Pleasure	0.1695	0.1670	0.1444	**0.1124**
Best of All	0.1761	0.1729	0.1217	0.1081
GA Weighting	**0.1838**	**0.1803**	**0.1510**	0.0598
Improvement	7.2817%	7.1981%	1.5674%	6.8860%

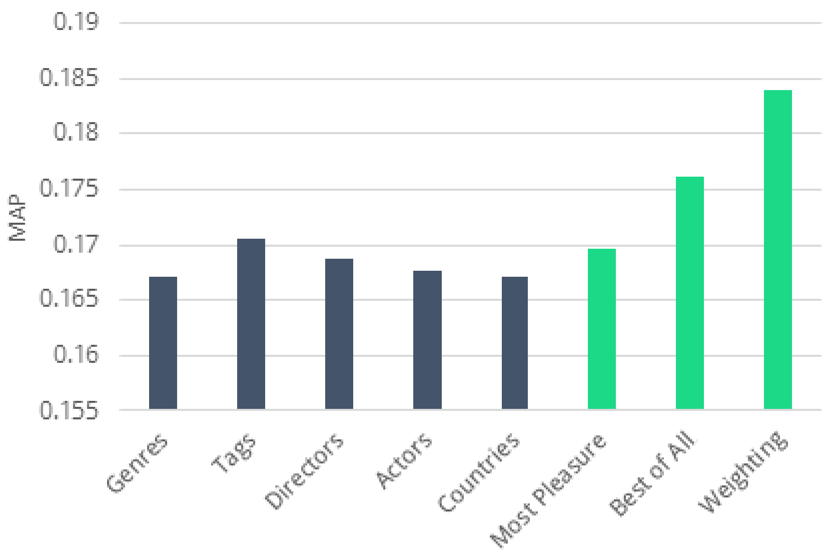


Figure 5.10: MAP score results using the MABPR algorithm. The first five bars are the results for the MABPR recommender algorithm using only one type of metadata, whereas the last three bars are the results for the proposed ensemble algorithms.

(Figure 5.11) are generally much better than those achieved by the other two algorithms. This is due to the fact that they are state-of-art recommender algorithms. They generated very similar results with a maximum MAP of 0.1838 for MABPR and 0.1803 for BPR-Mapping. On the other hand, the BPR-Linear achieved a lower MAP, of 0.1510 . Probably because it is a simpler algorithm .

Indeed, none of the evaluated ensemble method was optimal for all given scenarios. Consequently, one should look for the (base model, ensemble) pair that achieves the best results for the dataset at hand. However, the GA Weighting ensemble strategy showed as the most effective on three of four scenarios and may be considered as a good candidate

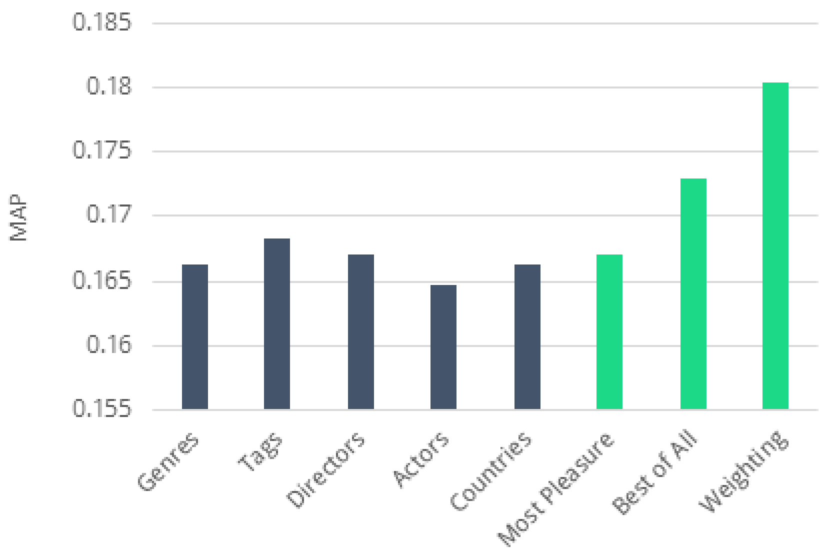


Figure 5.11: MAP score results using the BPR-Mapping algorithm. The first five bars are the results for the BPR-Mapping recommender algorithm using only one type of metadata, whereas the last three bars are the results for the proposed ensemble algorithms.

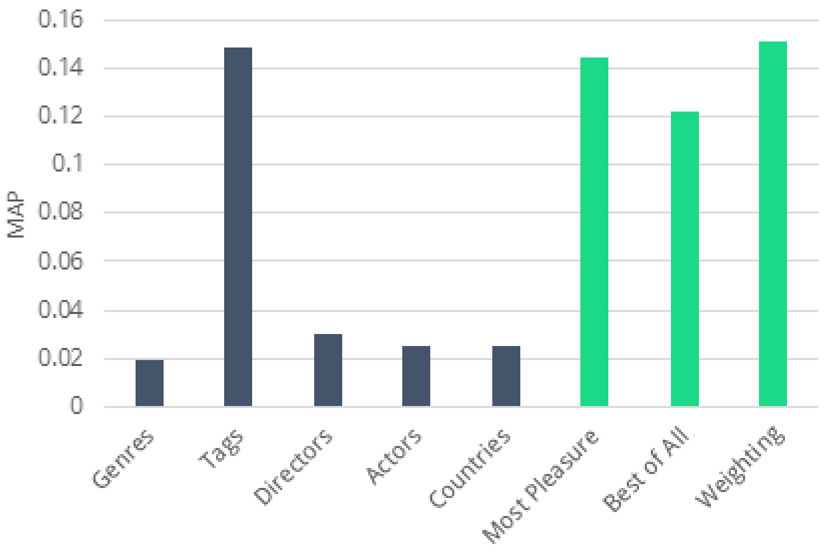


Figure 5.12: MAP score results using the BPR-Linear algorithm. The first five bars are the results for the BPR-Linear recommender algorithm using only one type of metadata, whereas the last three bars are the results for the proposed ensemble algorithms.

to implement in a real world scenario. This is because this strategy uses all metadata to make predictions, and it assigns different weights to the most relevant metadata according to the taste of each individual user.

While the GA Weighting strategy got promising results, the other two strategies should also be considered depending on the scenario. For instance, the MostPleasure strategy is the simplest and straightforward to implement, with a very low overhead as a probe run is not needed. Moreover, it got a good performance improvement on the weaker algorithms, and almost did not affect negatively the more complex algorithms. Likewise, the Best of All did produce an even higher improvement, and although it needs a probe run, it does require the GA weight optimization, an expensive step in the process.

Considering only the metadata individually, the *Tags* is the metadata that returned the best recommendations for three of the four analyzed algorithms, and in BPR-Linear yielded a similar result compared to the ensemble algorithms. This is probably because the Tags contains a more diverse set of information, and, sometimes, may even simulate a combination of metadata. The tags referenced information such keywords, actors, genres, directors, producers. Recommending movies based on a combination of metadata, as seen in the previous experiment, generates better combinations than with a single metadata.

Finally, we conclude that ensemble algorithms significantly improved the recommender prediction performance, with the GA Weighting strategy standing out with higher performance on most of the scenarios. Additionally, the algorithm MABPR obtained the best for the tested data.

5.4 EXPERIMENT 3: ENSEMBLE STRATEGIES USING HETREC 2011 2K DATASET TO COMBINE INTERACTIONS.

The previous experiments were all made using the movie domain for combining multiple metadata. However we wanted to verify if the proposed ensemble techniques could be using for another objective. In this experiment it was compared the proposed ensemble strategies with another work with a similar objective. However, the objective of the ensemble was not to combine metadata, but to combine different types of user interaction (historic, tags and explicit rating)

In order to evaluate the performance of the ensemble strategies, we used again the HetRec MovieLens 2*k* dataset. As explicit information, we used the ratings that users assigned to items, and as implicit information, we considered: i) whether a user tagged an item or not; and ii) the history of visited items, which is simulated by boolean values (visited or not) generated by the ratings and tagging activities. For implicit data interactions (history and tags), we used the BPR-MF, an implementation of the Bayesian Personalized Ranking (BPR)(GANTNER et al., 2010), a generic framework for optimizing different kinds of models based on training data containing only implicit feedback information. For explicit interactions (ratings), we used SVD++ (KOREN, 2008), also implemented originally in the MyMediaLite library.

For evaluation, we divided the base dataset into two sets, 80% for training and 20% for testing, where the training set is used to run the isolated algorithms and predict weights for each pair of algorithms (simulate the real-time interaction from the user); and where the test-set is used with the *All but One* protocol to evaluate the approaches.

Table 5.3 shows the results of this evaluation, considering single interactions and ensembles. From the results we can see that the best performing single interaction was

Table 5.3: Algorithms' performance in Precision@1, 3, 5 and 10.

	Prec@1	Prec@3	Prec@5	Prec@10	Map@5	Map@10
Historic	0.000047	0.000047	0.000037	0.000033	0.000104	0.000120
Tags	0.002082	0.002035	0.001874	0.001628	0.004569	0.005456
Ratings	0.000094	0.000047	0.000037	0.000018	0.000119	0.000119
BestOfAll	0.001988	0.001988	0.001845	0.001614	0.004458	0.005345
GA	0.001988	0.001971	0.001845	0.001614	0.004441	0.005334
MostPleasure	0.000047	0.000047	0.000037	0.000033	0.000104	0.000120
BPR Learning	**0.002366**	**0.002366**	**0.002271**	**0.001845**	**0.005229**	**0.006044**
Improvement	12.0 %	12.1 %	21.1 %	13.3 %	14.4 % %	10.7 %

the *tags*, in this way, we compared the ensemble performance to this best performing interaction. As seen, the *BPR Learning* strategy achieved statistically better results than the baseline, as proven by the t-student analysis (with $p < 0.05$) in Table 5.4. Figure 5.13 illustrates the algorithms' precision@1, 3, 5 and 10 using the *All But One* Protocol.

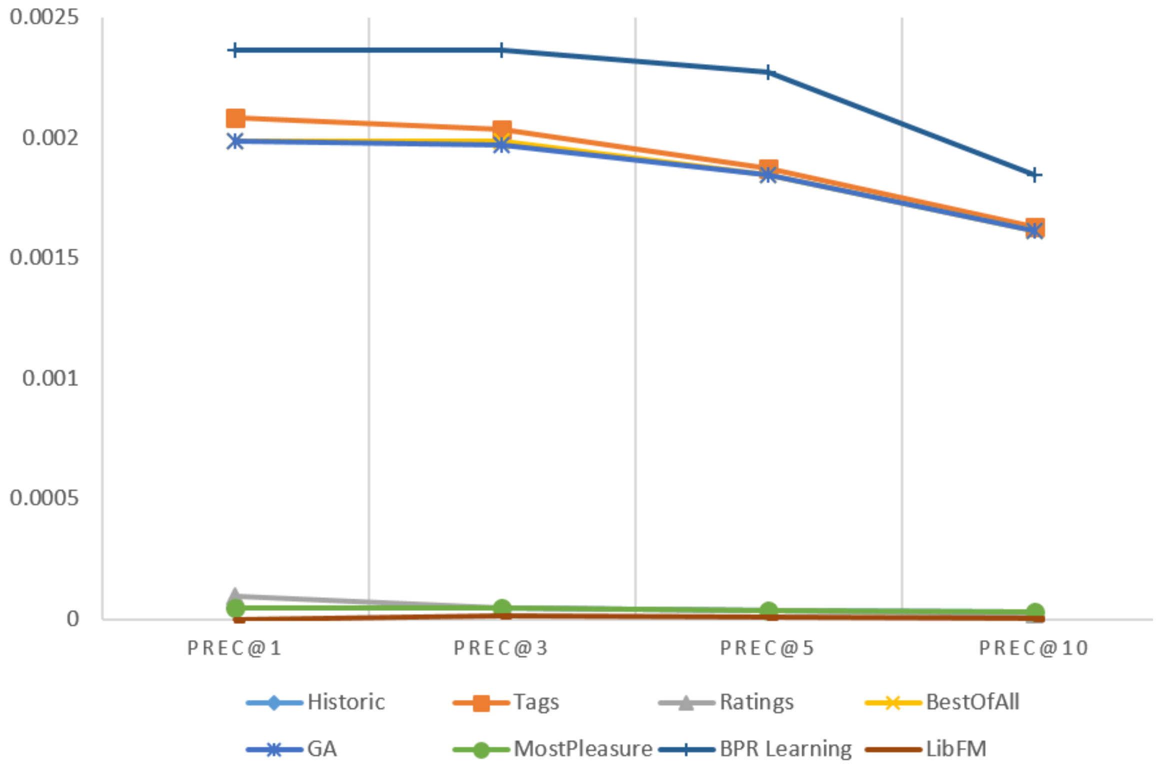


Figure 5.13: Comparative of Precision@1, 3, 5 and 10 using All But One Protocol.

The results indicate that we were able to significantly improve the baseline results of using a single interaction in our work. The improvement level was between 10.7% and 21.1% compared to the best performing interaction. These improvements were significant

Table 5.4: t-Test comparing MAP@5 using BPR Learning with Tags.

	BPR Learning	Tags
Mean	0.005115	0.004569
Variance	3.16E-07	1.07E-07
Observations	10	10
df	14	
t Stat	-2.65099	
P(T<=t) one-tail	0.009496	
t Critical one-tail	1.76131	
P(T<=t) two-tail	0.018992	
t Critical two-tail	2.144787	

as increasing the MAP and precision is a difficult problem, and every increment in MAP is difficult to achieve.

Surprisingly, the *tags* interaction got considerably higher scores than others did for all tested metrics. This scenario is very interesting for real-world applications. Not all companies can afford a skilled engineer to analyze what interaction can provide the best performance and end up using a public available recommender library with an interaction chosen empirically.

The BPR Learning Ensemble strategy was optimal for all given scenarios since it uses all metadata to make predictions, and it assigns different weights to the most relevant metadata according to the taste of each individual user. On the other hand, the MostPleasure strategy achieved the lowest performance among the ensemble strategies.

However, the Weighting ensemble and Best of All strategies obtained a good performance, close to the best performing interaction. The Best of All strategy is simple to implement and do not require weight optimization, an expensive step in the process required for BPR Learning Ensemble and GA Ensemble. Alternatively, GA ensemble does requires a weight optimization step, but as it uses a Genetic Algorithm, one can manually set the parameters and set a trade off of speed or performance.

The overall results obtained and described in this work are small because of the Sparsity and evaluation protocol used in the experiments. The *All But One* protocol hides one item from each user in the test set and considers it as the ground truth. As we are recommending top N items, the precision and MAP will decrease because the system thinks there are N relevant items, although the protocol has set only the hided item as relevant. The high sparsity presents as another challenge to provide valuable recommendations, as many users had not rated any movies, only tagged it. In this case, the rating prediction cannot be made. Another issue is that the rating rank is built using the rating predictions in a decreasing order from the SVD++ algorithm and in the dataset can have items with a low score, lowing the metrics related to this interaction as the test dataset is generated randomly. In this way, it is important to rely only on the differences among the approaches, and we managed to increase the results of our proposal when compared to the baselines.

Finally, we conclude that ensemble algorithms significantly improved the recommender prediction performance, with the BPR Learning strategy standing out with higher performance on most of the scenarios.

5.5 EXPERIMENT 4:

To validate our finds and an experiment in a different domain was needed in order to verify if the proposed ensemble techniques generalize or were specific for the movies domains. This experiment is then executed with the Book-Crossing dataset, containing 278,858 users providing 1,149,780 ratings (explicit / implicit) about 271,379 books. This dataset was important because it utilizes another domain from the previous datasets. Each Book has the ISBN, Title, Author, Year of Publication, Publisher and cover image. Because this dataset was very sparce, it was then selected only books and users which at least 20 ratings. The final dataset consisted of 7.485 users with 253.902 ratings in 3156 books.

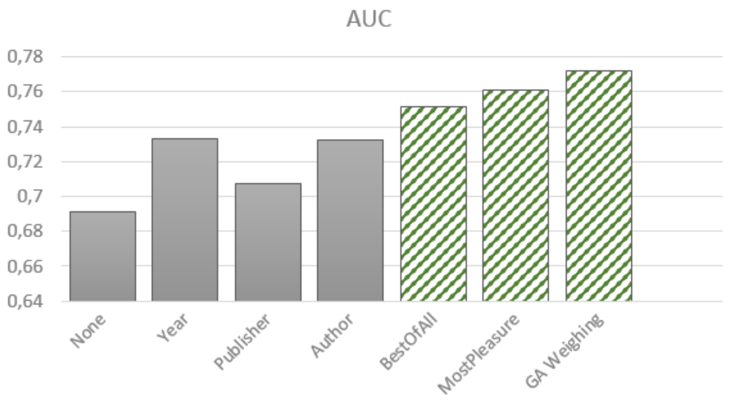
In this experiment we utilized only the BPR-Mapping algorithm to generate the personalized raking, it was utilized with a fixed latent factor of 10 as in a preliminary run, it achieved the highest MAP score for the majority of cases. The Genetic Algorithm (GA) uses a population of size 40 with 90 generations, a crossover probability of 80% and a mutation probability of 8%. We also split the dataset randomly in an 80:20 proportion and used as training and evaluation respectively, with 25% of training dataset for the probe run, resulting in a 60:20:20 split as previously illustrated in Figure 5.9. A 10-fold-cross-validation was utilized.

Table 5.5: Algorithms' performance in Precision@5 AUC and MAP.

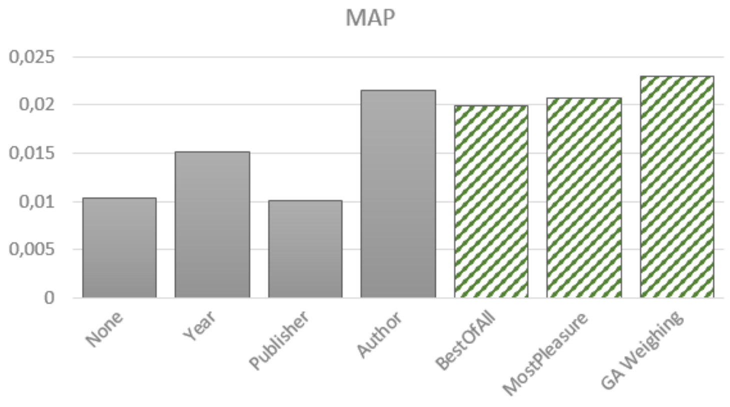
	Prec@5	**AUC**	**MAP**
None	0,01441	0,691	0,01035
Year	0,01848	0,73260	0,01514
Publisher	0,01441	0,70716	0,01014
Author	**0,01842**	**0,73265**	**0,02147**
BestOfAll	0,0211	0,75127	0,01988
MostPleasure	0,01987	0,76106	0,02071
GA Weighing	**0,02238**	**0,77189**	**0,02292**
Best Improvement	21,4%	5,3%	6,7 %

Finally, we executed the algorithm with no metadata, and with: Year; Publisher and Author metadata, and used them as baseline for performance evaluations of three ensemble strategies. The obtained results are listed in the Table 5.5.

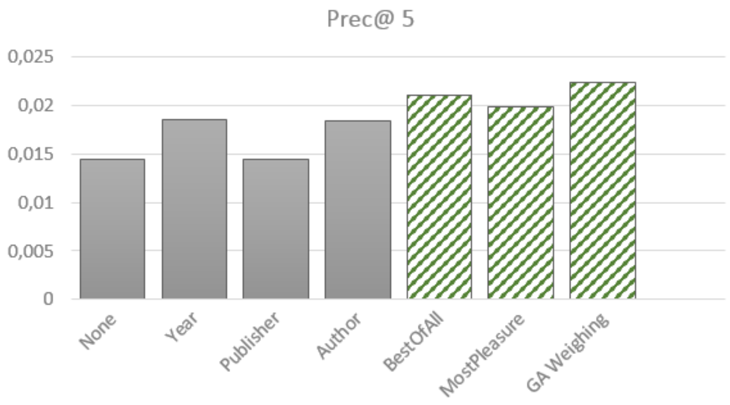
The results indicate with ensembles strategies; it was possible to significantly improve the baseline results of using a single metadata. The improvement level was between 5.3% and 21.4%. In the Figure 5.14, it can be seen the performance comparative of different metadata and ensemble strategies for the AUC, MAP and Prec@5. The overall scores were lower than in the previous Experiments, probably because this dataset is much sparser than the movie dataset used in the previous experiments.



(a) AUC scores



(b) MAP scores



(c) Prec@5 scores

Figure 5.14: Experiment 4 Algorithms' performance in AUC, MAP and Prec@5

The GA Weighting ensemble strategy achieved statistically better results than the baseline, as proven by the t-student analysis (with $p < 0.05$) in Table 5.4. It was the most effective on all scenarios, and may be considered as a good candidate to implement in a real world scenario. This is because this strategy uses all available metadata to make

Table 5.6: t-Test comparing the AUC using GA Weighing with Year.

	GA Weighing	Year
Mean	0.77189	0.732693
Variance	0.00058	0.000237
Observations	10	10
df	15	
t Stat	-4.33783	
P(T<=t) one-tail	0.000293	
t Critical one-tail	1.75305	
P(T<=t) two-tail	0.0000586	
t Critical two-tail	2.13145	

predictions, and it assigns different weights to the most relevant metadata according to the taste of each individual user. While the GA Weighting strategy got the highest performance, the another strategy should also be considered depending on the scenario, the MostPleasure strategy. It was more performant than any of the single metadata in all metrics, and as previously discussed, the MostPleasure strategy extremely straightforward to implement, with a very low overhead as a probe run is not need, and very fast to execute. On the other hand, the Best of All, although got a higher score than any single metadata in two of three scenarios, it was worse in a single case – in the MAP metric where the Author metadata got a higher score. MostPleasure may be a better option as it got higher scores and does not need a probe run, an expensive step, differently from the Best of All strategy.

Considering only the metadata individually, the *Author* is the metadata that returned the best recommendations for all analyzed scenarios. This is somehow intuitive, as often users become fans of a specific author and they usually release books with similar themes. Another interesting point is that the improvement of using ensembles were higher in MAP and Prec@5 metrics than in AUC, one reason to justify this could be because during the development of the ensemble strategies, that target was optimizing the MAP, in fact, in GA Weighing the Fitness function optimizes for MAP.

Finally, we conclude that ensemble algorithms significantly improved the recommender prediction performance, with the GA Weighting strategy standing out with higher performance on most of the scenarios. Additionally, comparing the performance, Author is the most significant metadata for the tested data.

5.6 DISCUSSION

Experiment 1 evaluated four different recommender algorithms and utilized the *Naïve Combination Strategy* to combine movie metadata to generate recommendations of movies. These algorithms were tested with five types of metadata and combining all pair combinations of metadata without repetition in order to infer which one achieves better results according to MAP measure. After comparing the metadata with four different algorithms, we can conclude that combining multiple metadata can improve the performance and the best algorithms in our tests are MABPR and BPR_Mapping, as all the tested metadata achieves the best results with them. In addition, using actor combined with writer metadata in BPR_Mapping algorithm produces better recommendations than other types of metadata, and genre combined with keyword produces the best recommendations when using MABPR algorithm.

Evolving the previous experiment, in experiment 2 was evaluated three different strategies that do not require modification of the recommender algorithm, namely Most Pleasure, Best of All and Genetic Algorithm Weighting. The considered recommender algorithms did not take advantage of multiple item metadata and our ensemble algorithm was able to enable those recommenders to take advantage of this metadata. Most Pleasure, the simplest strategy, consisted of combining the predictions based on score. Best of All determined a single metadata that was more preferred for a user, and finally the Weighting strategy uses multiple metadata and weights them with a Genetic Algorithm that optimizes the MAP. Empirical evaluation showed a considerable MAP improvement between 1.5% and 7.2% when using the ensemble algorithms, with the Weighting strategy producing the best recommendation for the majority of scenarios. These encouraging results indicate that ensemble algorithms can be used to enhance the recommenders' algorithms with multiple metadata.

In Experiment 3 was evaluated four ensemble strategies to unify different types of feedback from users when consuming content in order to provide better recommendations. All ensemble strategies utilized in the previous experiment was used, with the addition of *BPR Learning* an ensemble strategy that uses LearnBPR to optimize the weights related to AUC. The considered recommender algorithms did not take advantage of multiple types of interactions and the evaluated ensemble algorithms were able to enable those recommenders to take advantage of all interactions. The experiments were executed with the HetRec2011 movielens 2k dataset and the results show the effectiveness of combining various types of interactions in a single model for recommendation using ensemble learning. Our evaluation showed a considerable MAP improvement between 10.7% and 21.1% when using the ensemble algorithms, with the *BPR Learning* producing the best recommendation for the majority of scenarios. These results indicate that the ensemble algorithms could be successfully used for combining multiple types of interactions.

Finally, in Experiment 4, another domain was evaluated in order to verify if the proposed ensemble techniques generalize or were specific for the movies domains. This experiment utilized the Book-Crossing dataset consisting of 7.485 users with 253.902 ratings in 3156 books. was possible to significantly improve the baseline results of using a single metadata. The improvement level was between 5.3% and 21.4% and indicates that the proposed ensemble techniques could be utilized in a Book domain to improve the precision of the recommendation.

5.7 SUMMARY

In this chapter, the main results obtained during development of the experiments were presented, related to the implementation process of the combination of metadata in Recommender Systems. Initially, was presented to the evaluation methodologies employed in the studies describing the tools, datasets, metrics and evaluation protocols utilized. Next, multiples experiments were presented to validate our proposed ensemble techniques.

The studies presented in this chapter show that the proposed techniques were effective in regard to reduce the problem addressed in this work in different application domains. The next chapter presents the final considerations of this work, as well as the contributions and future work.

Chapter 6

CONCLUSION

Now this is not the end. It is not even the beginning of the end. But it is, perhaps, the end of the beginning.
—WINSTON CHURCHILL (The End of the Beginning)

In the previous chapter were presented all the experiments carried out on this work, along with a discussion of the results. This chapter, concludes this work, presenting a summary of the work, contributions, achievements and finally some directions for future work.

6.1 OVERVIEW

Recommender systems have become increasingly popular widely adopted by many sites and are important tools in assisting users to filter what is relevant for them in this complex information world. Hybrid recommenders aim at grouping the benefits of content based and collaborative filtering approaches. The downside of hybrid recommenders which primarily exploit latent factor models are: i) do not consider all the metadata associated to the content, which could provide significant and meaningful information about the user's interests, and ii) usually process only one item attribute missing the exploitation of combination of the metadata available.

With that in mind, this dissertation proposed three ensemble strategies for combining multiple metadata in hybrid Recommender Systems, with the aim of improving the top-performing state-of-art algorithms to leverage the available item metadata with an ensemble of this information in a computationally feasible way. Four experiments were performed using state-of-art datasets and algorithms and the results indicate that we were able to achieve a considerable MAP improvement of up to 21% when using the ensemble algorithms. The experiments are composed of three experiments in the movie domain: i) Naïve combination of metadata; ii) an experiment using all the proposed ensemble techniques in the movie domain iii) an experiment combining different kinds of user interactions, and iv) a final experiment in the Books domain was performed to guarantee that the techniques proposed in this papers generalizes to another domain.

6.2 RESEARCH CONTRIBUTION

During the development of the proposal, our prototype originated three recommendation techniques, each with advantages and disadvantages. Most Pleasure, the simplest strategy, consisted of combining the predictions based on score. Best of All determined a single metadata that was more preferred for a user, and finally the Weighting strategy uses multiple metadata and weights them with a Genetic Algorithm that optimizes the MAP. It was created a tool implementing those techniques. This tool is public-available with an open-source license. Another important contribution of this work, was the study and comparison of the best performing metadata in four different scenarios. The finds of this research was the subject of presentations in local conferences such as SEMCOMP [1], these actions have helped to foster the local research and startup ecosystem.

The main contributions of this research are described as follows:

- **A study on existing solutions**, which can provide the research and professional community an overview of the state-of-the-art algorithms in the field that support multiple metadata.

- **A study on the best performing metadata for movies dataset**, specifying which metadata provides best increase in precision for the movie domain. This information is valuable for researchers to further create novel tools and algorithms.

- **Three ensemble strategies** that can be used to improve the performance of existing Recommender Systems when using multiple metadata.

- **An open-source application implementing the techniques proposed in this work.** The MyEnsembleLite tool is a fully-featured, publicly available open-source software implementing the techniques proposed.

6.2.1 Published Papers

This section presents the published and submitted papers resulting from this work :

- **Personalized Ranking of Movies: Evaluating Different Metadata Types and Recommendation Strategies, Revista de Sistemas e Computação, 2014**

  Abstract: This paper proposes a study and comparison among a variety of metadata types in order to identify the most relevant pieces of information in personalized ranking of movie items. We used four algorithms available in the literature to analyze the descriptions, and compared each other using the metadata extracted from two datasets, namely MovieLens and IMDB. As a result of our evaluation, we found out that the movies' genres and actors are the kind of description that generates better predictions for the considered content-based recommenders.

[1]http://www.semcomp.com.br/

Reference: BELTRAO, R. D. ; CABRAL, B. S. ; MANZATO, M. G. ; DURAO, F. A. . PERSONALIZED RANKING OF MOVIES: EVALUATING DIFFERENT METADATA TYPES AND RECOMMENDATION STRATEGIES. Revista de Sistemas e Computação - RSC , v. 4, p. 53-58, 2014.

- **Personalized ranking of movies: Evaluating different metadata types and recommendation strategies using multiple metadata, BRACIS, 2014**

  Abstract: This paper proposes a study and comparison of the combination of multiple metadata types to improve the recommendation of movie items according to users' preferences. We used four algorithms available in the literature to analyze the descriptions, and compared each other using all the possible combinations of the metadata extracted from two datasets, namely MovieLens and IMDB. As a result of our evaluation, we found out that combining metadata generates better predictions for the considered content-based recommenders.

  Reference: BELTRAO, RENATO DOMPIERI ; CABRAL, BRUNO SOUZA ; MANZATO, MARCELO GARCIA ; DURAO, FREDERICO ARAUJO . Evaluating the Combination of Multiple Metadata Types in Movies Recommendation. In: 2014 Brazilian Conference on Intelligent Systems (BRACIS), 2014, Sao Paulo. 2014 Brazilian Conference on Intelligent Systems, 2014. p. 55.

- **Combining Multiple Metadata Types in Movies Recommendation Using Ensemble Algorithms, WEBMEDIA, 2014**

  Abstract: In this paper, we analyze the application of ensemble algorithms to improve the ranking recommendation problem with multiple metadata. We propose three generic ensemble strategies that do not require modification of the recommender algorithm. They combine predictions from a recommender trained with distinct metadata into a unified rank of recommended items. The proposed strategies are Most Pleasure, Best of All and Genetic Algorithm Weighting. The evaluation using the HetRec 2011 MovieLens 2k dataset with five different metadata (genres, tags, directors, actors and countries) shows that our proposed ensemble algorithms achieve a considerable 7% improvement in the Mean Average Precision even with state-of-art collaborative filtering algorithms.

  Reference: CABRAL, B. S. ; DOMPIERI BELTRAO, RENATO ; GARCIA MANZATO, MARCELO ; ARAÚJO DURÃO, FREDERICO . Combining Multiple Metadata Types in Movies Recommendation Using Ensemble Algorithms. In: the 20th Brazilian Symposium, 2014, João Pessoa. Proceedings of the 20th Brazilian Symposium on Multimedia and the Web - WebMedia '14, 2014. p. 231.

- **Evaluating Multiple User Interactions for Ranking Personalization Using Ensemble Methods, PENDING SUBMISSION**

  Abstract: The variety of interaction paradigms on the Web, such as clicking, commenting or rating are important sources that help recommender systems to gather accurate information about users' preferences. Ensemble methods can be used to combine all these pieces of information in a post-processing step to generate recommendations that are more relevant. In this paper, we review the application of existing ensemble methods to improve ranking recommendations in the multimodal interactions context. We compared four ensemble strategies, ranging from simple to complex algorithms including Gradient Descent and Genetic Algorithm to find optimal weights. The evaluation using the HetRec 2011 MovieLens $2k$ dataset with three different types of interactions shows that a considerable 7% improvement in the Mean Average Precision can be achieved using ensembles when compared to the most performant single interaction.

6.3 FUTURE WORK

An initial prototype was developed and evaluated in this work. However, we are aware there is room for its continuation and improvement. In this section, some suggestions are presented as a continuation of this research:

- **Implement more complex ensemble strategies and evaluate the algorithms with a higher number of metadata** to verify whether metadata can generate better recommendations. In order to do so, it will be necessary to find a more extensive dataset and evaluate the performance of the algorithms with this increased work.
- **Increase the number of metadata combined**. The highest number of metadata combined in this work was five. An interesting path to follow is verifying if the proposed ensemble strategies could still be efficient with a much higher number of metadata to combine.
- **Further evaluation**. In this work, we presented a case study. A more detailed evaluation is needed by applying the proposed ensemble techniques in other context such friends and products recommendations, in order to provide richer findings.
- **Validate the techniques in a real world application**. Although the tests were evaluated using datasets from real applications, during the implementation and deploy of a real world application, a number of issues and requirements may appear. It is important to be able to apply the proposed techniques in such scenario.
- **Online update of ensemble weights**. The proposed ensemble techniques that utilize weighing require the generation of the entire model when updating information. It would be interesting to iteratively update the weights for saving unnecessary computation.

What we gathered from observing the industry is that ensembles are omnipresent and it is a trump card for the industry. However, they keep the implementation details as a secret, usually because this is very coupled to their business logic. An open study and availability of ensemble techniques could provide as a valuable competitive advantage to small business and startups. In this way, future works should focus on implementing ensemble techniques in real-world systems.

BIBLIOGRAPHY

ADOMAVICIUS, G.; TUZHILIN, A. 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*, v. 17, n. 6, p. 734–749, 2005.

BANGALORE, S.; RAMBOW, O. Corpus-based lexical choice in natural language generation. In: ASSOCIATION FOR COMPUTATIONAL LINGUISTICS. *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*. [S.l.], 2000. p. 464–471.

BAR, A. et al. Improving simple collaborative filtering models using ensemble methods. In: *Multiple Classifier Systems*. [S.l.]: Springer, 2013. p. 1–12.

BELTAO, R. et al. Personalized ranking of movies: Evaluating different strategies using multiple metadata. *BRACIS*, 2014.

BICKEL, S. 2006. (http://www.ecmlpkdd2006.org/challenge.html).

BREESE, J. S.; HECKERMAN, D.; KADIE, C. Empirical analysis of predictive algorithms for collaborative filtering. In: . San Francisco, CA, USA: [s.n.], 1998. (UAI'98), p. 43–52. ISBN 1-55860-555-X. Disponível em: ⟨http://dl.acm.org/citation.cfm?id=2074094.2074100⟩.

BREIMAN, L. Bagging predictors. *Machine learning*, Springer, v. 24, n. 2, p. 123–140, 1996.

BREIMAN, L. et al. Heuristics of instability and stabilization in model selection. *The annals of statistics*, Institute of Mathematical Statistics, v. 24, n. 6, p. 2350–2383, 1996.

BURKE, R. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, Springer, v. 12, n. 4, p. 331–370, 2002.

BURKE, R. Hybrid web recommender systems. In: *The adaptive web*. [S.l.]: Springer, 2007. p. 377–408.

CABRAL, B. et al. Combining multiple metadata types in movies recommendation using ensemble algorithms. *WEBMEDIA*, 2014.

CANTADOR, I.; BRUSILOVSKY, P.; KUFLIK, T. 2nd workshop on information heterogeneity and fusion in recommender systems (hetrec 2011). In: *Proceedings of the 5th ACM conference on Recommender systems*. New York, NY, USA: ACM, 2011. (RecSys 2011).

CASINELLI, P. Evaluating and implementing recommender systems as web services using apache mahout. 2013.

DIETTERICH, T. G. Ensemble methods in machine learning. In: *Multiple classifier systems*. [S.l.]: Springer, 2000. p. 1–15.

DITTERRICH, T. Machine learning research: four current direction. *Artificial Intelligence Magzine*, v. 4, p. 97–136, 1997.

DŽEROSKI, S.; ŽENKO, B. Is combining classifiers with stacking better than selecting the best one? *Machine learning*, Springer, v. 54, n. 3, p. 255–273, 2004.

EKSTRAND, M. D.; RIEDL, J. T.; KONSTAN, J. A. Collaborative filtering recommender systems. *Foundations and Trends in Human-Computer Interaction*, Now Publishers Inc., v. 4, n. 2, p. 81–173, 2011.

FORTES, A. da C.; MANZATO, M. G. Ensemble learning in recommender systems: Combining multiple user interactions for ranking personalization. In: *Proceedings of the 20th Brazilian Symposium on Multimedia and the Web*. New York, NY, USA: ACM, 2014. (WebMedia '14), p. 47–54.

FREUND, Y.; SCHAPIRE, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, Elsevier, v. 55, n. 1, p. 119–139, 1997.

GANTNER, Z. *Supervised Machine Learning Methods for Item Recommendation*. Tese (Doutorado) — Hildesheim, Universita Hildesheim, Diss., 2012, 2012.

GANTNER, Z. et al. Learning attribute-to-feature mappings for cold-start recommendations. In: *Proceedings of the 2010 IEEE International Conference on Data Mining*. Washington, DC, USA: IEEE Computer Society, 2010. (ICDM '10), p. 176–185. ISBN 978-0-7695-4256-0. Disponível em: ⟨http://dx.doi.org/10.1109/ICDM.2010.129⟩.

GANTNER, Z. et al. MyMediaLite: A free recommender system library. In: *Proceedings of the 5th ACM Conference on Recommender Systems*. New York, NY, USA: [s.n.], 2011. (RecSys '11), p. 305–308.

GANTZ, J.; REINSEL, D. The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. 2012.

GOLDBERG, D. et al. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, ACM, New York, NY, USA, v. 35, n. 12, p. 61–70, dez. 1992. ISSN 0001-0782. Disponível em: ⟨http://doi.acm.org/10.1145/138859.138867⟩.

GOLDBERG, K. et al. Eigentaste: A constant time collaborative filtering algorithm. *Inf. Retr.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 4, n. 2, p. 133–151, jul. 2001. ISSN 1386-4564. Disponível em: ⟨http://dx.doi.org/10.1023/A:1011419012209⟩.

GOODRUM, A. Image information retrieval: An overview of current research. *Informing Science*, v. 3, p. 2000, 2000.

GUENTHER, R.; RADEBAUGH, J. Understanding metadata. *National Information Standard Organization (NISO) Press, Bethesda, USA*, 2004.

HALLOWELL, E. M. Overloaded circuits. *Harvard Business Review*, p. 11, 2005.

HAN, J.; KAMBER, M.; PEI, J. *Data mining: concepts and techniques: concepts and techniques*. [S.l.]: Elsevier, 2011.

HILL, W. et al. Recommending and evaluating choices in a virtual community of use. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1995. (CHI '95), p. 194–201. ISBN 0-201-84705-1. Disponível em: ⟨http://dx.doi.org/10.1145/223904.223929⟩.

JAHRER, M.; TöSCHER, A.; LEGENSTEIN, R. Combining predictions for accurate recommender systems. In: *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM, 2010. (KDD '10), p. 693–702. ISBN 978-1-4503-0055-1. Disponível em: ⟨http://doi.acm.org/10.1145/1835804.1835893⟩.

JOHANSSON, P. Madfilm - a multimodal approach to handle search and organization in a movie recommendation system. In: *In Proceedings of the 1st Nordic Symposium on Multimodal Communication*. [S.l.: s.n.], 2003. p. 53–65.

KOREN, Y. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM, 2008. (KDD '08), p. 426–434. ISBN 978-1-60558-193-4. Disponível em: ⟨http://doi.acm.org/10.1145/1401890.1401944⟩.

KOREN, Y. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Trans. Knowl. Discov. Data*, ACM, New York, NY, USA, v. 4, n. 1, p. 1:1–1:24, jan. 2010. ISSN 1556-4681. Disponível em: ⟨http://doi.acm.org/10.1145/1644873.1644874⟩.

KOREN, Y.; BELL, R.; VOLINSKY, C. Matrix factorization techniques for recommender systems. *Computer*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 42, n. 8, p. 30–37, ago. 2009. ISSN 0018-9162. Disponível em: ⟨http://dx.doi.org/10.1109/MC.2009.263⟩.

LLERENA, N. E. M. *Ensembles na classificação relacional*. Dissertação (Mestrado) — Universidade de São Paulo, São Carlos, BR, 2011.

MANZATO, M. G. gSVD++: supporting implicit feedback on recommender systems with metadata awareness. In: *Proceedings of the 28th Annual ACM Symposium on Applied Computing*. New York, NY, USA: ACM, 2013. (SAC '13), p. 908–913. ISBN 978-1-4503-1656-9. Disponível em: ⟨http://doi.acm.org/10.1145/2480486.2480536⟩.

MANZATO, M. G.; DOMINGUES, M. A.; REZENDE, S. O. Optimizing personalized ranking in recommender systems with metadata awareness. In: *Proceedings of the 2014 IEEE/WIC/ACM International Conference on Web Intelligence*. [S.l.: s.n.], 2014.

MASTHOFF, J. Group recommender systems: Combining individual models. In: *Recommender Systems Handbook*. [S.l.]: Springer, 2011. p. 677–702.

MCFEE, B. et al. The million song dataset challenge. *Proceedings of the 21st international conference companion on World Wide Web - WWW '12 Companion*, ACM Press, New York, New York, USA, p. 909, 2012. Disponível em: ⟨http://dl.acm.org/citation.cfm?doid=2187980.2188222⟩.

MITCHELL, T. M. *Machine Learning*. 1. ed. New York, NY, USA: McGraw-Hill, Inc., 1997. ISBN 0070428077, 9780070428072.

MORAIS, S. Sistemas de recomendação em rapid miner: um caso de estudo. 2012.

MUSTO, C. Enhanced vector space models for content-based recommender systems. In: ACM. *Proceedings of the fourth ACM conference on Recommender systems*. [S.l.], 2010. p. 361–364.

NEWCOMBE, J. *Intelligent Radio: An Evolutionary Approach to General Coverage Radio Receiver Control*. Dissertação (Mestrado) — DeMontfort University, UK, 2013.

ORTEGA, F. et al. Improving collaborative filtering-based recommender systems results using pareto dominance. *Inf. Sci.*, Elsevier Science Inc., New York, NY, USA, v. 239, p. 50–61, ago. 2013. ISSN 0020-0255. Disponível em: ⟨http://dx.doi.org/10.1016/j.ins.2013.03.011⟩.

OTT, A. *The 24-Hour Customer: New Rules for Winning in a Time-Starved, Always-Connected Economy*. HarperCollins, 2010. ISBN 9780062002792. Disponível em: ⟨http://bks6.books.google.com.vn/books?id=O1YvdYoVe4EC⟩.

PIOTTE, M.; CHABBERT, M. The pragmatic theory solution to the netflix grand prize. *Netflix prize documentation*, 2009.

RENDLE, S. Factorization machines. In: IEEE. *Data Mining (ICDM), 2010 IEEE 10th International Conference on*. [S.l.], 2010. p. 995–1000.

RENDLE, S. et al. BPR: Bayesian personalized ranking from implicit feedback. In: *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. Arlington, Virginia, United States: AUAI Press, 2009. (UAI '09), p. 452–461. ISBN 978-0-9749039-5-8. Disponível em: ⟨http://dl.acm.org/citation.cfm?id=1795114.1795167⟩.

RESNICK, P.; VARIAN, H. R. Recommender systems. *Communications of the ACM*, ACM, v. 40, n. 3, p. 56–58, 1997.

RICCI, F.; ROKACH, L.; SHAPIRA, B. Introduction to recommender systems handbook. In: *Recommender Systems Handbook*. [S.l.]: Springer, 2011. p. 1–35.

SAID, A. *Evaluating the Accuracy and Utility of Recommender Systems.* Tese (Doutorado) — Technischen Universitat Berlin, 2013.

SEAGATE. *Belajar Sistem Perekomendasi - Corat-coret di Halaman Web.* 2015. Disponível em: ⟨http://blog.seagatesoft.com/2013/07/14/belajar-sistem-perekomendasi/⟩.

SENOT, C. et al. Analysis of strategies for building group profiles. In: *User Modeling, Adaptation, and Personalization.* Springer, 2010, (Lecture Notes in Computer Science, v. 6075). p. 40–51. ISBN 978-3-642-13469-2. Disponível em: ⟨http://dx.doi.org/10.1007/978-3-642-13470-8_6⟩.

SHARDANAND, U.; MAES, P. Social information filtering: Algorithms for automating &ldquo;word of mouth&rdquo;. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.* New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1995. (CHI '95), p. 210–217. ISBN 0-201-84705-1. Disponível em: ⟨http://dx.doi.org/10.1145/223904.223931⟩.

SIERSDORFER, S.; SIZOV, S. Social recommender systems for web 2.0 folksonomies. In: ACM. *Proceedings of the 20th ACM conference on Hypertext and hypermedia.* [S.l.], 2009. p. 261–270.

STECK, H. Item popularity and recommendation accuracy. In: *Proceedings of the Fifth ACM Conference on Recommender Systems.* New York, NY, USA: ACM, 2011. (RecSys '11), p. 125–132. ISBN 978-1-4503-0683-6. Disponível em: ⟨http://doi.acm.org/10.1145/2043932.2043957⟩.

TERVEEN, L.; MCDONALD, D. W. Social matching: A framework and research agenda. *ACM Trans. Comput.-Hum. Interact.*, ACM, New York, NY, USA, v. 12, n. 3, p. 401–434, set. 2005. ISSN 1073-0516. Disponível em: ⟨http://doi.acm.org/10.1145/1096737.1096740⟩.

TÖSCHER, A.; JAHRER, M.; BELL, R. M. The bigchaos solution to the netflix grand prize. *Netflix prize documentation*, 2009.

TSO-SUTTER, K. H.; MARINHO, L. B.; SCHMIDT-THIEME, L. Tag-aware recommender systems by fusion of collaborative filtering algorithms. In: ACM. *Proceedings of the 2008 ACM symposium on Applied computing.* [S.l.], 2008. p. 1995–1999.

VOORHEES, E. M.; HARMAN, D. K. *TREC: Experiment and Evaluation in Information Retrieval (Digital Libraries and Electronic Publishing).* [S.l.]: The MIT Press, 2005. ISBN 0262220733.

WOLPERT, D. H. Stacked generalization. *Neural Networks*, v. 5, p. 241–259, 1992.

YANG, B. et al. Online video recommendation based on multimodal fusion and relevance feedback. In: *Proceedings of the 6th ACM International Conference on Image and Video Retrieval.* New York, NY, USA: ACM, 2007. (CIVR '07), p. 73–80.