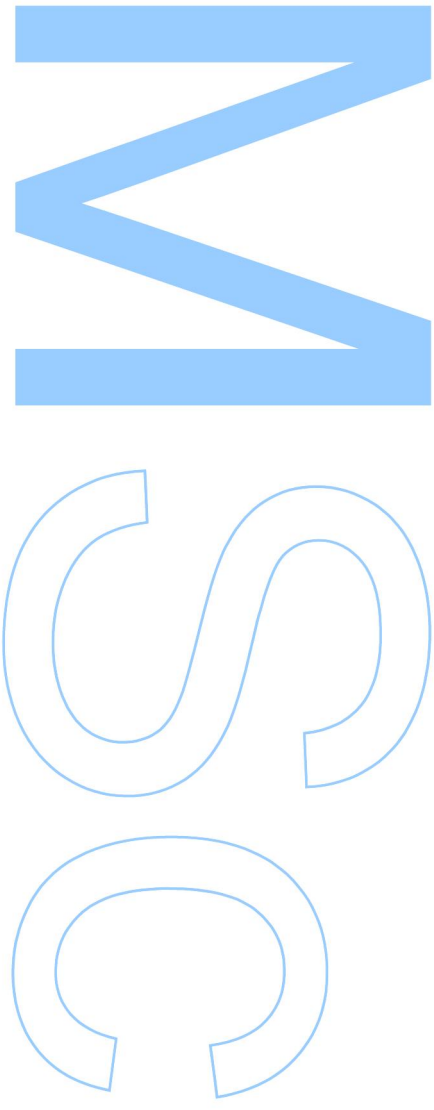
Hierarchical Recommender Systems

Bruna Raquel Ribeiro Madeira
Mestrado em Ciência de Dados
Departamento de Ciência de Computadores
2020

Orientador
João Nuno Vinagre Marques da Silva
Professor Auxiliar Convidado
Faculdade de Ciências da Universidade do Porto

Orientador Externo
Kelwin Correia Fernandes



Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, _____/_____/_____

Acknowledgements

First and foremost, I would like to express my gratitude to my supervisor, Prof. João da Silva, for all the support and guidance throughout this process. I also want to thank my co-supervisor, Kelwin Fernandes, for all knowledge transmitted.

Also, I would like to thank my family. My parents for supporting me unconditionally and for always believing in me. To my sister Filipa, who also taught me not to give up and for making me laugh even through tough times.

Finally, to Rui for giving meaning to everything I do and for believing in something greater than myself. This work would not be possible without them. My biggest thank you!

Abstract

Recommender systems have become a widespread application of machine learning technologies in an extensive spectrum of daily requests and a well-known engine to the general public. Considering information overload situations, such as social networks, e-commerce applications, or streaming platforms, offering personalized recommendations has proven to be a significant source of improved functionality, customer satisfaction, and revenue enhancements. In particular, recommender systems are increasingly being explored by companies in the telecommunications and media industry to improve user satisfaction with their services, namely in the television department.

Individuals spend a significant part of their days consuming television content. The television broadcast services present users with an overwhelming scope of possibilities from which they can choose. Concretely, thousands of programs are available to watch, including live-tv and modern technologies, such as time-shifting and video-on-demand (VODs), representing a massive information overflow problem. Several studies have also shown that people get quickly frustrated when searching for something to watch due to the vast number of possible choices, which decreases the customer's satisfaction relative to television services. Therefore, recommender systems arise as an indispensable mechanism to overcome the difficulties mentioned above by assisting clients in their decisions, increasing user engagement, and content consumption.

Although researchers have proposed numerous recommender systems for the television domain, most recommendation approaches do not consider the intrinsic hierarchical structure of audiovisual content. However, some studies have recently revealed that including the hierarchy of items or user preferences provides valuable information for building personalized recommendations, improving recommender systems' predictive performance.

In this dissertation, we investigate the problem of recommending content at multiple granularity levels using item hierarchies, aiming to provide relevant recommendations and maximize recommendation systems' applicability in several use-cases. We propose the development of a hierarchical recommendation engine for television providers through matrix factorization techniques from collaborative filtering (CF). This system manages the information available in these environments to predict the users' engagement with TV content through implicit feedback.

For the development of this work, particular emphasis is given in the study of the hierarchies within recommendations lists, for which we present two different contributions. The first one

regards the inclusion of techniques using hierarchical information to cope with the ambiguity of user interests and tastes. On the other hand, we propose a solution to fuse novelty in the recommendations to address users' need for unique recommendations. Generally, the main concern for developing recommender systems is to improve the predictive accuracy of the users' preferences. However, there is an increasing awareness that additional metrics are essential to user satisfaction and business performance. For this reason, we address novelty as an overall quality from the system point of view, and we propose a solution for the problem of recommending new items by using higher levels of the item hierarchies.

Our proposal is evaluated on a standard experimental design that considers a dataset of live television content and a well-known baseline recommendation algorithm. The results of our experiments verify the efficacy of the contributions and allow further insights when applied to different environments. We demonstrate that this item hierarchy can be useful in making the recommendations more explainable and increasing the recommendations' novelty without compromising much on the accuracy.

Keywords: Recommender Systems, Hierarchical Recommendation, Television Show Recommendation, Collaborative Filtering, Matrix Factorization

Resumo

Os sistemas de recomendação tornaram-se numa aplicação extensiva das tecnologias de Machine Learning e um mecanismo comum ao público em geral, considerando um amplo espectro nas situações do dia a dia. Tendo em conta situações nas quais existem excesso de informação, nomeadamente redes sociais, aplicações de e-commerce ou plataformas de *streaming*, a produção de recomendações relevantes denota um papel importante para melhorar a funcionalidade e aumentar tanto a receita, como a satisfação do cliente. Em particular, as empresas de telecomunicação e media têm vindo a apostar mais na exploração destes sistemas de forma a melhorar a experiência do consumidor em relação aos seus serviços, nomeadamente no sector televisivo. Atualmente, os indivíduos passam uma parte significativa dos seus dias a ver televisão. Os serviços televisivos apresentam uma panóplia de opções nas suas grelhas de programação, através das quais as pessoas podem escolher o que assistir. De facto, existem milhares de programas para ver, incluindo programas transmitidos em direto e novas tecnologias, tais como, *Catch-up TV* e *Video-on-demand* (VODs), criando um problema de excesso de informação. Por esta razão, os sistemas de recomendação surgem como um mecanismo indispensável para superar todas estas dificuldades, auxiliando os consumidores nas suas decisões com vista a aumentar o envolvimento e o consumo por parte dos clientes. Embora diversas pesquisas tenham proposto vários sistemas de recomendação para o domínio televisivo, a maioria das abordagens não considera a estrutura hierárquica intrínseca do conteúdo audiovisual. No entanto, alguns estudos recentes revelaram que incluir a hierarquia de itens ou preferências do utilizador fornece informações valiosas para a construção de recomendações personalizadas, melhorando o desempenho preditivo dos sistemas de recomendação. No decorrer desta dissertação, investigamos o problema de recomendar conteúdo em vários níveis de granularidade usando hierarquias de itens, com o objetivo de fornecer recomendações relevantes e maximizar a aplicabilidade dos sistemas de recomendação em diversos casos. Assim, propomos o desenvolvimento de um mecanismo de recomendação hierárquica para os serviços televisivos, com recurso a técnicas de factorização de matrizes através de filtragem colaborativa (FC). Este sistema gerencia as informações disponíveis nesses ambientes para prever o envolvimento dos utilizadores com o conteúdo televisivo a partir de feedback implícito. No desenvolvimento deste projeto, demos particular destaque ao estudo das hierarquias em listas de recomendação, para as quais apresentamos duas contribuições distintas. A primeira diz respeito à inclusão de técnicas de informação hierárquica para lidar com a ambiguidade de interesses e gostos do utilizador. Por outro lado, propomos uma solução para fundir a novidade nas recomendações de forma a ir ao encontro das necessidades dos utilizadores através de recomendações exclusivas. Geralmente, a maior dificuldade no desenvolvimento de sistemas de recomendação é melhorar a precisão preditiva das preferências dos utilizadores. No entanto, há uma consciência crescente de que existem métricas adicionais que são essenciais para a satisfação do utilizador e prestação da empresa. Por esta razão, consideramos a novidade nas recomendações como uma qualidade geral do ponto de vista do sistema e propomos uma solução para o problema de recomendação de novos itens através de níveis mais elevados na hierarquia. A nossa proposta está assente num projeto experimental padrão que considera um conjunto de dados televisivos de uma operadora de

telecomunicações e um algoritmo de recomendação bem conhecido. Os resultados das várias experiências que realizamos verificam a eficácia das contribuições e permitem extrair conhecimentos para aplicar em diferentes ambientes. Deste modo, demonstramos que estruturas hierárquicas de itens podem ser úteis no âmbito de tornar as recomendações mais explicáveis e aumentar a relevância das mesmas sem comprometer a precisão.

Palavras-chave: Sistemas de Recomendação, Recomendação Hierárquica, Recomendação de programas televisivos, Filtragem Colaborativa, Fatorização de Matrizes

Contents

List of Tables

List of Figures

Acronyms

CF	Collaborative Filtering
EPG	Electronic Programming Guide
HMF	Hierarchical Matrix Factorization
IPTV	IP Television
MAE	Mean Absolute Error
MF	Matrix Factorization
RMSE	Root Mean Squared Error
STB	Set-top-box
VOD	Video-on-Demand

Chapter 1

Introduction

Online communities have become an essential part of the business world and are continuously growing. Nowadays, people benefit from easily accessible data, which contributes to the dissemination of information and knowledge. The awareness of how data flows and is scattered, and the way we handle information, are increasingly becoming fundamental insights in management for decision making, performing and controlling activities and, consequently, fundamental to the success of companies.

At the same time, the amount of data and the wide variety of content is overwhelming, which makes it harder for users to filter information. In most online systems, it is not feasible to browse through all the available data without automated filtering. This problem and its importance have motivated the development of information filtering algorithms, which has served as a driving force for the expansion of recommender systems technology.

Individuals are facing an increasing number of choices in every aspect of their lives, such as TV programs, music, and books, other taste-based questions such as tourist destinations, restaurants, and specific areas like health insurance plans, job searches, education and learning, and many different domains in which choice matters significantly. The field of recommender systems has a pivotal role in using the wealth of data now available to make these choices manageable, effectively leading people to the best options, resulting in more favorable decisions.

1.1 Motivation

One of the most representative examples of the expanding amount of data is multimedia content, especially in the field of television services. On a typical cable TV service, customers not only have access to live programs but also video-on-demand (VoD) services. In the case of IP Television (IPTV), which broadcasts multimedia content in digital format through broadband Internet networks, it also includes scheduled television programs.

The expansion of digital television and the convergence between conventional broadcasting and IPTV contributed to the gradual increase in the number of available channels and on-demand video content. Although this scenario allows for an improvement in television experience, it also carries new challenges.

In this sense, the rising volume of television content makes it harder for viewers to select relevant items [Hsu et al., 2007]. Overloaded by the several alternatives, many viewers tend to zap systematically between

channels and most of the time, choose the same show repeatedly, or in the worst scenario, give up watching [Gomez-Uribe and Hunt, 2016]. For the television industry, this leads to poor user experiences, which represents a dominant cause in dissatisfaction and, ultimately, users may end up abandoning subscriptions (customer churn). Over time, this problem is getting stronger due to the amplifying number of content brought on by the generalization of digital television and streaming services. Additionally, conventional television content search tools are extensive and do not efficiently meet the viewers' current information needs.

In this perspective, recommendation systems are among the solutions to this evolving world [Adomavicius and Tuzhilin, 2005]. These systems aim to assist users in the search for relevant TV content according to their interests, through explicit user feedback or the monitoring of past user behavior [Bambini et al., 2011].

Despite the growing notoriety of such systems, their actual application in the context of recommending television programs still has open problems in the related literature. For instance, current solutions disregard the unique properties of hierarchical information, which have been proven relevant in different domains. Indeed, user preferences and product catalogs typically follow a hierarchical nature, and this knowledge has been demonstrating its value in the quality of recommendations.

In this context, television shows are a representative example of the inherent nature of hierarchies, in which it is possible to partition programs into categories such as channel, genre, or subgenre. Since users may have preferences regarding any level of the hierarchy, recommender systems for television content should be able to make recommendations at all hierarchical levels, despite most recommendation algorithms typically recommend only at a pre-defined level.

1.2 Problem Definition and Objectives

Given the hierarchical structure of audiovisual content, it becomes necessary to recommend content at multiple granularity levels, aiming to maximize the applicability of recommendation systems in several use cases. For instance, series, seasons and episodes, or even channels, programs category and subcategory, are representative examples of the many use cases that follow the television content structure.

Traditional television holds a specific program grid, so recommendation systems need to adapt to this reality. With the development of a hierarchical recommender system, it is possible to have a model that is applicable in all these cases.

In this dissertation, we propose a recommendation approach that supports the natural hierarchies for television content, to ensure various forms of interaction and improve the recommendations. The main focus is the development of a recommendation method able to predict the best television content offer to individual customers at any hierarchical level. In this project, we deal exclusively with positive-only data, which regards of user-item interactions indicating positive preferences. This type of feedback contains only positive interactions between users and items and does not provide information covering non-positive, neutral and negative, opinions.

The main reason to center our research on positive-only data is availability, considering that almost any system that involves interactions between users and items has a source of positive feedback. In real-world online systems, representative examples for positive-only data are endless, such as for music streaming records, social network patterns, shopping habits, news reading, and movies or series visualization history.

However, unlike recommender systems that deal with numeric ratings – e.g. 1 to 5 stars –, systems that learn from positive-only data face the problem of not having information about negative preferences, the not desirable items. This circumstance adds additional complexity to the task of identifying suitable items to recommend since it is not clear to distinguish between items users do not like and items that users do not know. Thus, there is a fine line among elements that users do not want and items that users are unaware of, and being able to establish this segregation can make the difference between an inadequate recommendation or an ideal one.

Given the general problem setting, the goal is to learn recommendation models from positive-only feedback data, which, in this context, arises from the history of the television set-top box (STB) usage.[1]

[1]This work was developed within the scope of an internship in a Portuguese telecommunications company

1.3 Contributions

The accomplishment of this dissertation lead to the following main contributions:

- A survey of the state-of-the-art on recommendation systems, with particular focus on the ones that deal with hierarchical models. Based on this research, we observe that exploiting hierarchical structures are frequently advantageous in recommender systems and identify the most significant related challenges.
- A complete characterization of the Cable TV data and the development of a reliable television dataset, fundamental for relevant and personalized recommendations.
- A study on the application of hierarchical models for audiovisual content to maximize and optimize the applicability of recommendation systems in various use cases, and its comparison to standard pre-defined level solutions.
- Development of multiple tests specific to the domain of recommendation systems for television content.
- Proposal and evaluation of hierarchical strategies in the context of the recommendation of television programs that combine different types of procedures, including machine learning algorithms that aim to improve the accuracy of results.

1.4 Dissertation Structure

After this initial chapter, this dissertation is organized as follows:

- **Chapter 2** introduces the fundamental concepts that constitute the foundation for our research on recommendation systems. We identify the tasks of recommender systems and provide a general view on the algorithms that can be used to accomplish those tasks, with a particular focus on the ones directly useful for the core problem of the dissertation. We also analyze the main related works to provide an overview of algorithms that exploit hierarchical information to improve recommendations.
- **Chapter 3** begins by detailing the specific properties of Cable TV content and provides a complete characterization for this context. After, we define the dataset applied for this project and present an in-depth analysis of the data.
- **Chapter 4** converges specifically on recommendation algorithms that are capable of learning from television box usage data. Then, describes the proposed methodologies that are suitable for recommendation with television content data. Finally, this chapter presents the developed system pipeline and implementation.
- In **Chapter 5** the evaluation methodology is presented, as well as the results obtained and respective discussion.
- Finally, we draw conclusions in **Chapter 6**, by presenting the core of our findings, the limitations of our proposals, and possible future paths of work that can complete or complement this dissertation.

Chapter 2

Background and Related Work

Throughout this chapter, we present a review of the related work in the topics of interest and a broader perspective of the latest advances in the field, especially in content-based filtering, collaborative filtering, hybrid approaches, and evaluation methodologies. Furthermore, as part of our vision of the recommendation problem as a hierarchical approach, we convey this study as a potential source of new perspectives and techniques to improve recommendations using hierarchical models. This chapter is organized as follows. First, Section 2.1 introduces a brief overview of the Recommender Systems field, including essential concepts and objectives. Then, Section 2.2 describes some introductory background and context. Concerning section 2.3, we exhibit the possible tasks of recommender systems and also scenarios in which they may occur. In Section 2.4, we delve into the study of the related work on Recommender Systems, starting with the analysis of extended perspectives in different use cases and, later, focusing on recommendation models for television content. Conclusively, Section 2.5 covers the literature in hierarchical techniques for Recommender systems, but also different Machine Learning problems relevant to our research.

2.1 Recommender Systems: An overview

Recommender Systems are software tools and techniques that provide suggestions for users regarding a catalog of items - music, books, movies, touristic destinations, or any other content of interest - in a personalized manner [Adomavicius and Tuzhilin, 2005; Resnick and Varian, 1997; Ricci et al., 2011]. Recommender systems are mostly designed for online communities that include a substantial number of users that browse through a large number of items.

In scenarios of information overload [Toffler, 1970], facing an overwhelming array of choices that makes the exploration and selection of relevant content a difficult task for the user, these systems become especially useful. The personalized assistance offered by recommender systems has proven to be an effective method to improve user satisfaction and increase the revenue of many e-commerce and media streaming platforms such as Amazon, Netflix, Spotify or YouTube, and social networks such as Facebook or Twitter.

Over the last decade, Recommender Systems prevail as an active field of research and continues to attract an increasing level of attention, since it builds on the set of multidisciplinary technologies and algorithms from various fields, including information systems, Information Retrieval, Machine Learning, Human-Computer Interaction, and even more distant areas, such as Marketing or Economics.

For the remaining content of this section, we present the revision of the state-of-the-art in Recommender Systems. Firstly, focus on the particularities of different domains in which recommendations can be offered. Then, provide a classification of the recommendation algorithms observed in the literature. The evaluation of Recommender Systems, a central topic of this dissertation, is also introduced here. Afterward, we discuss several limitations, issues, and challenges that current research in the area faces. Finally, we present some of the arising techniques and services available for the implementation and deployment of hierarchical information on recommender systems and other applications, with particular emphasis on the context of interactive television.

2.2 Definitions and Notation

In this dissertation, we focus on hierarchical recommender systems for television content. However, to describe these technologies, some introductory background and context are required. This section provides a general introduction to recommender systems, focusing on the basic concepts and main definitions that are essential for the course of this dissertation and also to the reader's support. In general, the recommendation problem can be expressed as the suggestion of items from a catalog I in a particular recommendation situation, such as music, movies, television programs, or other products, to a community of users U - typically human beings. Additionally, to make personalized recommendations, some prior knowledge about the users is required, that can occur in the form of the items they have rated, bought, or even consumed. The interaction data is represented in the form of a matrix R, $|U| \times |I|$, whose elements R_{ui} represent the interaction between users and items. Figure 2.1 is an illustration of this user-item matrix. The most simple form refers to binary interactions between users and items, and the interaction matrix contains only the values 0 and 1, i. e., $R \in 0, 1^{|U| \times |I|}$. This type of representation indicates if the user bought, consumed, watched, or listened to a particular item, etc. On the other hand, there is the case when users provide numerical ratings, for example, 1 to 5 stars. In this situation, the matrix R assumes values in the range $\{0, 1, 2, 3, 4, 5\}$, whereby 0 denotes the absence of a rating.

	I_1	I_2	I_3	I_4	I_5	$\cdots$	I_n
U_1		2	5				2
U_2				5			
U_3		4		2	3		
$\vdots$							
U_n		4					3

Figure 2.1: An example user-item rating matrix.

Under circumstances where it is possible to quantify the interaction between users and items, such as the amount of time listening to some music or watching a particular television program, the interaction matrix can assume values included in the set of natural numbers $\mathbb{N}$. In the case of more complex cases, so as time data for the interactions, it is straightforward adjusting to similar formulations. For all former cases, the interaction matrix R can be interpreted as the pairs of users and items $R \subset U \times I$ with any interaction. Following this representation, $I_u = \{i \in I : (u, i) \in R\}$ expresses the user profile or subset of items that

user u interacted with, and $U_i = \{u \in U : (u, i) \in R\}$ denotes the item profile or subset of users that had any interaction with the item i. Applying the interaction data, provided by matrix R, a recommender system S aims to produce recommendations R_u^S for every user. Consequently, a recommendation is denoted as a set of items $R_u^S \subset I$ presented to the user.

2.3 User tasks for Recommender Systems

As declared in the preceding section, users and items are the two central entities in recommendation problems. Recommender systems are typically designed for online communities encompassing a large number of users that browse through a large number of products in the system. The scope behind all recommender systems is essentially the same, i.e., provide items recommendations to users. However, to properly evaluate a recommender system, it is necessary to understand the end-user tasks and purposes for which it is being employed. Herlocker et al. [2004] identified a set of recommender tasks that characterize the user goals for a recommender system:

Annotation in Context This user task involves the annotation of items with a value that shows the item's relevance/usefulness to the user. The recommender system supports learners to achieve this specific learning goal, which means to predict how much the product is relevant to the user. These recommendations occur while the user carries out other tasks, e.g., when predicting how relevant the links are within a page.

Find Good Items The principal motivation for users to use recommendation systems is to find the set of items they are most likely to prefer, e.g., receive a list of web pages to visit. This type of system suggests particular items to users, providing users with a ranked list of the recommended items, along with predictions for how much the users would like them [Hill et al., 1995; Shardanand and Maes, 1995]. Often referred to as the top-N recommendation, this core recommendation task is present in numerous areas of research and other commercial systems.

Find All Good Items Recommendation of all relevant elements, for instance, receive a complete list of references on a topic. This user-end task is a variation of the *Find good items*, with the condition of not disregard any significant recommendation.

Recommend Sequence Applied to recommendations of a sequence of items. In this case, a possible illustration could be to propose a sequence of songs in a music streaming system. The purpose of these models is to obtain a good sequence of items that the user can access arbitrarily.

Just Browsing Recommendations out of the box while the user is browsing. Some users find it amusing to browse through a product catalog, even without any imminent interest in its consumption. For these cases, users are more interested in the navigation functionality and interface provided by recommender systems.

Find Credible Recommender Recommendations during the initial exploration/testing phase of a system, e.g., suggest movies that the user will definitely like. This is a common task of new users to detect if the recommender is capable of accurately match their preferences. In these earlier testing stages, users can only evaluate recommendations of items that they already know of, which naturally causes bias towards inadequate recommendations.

The user tasks mentioned above convey different implications in the formulation of the recommendation problem and the respective evaluation methodology and metrics.

Accordingly, it becomes imperative to identify the task description for which the recommender is designed to help distinguish among different evaluation measures. In the recommendation domain, the vast majority of research converges on either *Annotation in Context* or *Find Good Items* [Herlocker et al., 2004].

Considering the purpose of this dissertation, the user task to be covered is *Find good items*, also known as *top-N recommendation*.

2.4 Types of Recommender Systems

The problem of recommending items has been studied extensively, and several approaches have emerged to the development of recommendation systems, which can be implemented in different domains. Therefore, recommendation algorithms can be classified according to multiple criteria, such as the type of feedback, the recommendation task to which it returns, and many other examples. However, the most common division observed in the literature focuses on the distinction of how it is used the information from user preferences toward items. Thus, is usual to classify these systems into three broad groups, and the following architectures are established:

- **Content-based Filtering:** examines the properties of items and recommendations are based on the content or features of the items in the recommendation domain.
- **Collaborative Filtering:** focus on the relationship between users and items, and recommendations are based on the consumption patterns of the users.
- **Hybrid recommenders:** a combination of the previous methods.

Some authors also refer to additional types of recommendation systems with less research activity, such as demographic, utility-based, and knowledge-based recommendations [Burke, 2002].

2.4.1 Content-based filtering

Content-based filtering analyzes the user profiles, or history, and combines the obtained information with the content of the items for issuing recommendations [Pazzani and Billsus, 2007]. In content-based methods, the user profiles are built based on features and descriptive attributes of items rated by the user [Lops et al., 2011]. The recommendation process consists of matching up the properties of a user profile in which preferences and interests are stored, with the attributes of a content object. The result is a significant judgment that represents the user's level of interest in a certain item, used to provide relevant recommendations. For instance, in a recommender system for news articles, if the information about a

user indicates a preference for politics, news with significant content about these fields of interest will be recommended.

Content-based recommendation approaches can be applied to a variety of domains, extending from recommending television programs, web pages, restaurants, news, and items for sale. In all applicable scenarios, content-based filtering methods involve the description of items that can be recommended, the creation of a user profile that describes the type of items the user likes, and a comparison of items to the user profile to determine what to recommend. First, to match and measure the similarity between the profiles, it is assumed that a user's and an item's profile share a common method of representation, for instance, using keywords. Then, the result of the matching process indicates the similarity between the item and the user's profile [Shoval et al., 2008].

2.4.1.1 Item Profiles

In content-based recommender systems, the descriptive attributes of items are used to provide relevant content recommendations. Consequently, it is necessary to create a profile for each item, which is a record or a set of records representing the relevant characteristics of that item. In more manageable situations, the profile includes some properties of the item that are instantly revealed. To illustrate, item profiles concerning the film industry denote a vector of features of a particular movie, namely the group of actors, the corresponding genre, or even the released year. These are all representative characteristics that might be relevant to a recommendation system.

Many other classes of items also allow us to obtain features from available data. Indeed, products usually have descriptions written by the manufacturer, giving features relevant to the class of products, like the screen size or the display resolution for a television. Books have information descriptions analogous to the ones for movies, so it is possible to obtain features like the author, year of publication, or genre. Similarly, the music industry holds available features such as artists, composers, and style.

However, not all classes of items are that simplistic, and there are other situations in which it is not directly clear what the values of features should be. Among other exemplary cases, document collections and images are typically mentioned in the literature [Rajaraman and Ullman, 2011], which will be discussed below.

There are several types of documents for which a recommendation system can be useful. For example, a content-based recommender system can suggest a collection of documents such as web pages or articles addressing topics appealing to the user. The main issue regarding this group of documents is the lack of readily available information giving features. For extracting features from documents, the identification of words that characterize the subject of a document, typically relevant or unusual words, has been a useful practice. Term Frequency Inverse Document Frequency (TF-IDF) is a simple and effective algorithm to determine which words, in a corpus of documents, might be more relevant for characterizing the document [Ramos, 2003]. Words with higher TF-IDF scores indicate a powerful connection with the document they appear in, suggesting that if a specific word appears in a query, the document could be of relevance to the user.

Along with document collections, image databases are also an illustrative example in which the problem of discovering useful features for items subsists. There have been several approaches using content-based tag processing for images, covering the need to obtain information about features of items by requesting users to tag the items with appropriate words or phrases that express the item [Liu et al., 2010].

2.4.1.2 User Profiles

Content-based filtering systems explain the affinity of each user to each piece of content based on the user's historical content consumption. This profile may consist of many different types of information and is created by measuring the frequency with which certain features appear in the items the user likes. In this way, it is possible to estimate the degree to which a user will like an item by the closeness of the item's profile to the user's profile. As stated by Pazzani and Billsus [2007], content-based recommendation systems mostly rely on user profiles, which belong to one of two types:

- **A user preference model**, corresponding to a description of the kind of items that are interesting for the user. There are several representations of this description, but a common one is a function that estimates, for any item, the probability that the user is interested in that item [Pazzani and Billsus, 2007].
- **A history of user activity**, which may include the set of items that a user accessed or interacted with in the past, such as purchased items or explicit ratings given by the user.

As a result, content-based filtering systems require proper techniques for representing the items and designing the user profile, and some essential strategies for comparing the respective user profile with the item representation. According to Lops et al. [2011], content-based systems are structured into three different elements:

Content Analyzer The purpose of this module is to represent the content of items in a suitable way for processing by the following components. This component analyses unstructured information and provides them as structured data, e.g., an unstructured text can be converted to a structured keyword list. Usually, content analyzers use techniques from information retrieval systems [Lops et al., 2011; Manning et al., 2008].

Profile Learner This component exploits data retrieved from the Content Analyzer to collect representative information of the user interests. Then, it tries to generalize this data, aiming to build the user profile, which is a structured representation of user preferences used to recommend relevant items. In the specific case of television, it can be a list of the program categories with its associated weight.

Filtering Component This module compares the user profile with the content of each item, and thus determines the relevance that the item presents to the user. Similarity metrics, such as Jaccard Index or Cosine Distance, can be used to measure the similarity between items and users.

2.4.1.3 Benefits and Limitations

The content-based recommendation paradigm presents some advantages in its application when compared to the collaborative approach. One of the major problems in recommender systems is known as the cold-start problem [Lika et al., 2014], which occurs when the number of initially available ratings is relatively small, so there is not enough information to compute a reliable recommendation model. In this context, content-based filtering is a more robust technique since it can deal with the new item problem, which means it has the capability of recommending new items when user feedback is nonexistent [Melville

et al., 2002]. Moreover, this method does not require data from other users since it solely exploits ratings provided by the active user to build its profile [Lops et al., 2011]. Accordingly, it is suitable for providing item recommendations to users with unique tastes. Another inherent benefit of using content-based recommender systems is the transparency of its recommendations [Lops et al., 2011], considering it can provide explanations of how the recommendation process works by listing the corresponding features or descriptions that led a particular item to be recommended. This explicit information holds a powerful and intuitive indicator to decide whether a recommendation is relevant and trustworthy or should not be considered. From the user's point of view, it also confers a more engaging experience in the sense that it makes the user aware of the item's attributes that might be suitable for his profile.

Nevertheless, despite the previously mentioned advantages, content-based algorithms have numerous drawbacks. Contrarily to collaborative filtering methods, content-based techniques are very dependent on the recommendation domain. In this sense, content-based filtering operates when the user's information is available and when it is possible to obtain information from the item's content. The first obstacle arises from the necessity to collect information about users, as it is not possible to provide reliable recommendations if the users' information is not accessible or even incomplete [Adomavicius and Tuzhilin, 2005]. In the case of a new user, the system will not be able to collect enough ratings to understand the user's preferences, resulting in a lack of accurate recommendations. Since these systems rely on the availability of sufficient and accurate information about the properties of the items, another shortcoming emerges in finding the relevant features, which sometimes are costly to obtain.

Additionally, the limitations of content analysis denote a concrete restraint when applying this recommendation paradigm [Balabanović and Shoham, 1997; Shardanand and Maes, 1995]. Content-based filtering methods have a natural limit on the number and type of features associated with the items to be recommended [Lops et al., 2011]. Considering that content-based approaches rely on keyword similarity to interpret and recognize the content, this shortcoming translates into the inefficiency of capturing complex relationships at a deeper semantic level [Mobasher et al., 2001]. As these representations only capture certain aspects of the content, it leads to a decay in the recommendation performance, since irrelevant items are retrieved, and various items are incorrectly missed [Blair and Maron, 1985]. In pursuance to overcome these fragilities, extensions to the standard content-based method need to be considered, which may include domain knowledge, or even domain-specific ontologies [Shoval et al., 2008]. For the analysis of multimedia data (e.g., video, audio, or graphical images), extracting relevant information can also become an extremely challenging task [Adomavicius and Tuzhilin, 2005]. An alternative to handle this type of content involves adding metadata to items, such as text attributes [Pazzani and Billsus, 2007].

Content-based systems may also suffer from over-specialization [Balabanović and Shoham, 1997]. This drawback concerns the situation when the system can only recommend items that score highly against a user's profile, implying that the user is limited to being recommended items that are similar to those already rated. Under particular circumstances, this can be the desired behavior, but it can also hold several limitations as it never recommends items outside the user's profile. For example, a person with a specific preference for Mexican cuisine would never obtain a recommendation for other types of restaurants, regardless of its quality or notoriety. However, recommending all Mexican restaurants does not seem the ideal scenario to grant relevant suggestions. This shortcoming is also described as the serendipity problem to accentuate the tendency of the content-based approaches in providing recommendations with a limited degree of novelty [Lops et al., 2011].

Conclusively, as the number of products increases, the effectiveness of content-based filtering steadily decreases, considering that the number of items in the same content-based category also expands [Claypool et al., 1999].

2.4.1.4 Proposals for Content-based approaches

As previously stated, content-based filtering systems recommend an item to a user based upon a description of the item and a profile of the user's interests. In other words, suggest items identical to those that the user has shown interest in the past. Research on content-based recommender systems intersects with multiple computer science subjects, mainly Information Retrieval [Baeza-Yates et al., 1999] and Artificial Intelligence. Indeed, the literature for content-based recommendation algorithms presents various proposals that draw algorithms and perspectives from diverse domains, namely Machine Learning, Information Retrieval, and Semantic Web technologies. In the context of Artificial Intelligence, approaches of Machine Learning (ML) technologies exploits various techniques such as Bayesian classifiers, decision trees, clustering, and artificial neural networks (ANNs) for Web recommendation [Pazzani and Billsus, 2007]. Mooney and Roy [2000], describe a content-based book recommender system that implements a machine-learning algorithm for text categorization, using naive Bayes. Relating to the use of Information Retrieval systems, its research derives from the vision that the users' search for recommendations is involved in an information-seeking process, and, accordingly, similarities between items are measured through these techniques [Manning et al., 2008]. On recommender technologies, term-based weighting approaches were applied in early proposals about personalized news recommendation (Lang, 1995), automated Web browsing and recommendations [Balabanović and Shoham, 1997], and, in more recent outlines, to weighted lists of social tagging systems [Cantador et al., 2010]. Semantic Web applications have been proposed for content-based systems, as in the case of movie recommendation, news articles [Cantador et al., 2008], and music recommendations using an experimental linked information system [Ostuni et al., 2013; Passant, 2010]. Relatively to the film industry, the proposed method in Mak et al. [2003] recommends movies by using text categorization techniques to learn from movie synopses. As for the music domain, FOAFing the music [Celma et al., 2005; Golbeck and Rothstein, 2008] can recommend, discover, and explore music content. This system bases on user profiling via Friend of a Friend (FOAF) descriptions and content-based descriptions automatically extracted from the audio itself.

Through the literature review, it is possible to state that content-based recommendation algorithms are mostly used in text-based systems. However, recent advances in multimedia content analysis, particularly in speech recognition and image-based systems, have provided new tools and a broader spectrum of techniques for the research in content-based recommendation.

2.4.2 Collaborative filtering

As previously mentioned, many online communities incorporate a large number of users that browse through items in the system. Items may refer to movies, music streams, touristic destinations, books, or any other category regarding products of interest. Generally, these systems allow users to provide their personal opinion about items, by rating them explicitly, e.g., using a "like" button or a 5-star rating scale, or implicitly, e.g., the number of times a user watches a particular movie genre, or whether if a user has purchased the item or not.

Once again, we are denoting that a system has n users and m items. Through the acquisition of user feedback, it is possible to create a user-item interaction matrix $R_{n \times m}$, containing all ratings given by users to items. One of the most significant problems facing recommender systems is that R is a very sparse matrix, considering that users usually only rate a very small portion of all existent items.

Collaborative Filtering (CF) algorithms attempt to make predictions about individual user preferences

by exploiting the preferences of similar users. The collaborative approach reaches this by learning or training a predictive model that includes the available usage data, the feedback matrix R. Then, the recommendation model can be used to predict the user preferences that are missing in R. For training this model, there are several possible approaches to consider, and the choice relies upon the type of feedback collected.

2.4.2.1 Type of feedback

An important distinction in recommender systems, which determines the choice or development of an algorithm, is related between the two possible types of user feedback:

- **Numeric ratings or explicit feedback:** typically composed of triples in the form (u, i, r), consisting of the rating value r being given to the item i by user u;
- **Positive-only or implicit feedback:** a set of pairs in the form (u, i), that represents a positive interaction between user u and item i.

Given the two possible ways of representing user feedback, the content of the user-item matrix is naturally different for each case. The following figure (2.2) illustrates the user-item matrices for both numeric ratings and positive-only data.

	i_1	i_2	i_3	i_4	...	i_n
u_1	1		5			
u_2			4			2
u_3				3		
u_4		1		5		
...						
u_m			2			

	i_1	i_2	i_3	i_4	...	i_n
u_1	√		√			
u_2			√			√
u_3				√		
u_4		√		√		
...						
u_m			√			

Figure 2.2: Example of feedback matrices. On the left, a numerical ratings matrix. On the right, a positive-only feedback matrix, retrieved from Vinagre et al. [2015].

Considering the first type of feedback, when numeric ratings are available, the main task involves predicting missing values in the interaction matrix, and the problem is deemed as a simple rating prediction task. On the other hand, the formulation differs in systems that employ positive-only ratings. These types of systems are usually present in many acts of daily life, and typical examples can be music streaming, shopping carts, news reading, like buttons, among many others.

In such cases, the interaction matrix R corresponds to a Boolean matrix where true values designate a positive user preference and false ones, often the vast majority, indicating that either the user dislikes the item or is unaware of it. In systems with positive-only user feedback, the task is to predict true values in

R, which is more closely associated with classification problems. In the literature, positive-only feedback is also known as implicit feedback [Hu et al., 2008], or even binary ratings [Volkovs and Yu, 2015].

In this dissertation, we decided to adopt the term positive-only feedback, since the other designations may lead to erroneous interpretations. The term binary may insinuate the existence of both positive and negative feedback, and the term implicit, on the other hand, may not be a precise denomination. When a user clicks in the like button, for instance, it is hard to perceive it as an implicit preference statement. Recommendation using positive-only feedback is also identified in the literature as one-class collaborative filtering [Pan et al., 2008].

Later, a forceful concept was also introduced to improve the search in multimedia databases entitled relevance feedback [Rui et al., 1998; Wu et al., 2000].

The term relevance feedback has been used to define the process of collecting information from users about the relevance of features in a given system. The main idea of this technique is to exploit the content of features to set a weight measure, corresponding to a quantitative level of user preferences [Wu et al., 2000]. Subsequently, the goal is to determine the appropriate weights to model the users' requirements. This information can be provided through a variety of methods, and each system can use it in a specific way to enhance its performance.

Considering the core of this dissertation, this type of feedback expresses a significant concept since the box usage record can gather more than solely positive-only data. In television recommendation systems, it is possible to obtain the number of times the user has watched a particular program, but also the amount of time they spent watching it. In addition to a binary matrix that identifies whether a user has seen a specific television show or not, relevance feedback also retrieves a weighted matrix, representing the user's visualization time of the program.

As outlined before, the focus of this research covers a *top-N recommendation* task. In recommender systems, it is important to highlight that the type of feedback holds significant implications. For instance, operating with numeric ratings, the list of recommendations is created by simply sorting items in descending order of predicted rating. On the other hand, in terms of positive-only data, the same procedure becomes less trivial.

In general, there are two distinct courses to choose, either predict a preference level for items, and then sort them for each user, or address the task using learning to rank methods, aiming to provide a personalized list of items ranked according to the user' preferences [Liu et al., 2009].

In the following section, we review the most practiced algorithms in collaborative filtering approaches. Fundamentally, the methods observed in the literature differ on the strategy used to process data in the matrix of ratings and can be covering neighborhood approaches, or latent factor techniques [Koren et al., 2009].

Neighborhood methods focus on measuring the relationships between items or users. The item-based strategy evaluates a user's preference for an item based on ratings of "neighboring" items by the same user. A product's neighbors are other products that get similar ratings when rated by the same user [Li et al., 2017]. On the opposite side, Latent factor models denote an alternative approach that seeks to explain the ratings by defining both items and users on factors inferred from the ratings' patterns. The following subsections proceed into more widespread information on these two possible approaches.

2.4.2.2 Neighborhood-based algorithms

The fundamental idea behind Neighborhood-based algorithms is to calculate user or item neighborhoods using similarity measures, in particular, the cosine distance or Pearson correlation coefficient [Sarwar et al., 2001]. Similarity functions are applied to calculate how similar two items or two users are, and a higher value means a more powerful relation between them. For example, the similarity between two users should be higher if both users rated the same items equally, indicating they have identical tastes.

Considering the interaction matrix R in which rows correspond to users and columns refer to items, the similarity between two users, u and v, is acquired through the corresponding rows to those users, R_u and R_v. Analogously, the similarity between two items, i and j, can be obtained using the columns corresponding to those items, R_i and R_j. In applying this method, the recommendations are measured through search and aggregations of user or item neighborhoods.

Like all algorithms, neighborhood methods own advantages and disadvantages in its application. The main benefits are related to the ease and simplicity of implementation, and the trivial explainability of the recommendations, since similarities of user and item are intuitive concepts. Nonetheless, neighborhood-based algorithms suffer from a lack of scalability, as the complexity of the system increases as the number of users and items rises as well. Based on the type of collaborative objects, these methods are divided into two categories, user-based CF and item-based CF. The following subsections describe the conception of these groups and the meaningful differences between them.

User-based CF

As the name indicates, user-based CF exploits similarities between users to create user neighborhoods. The likeness between two users, u and v, is computed using similarity measures, usually based on the Cosine distance or, most commonly, the Pearson correlation coefficient [Cremonesi et al., 2010].

- **Cosine Similarity**

  A similarity measure that transforms each user ratings in a vector and calculates the cosine between both vectors. Given two users u and v, the cosine distance considers the rows of the matrix of ratings, R_u and R_v, as vectors in a space with dimension equal to the number of items rated by u and v:

$$sim(u,v) = cos(R_u, R_v) = \frac{R_u \cdot R_v}{\|R_u\| \times \|R_v\|} = \frac{\sum_{i \in I_{uv}} R_{ui} R_{vi}}{\sqrt{\sum_{i \in I_u} R_{ui}^2} \sqrt{\sum_{i \in I_v} R_{vi}^2}} \tag{2.1}$$

  In equation 2.1, $R_u \cdot R_v$ describes the scalar product between R_u and R_v, I_u and I_v are the collections of items rated by u and v, respectively. $I_{uv} = I_u \cap I_v$ is the group of items rated by both users u and v. Each vector contains the ratings the user gave to the items and zero for unrated items. This method does not contemplate the effects of mean and variance. Cosine-based similarity holds a natural drawback, considering that distinct users may have different conceptions of the rating scale. For example, in a recommender system using a rating scale from 1 to 5, a particular user can interpret the value 3 as a positive rating, while another can perceive it as negative feedback. As a result, different preference levels can be represented using the same rating value, and equal likeness levels may output in different ratings. Alternatively, it is possible to use the Pearson Correlation coefficient to overcome this problem.

- **Pearson Correlation Coefficient**

  For users u and v, the Pearson Correlation is measured by:

$$sim(u,v) = \frac{\sum_{i \in I_{uv}} (R_{ui} - \overline{R}_u)(R_{vi} - \overline{R}_v)}{\sqrt{\sum_{i \in I_{uv}} (R_{ui} - \overline{R}_u)^2} \sqrt{\sum_{i \in I_{uv}} (R_{vi} - \overline{R}_v)^2}} \tag{2.2}$$

  where $\overline{R}_u$ and $\overline{R}_v$ are the average ratings provided by users u and v, respectively. When using the Pearson correlation measure, the scale interpretation problem lessens, since the averages bear a normalizing effect on the ratings given by different users. This correlation-based measure, adopted by Resnick et al. [1994], removes the effects of mean and variance, contrarily to cosine similarity.

Alternatively, it is possible to adopt different similarity coefficients such as the Overlap coefficient, Jaccard Index, or Dice similarity [Kaminskas and Ricci, 2011], taking advantage of the specific form of the interactions between users and items, as rating data, play counts, among others.

Rating prediction Essentially, neighborhood-based algorithms are heuristics that perform rating predictions based on the set of previously rated items by the users [Adomavicius and Tuzhilin, 2005]. This means that the value of the unknown rating $\widehat{R}_{ui}$, given by user u to item i, is normally computed using an aggregating function of the ratings of some other (usually, the k most similar) users for the same item i:

$$\widehat{R}_{ui} = \underset{v \in K_u}{\text{aggr}}\, R_{vi}, \tag{2.3}$$

where $K_u \subseteq U$ denotes the subset of k users most similar to user u. The value of k can range from 1 to the number of all users and its optimal value can be obtained using cross-validation. Next, some examples of the aggregation function are presented [Adomavicius and Tuzhilin, 2005; Breese et al., 2013]:

$$\widehat{R}_{ui} = \frac{\sum_{v \in K_u} sim(u,v) R_{vi}}{\sum_{v \in K_u} sim(u,v)} \tag{2.4}$$

$$\widehat{R}_{ui} = \overline{R}_u + \frac{\sum_{v \in K_u} sim(u,v)(R_{vi} - \overline{R}_v)}{\sum_{v \in K_u} sim(u,v)} \tag{2.5}$$

The most common aggregation approach is to perform a weighted average, shown in (2.4), in which weights are given by the similarities between users u and v. However, applying the weighted sum does not take into consideration that different users may use the rating scale unequally. The adjusted weighted sum, in equation (2.5), has been widely employed to address this restriction. The second approach includes the average ratings given by u and v, $\overline{R}_u$ and $\overline{R}_v$ respectively, instead of using the absolute values of ratings, to minimize differences in how users use the rating scale.

Item-based CF

Besides user similarity, the similarity between items can also be considered in recommender systems [Linden et al., 2003; Sarwar et al., 2001]. Item-based algorithms are often preferable, considering that the

majority of systems have a higher number of users than items. Along these lines, the similarity matrix is significantly diminished when using item-based CF. The supra mentioned similarity measures Cosine distance and Pearson Correlation, can also be used in item-based methods.

- **Cosine Similarity**

  In this case, U_i and U_j are the set of users that rated items i and j respectively, as a result, $U_{ij} = U_i \cap U_j$ denotes the group of users that rated simultaneously both items i and j. The similarity between the items i and j is calculated by the cosine of the angle formed by vectors R_i and R_j. The associated coordinates are all the ratings provided by the users to each of the items. Analogously to user-based CF, the item-based variant has the following formulation:

$$sim(i,j) = cos(R_i, R_j) = \frac{R_i \cdot R_j}{\|R_i\| \times \|R_j\|} = \frac{\sum_{u \in U_{ij}} R_{ui} R_{uj}}{\sqrt{\sum_{u \in U_i} R_{ui}^2} \sqrt{\sum_{u \in U_j} R_{uj}^2}} \tag{2.6}$$

  As it occurs in the user-based approach, the cosine similarity for item-based also faces the scale interpretation problem, meaning that different users can have distinct interpretations of the rating scale. Consequently, to overcome this limitation, Sarwar et al. [2001] presented the concept of adjusted cosine. In contrast to the traditional cosine distance, the adjusted-cosine similarity removes the differences in rating scale between different users by subtracting the corresponding user rating average from each co-rated pair [Chen et al., 2009].

$$sim(u,v) = \frac{\sum_{u \in U_{ij}} (R_{ui} - \overline{R}_u)(R_{uj} - \overline{R}_u)}{\sqrt{\sum_{u \in U_i} (R_{ui} - \overline{R}_u)^2} \sqrt{\sum_{u \in U_j} (R_{uj} - \overline{R}_u)^2}} \tag{2.7}$$

- **Pearson Correlation Coefficient**

  The Pearson Correlation Coefficient can be applied as well in item-based CF as a similarity measure between two items i and j. The formulation is described as follows:

$$sim(i,j) = \frac{\sum_{u \in U_{ij}} (R_{ui} - \overline{R}_i)(R_{uj} - \overline{R}_j)}{\sqrt{\sum_{u \in U_{ij}} (R_{ui} - \overline{R}_i)^2} \sqrt{\sum_{u \in U_{ij}} (R_{uj} - \overline{R}_j)^2}} \tag{2.8}$$

  where $\overline{R}_i$ and $\overline{R}_j$ expresses the average ratings given to items i and j, respectively.

Rating prediction Once again, the process of predicting the unknown ratings is parallelly equivalent to items. The prediction of the rating $\widehat{R}_{ui}$ given by the user u to item i is calculated using an aggregating function of the ratings of some other (usually, the k most similar) items for the same user u:

$$\widehat{R}_{ui} = \underset{j \in K_i}{\text{aggr}}\, R_{uj}, \tag{2.9}$$

where K_i denotes the subset of k items most similar to item i. The value of k can range from 1 to the number of all items and its optimal value is typically obtained using cross-validation.

According to Sarwar et al. [2001], the following aggregation functions can be applied:

$$\widehat{R}_{ui} = \frac{\sum_{j \in K_i} sim(i,j) R_{uj}}{\sum_{j \in K_i} sim(i,j)} \tag{2.10}$$

$$\widehat{R}_{ui} = \overline{R}_i + \frac{\sum_{j \in K_i} sim(i,j)(R_{uj} - \overline{R}_j)}{\sum_{j \in K_j} sim(i,j)} \tag{2.11}$$

The equation (2.10) is the weighted sum of the ratings given by u to the items similar to i. In the subsequent formulation (2.11), the weighted sum uses their deviations from the average rating of the corresponding item to eliminate eventual biases on how those items are ranked.

Neighborhood-based CF for positive-only data

The previously stated formulations are designed for explicit feedback, such as numerical ratings. However, as this research deal with positive-only data, this section will introduce the respective formulations for this type of feedback. Neighborhood-based algorithms for positive-only data can be addressed as a particular case of the above, by considering $R_{ui} = 1$ for all the observed pairs of user-item and $R_{ui} = 0$ for the remaining entries in the matrix R. When dealing with positive-only feedback, the problem of the rating scale interpretation of numeric ratings does not verify. Also, given this representation, both notation and implementation can be simplified. Therefore, the most functional similarity measure is the cosine distance, previously declared in equation (2.1). For user-based approach, the formulation can be expressed as:

$$sim(u,v) = cos(R_u, R_v) = \frac{\sum_{i \in I_{uv}} R_{ui} R_{vi}}{\sqrt{\sum_{i \in I_u} R_{ui}^2} \sqrt{\sum_{i \in I_v} R_{vi}^2}} = \frac{|(I_u \cap I_v)|}{\sqrt{|I_u|} \times \sqrt{|I_v|}}, \tag{2.12}$$

where I_u and I_v represent the group of items that are observed with u and v, respectively. Relatively to the item-based similarity, the simplification is equivalent:

$$sim(i,j) = cos(R_i, R_j) = \frac{\sum_{u \in U_{ij}} R_{ui} R_{uj}}{\sqrt{\sum_{u \in U_i} R_{ui}^2} \sqrt{\sum_{u \in U_j} R_{uj}^2}} = \frac{|(U_i \cap U_j)|}{\sqrt{|U_i|} \times \sqrt{|U_j|}} \tag{2.13}$$

In the equation above (2.13), U_i and U_j express the group of users that are observed with i and j, respectively. In these circumstances, the prediction task for user u and item i is performed using either formulation in (2.4) or (2.5), for user-based CF, or equation (2.10), for item-based CF. The value of $\widehat{R}_{ui}$ is no longer a rating, but a score ranging between 0 and 1. This way, it is possible to generate a recommendation list with candidate items and then sort it by descending order for each user.

2.4.2.3 Latent factor models

Besides neighborhood-based techniques, which analyze similarities between items or users, an alternative approach to exploit collaborative filtering data is Latent factor models, which directly profile both users and products. These widely used techniques depend on a learning phase, in which occurs the development of a descriptive model of user preferences based on the observed data to obtain the predictions.

Concretely, Latent factor models perform a dimensionality reduction of the interaction matrix R, in which a group of latent variables is used to illustrate the users' preferences. Such models include a subset of a

larger family of techniques, namely Latent Dirichlet Allocation [Blei et al., 2003], Probabilistic Latent Semantic analysis [Hofmann, 2004], and Matrix Factorization [Koren et al., 2009].

Some of the most propitious accomplishments of latent factor models derive from matrix factorization (MF). In a high-level description, matrix factorization characterizes both items and users through vectors of factors inferred from item rating patterns. When item and user factors present high correspondence, it leads to a recommendation. In recent years, these techniques have become prevalent by merging good scalability with predictive efficiency. Moreover, it offers versatility toward several real-life situations. The following subsection confers a more detailed explanation of how Matrix factorization methods operate.

Matrix Factorization methods

Matrix factorization methods have already proven to be a successful and popular technique in recommender systems, due to their simplicity and generally superior performance in large scale problems compared to neighborhood-based [Shi et al., 2014]. The feedback matrix R usually presents two inherent properties, sparsity and high dimensionality, that motivated the use of such algorithms. Indeed, matrix factorization methods provide forms to obtain lower rank and denser matrices than the original matrix R, with minimal loss of information. Matrix factorization models are closely related to singular value decomposition (SVD), despite being formally different methods. SVD is a well-established method for identifying latent semantic factors in information retrieval [Deerwester et al., 1990], and can be applied to generate low-rank approximations on matrices. In collaborative filtering, implementing Singular Value Decomposition requires factoring the rating matrix. However, the main shortcoming with SVD is that conventional factorization algorithms are not defined for sparse matrices, due to the large portion of missing values. Under the recommendation systems domain, this adverts an even bigger problem, considering that the user-item matrix in CF problems is typically very sparse. In previous researches, developed systems relied on imputation to identify the unknown classifications and, subsequently, obtain a dense classification matrix [Sarwar et al., 2000]. Howbeit, this technique increases the amount of data considerably, leading to an expensive operation.

Matrix factorization models map the users and items to a shared latent factor space with dimensionality k so that user-item interactions are modeled as inner products in that space [Koren et al., 2009].

Assuming a user-item matrix $R_{m\times n}$ with m users and n items, the algorithm consists in decompose R in two full factor matrices $P_{m\times k}$ and $Q_{n\times k}$ that represent all the users and items covering a k-dimensional latent feature space, where k is typically much smaller than the number of users or items.

Given this formulation, R is approximated by $\hat{R} = PQ^T$.

The matrix P crosses the user space, while matrix Q concerns to the item space. The k latent features describe users and items in a common space. For a given item i, the elements of Q_i calculate the extent to which the item owns those factors. On the other hand, for a given user u, the elements of P_u measure the level of interest the user has in items that are high on the corresponding factors [Koren et al., 2009].

The resulting dot product, $P_u \cdot Q_i$, captures the interaction between user u and item i—the user's overall interest in the item's characteristics. This approximates the predicted rating by user u to item i, denoted by R_{ui}, which leads to the estimate:

$$\hat{R}_{ui} = P_u \cdot Q_i \tag{2.14}$$

The training task consists of learning the factor vectors (P_u and Q_i) so that their product accurately approximates the matrix R. This is accomplished by minimizing the regularized squared error on the set of known ratings in the system:

$$\min_{P.,Q.} = \sum_{(u,i)\in D} (R_{ui} - P_u \cdot Q_i^2) + \lambda_u \left\|P_u\right\|^2 + \lambda_i \left\|Q_i\right\|^2 \tag{2.15}$$

In the equation above, D is the set of the (u, i) pairs for which R_{ui} is known. The parameter λ, usually determined by cross-validation, controls the extent of regularization. The system applies the regularization terms $\lambda_u \left\|P_u\right\|^2$ and $\lambda_i \left\|Q_i\right\|^2$ to avoid overfitting the observed data.

Equation 2.15 can be simplified considering that $\lambda = \lambda_u = \lambda_i$, which drives to a single regularization term:

$$\min_{P.,Q.} = \sum_{(u,i)\in D} (R_{ui} - P_u \cdot Q_i^2) + \lambda(\left\|P_u\right\|^2 + \left\|Q_i\right\|^2) \tag{2.16}$$

This optimization problem can be solved using several methods, but the most efficient is Stochastic Gradient Descent (SGD) [Rendle and Freudenthaler, 2014], Alternating Least Squares (ALS) [Bell and Koren, 2007; Hu et al., 2008], and Maximum Margin Matrix Factorization [Weimer et al., 2008].

In 2006, Simon Funk developed a stochastic gradient descent optimization of equation (2.15) that combined implementation ease with a relatively fast running time. In most cases, this successful SGD approach confers a better performance when compared to ALS, even in large and sparse datasets [Koren, 2008; Paterek, 2007].

In Chapter 4, we provide additional information regarding the Collaborative Filtering algorithms employed in the common experimental design of our contributions, namely implicit Matrix Factorization for TV content.

2.4.2.4 Other proposals

An extensive amount of research on Collaborative Filtering problems is focused on neighborhood-based algorithms or matrix factorization methods. Nevertheless, other alternatives have been introduced in the recommender systems domain. Additional neighbors methods contain the probabilistic framework proposed by Yu et al. [2004] or the associate retrieval techniques of Huang et al. [2004]. Moreover, Verstrepen and Goethals [2014] suggested a reformulation of the Nearest neighbors algorithms that merges both user and item-based variants for the one-class CF problem [Pan et al., 2008; Verstrepen and Goethals, 2014]. Regarding latent factor models, the algorithms of this family can also be represented in machine learning techniques such as artificial neural networks [Salakhutdinov et al., 2007], Bayesian networks [Breese et al., 2013], or clustering [Ungar and Foster, 1998]. Bayesian networks denote a graphical model that encodes probabilistic relationships among variables of interest [Heckerman et al., 2008], and probability theory to assign reliability levels [Russell et al., 1995]. Zhang and Koren [2007] proposed a Bayesian model in which users are connected through a root node that also allows the profile of new users to be initialized, solving the cold-start problem. Another example of the probabilistic models used is the Boltzmann machines, a type of stochastic recurrent neural network proposed by Geoffrey Hinton and Terry Sejnowski [Aarts and Korst, 1988; Hinton et al., 1984]. Moreover, Salakhutdinov et al. [2007] and Truyen et al. [2012] applied Boltzmann machines in the recommendation field and provided an implementation of a collaborative filtering system item-to-item. In earlier works, a statistical model for collaborative filtering was described by Ungar and Foster [1998] in which users and respective items of interest are distributed

into clusters with link probabilities associated. Clustering methods are motivated by the premise that it is achievable to group users in clusters according to their preferences. In George and Merugu [2005], a weighted co-clustering algorithm was carried out that involved the simultaneous clustering of users and items, with efficient computational performance and without significant loss of precision.

2.4.2.5 Benefits and Limitations

Contrarily to content-based methods, traditional collaborative filtering employs ratings from other users, which allows it to deal with every type of content. As a result, collaborative systems provide recommendations for any item, even the ones that are different from those seen in the past [Adomavicius and Tuzhilin, 2005]. Given a small number of ratings of a particular user, CF maintains an effective performance since other users' feedback influences what is recommended [Balabanović and Shoham, 1997]. However, collaborative approaches own their limitations, as described below.

Similarly to content-based methods, it faces the new user problem, meaning that the system must first learn the user's preferences from the rating history to be able to produce reliable recommendations. Since collaborative filtering relies solely on users' preferences to provide recommendations, the second problem emerges when new items arrive in the system. Accordingly, until the new item is ranked by a significant amount of users, the CF system would not be capable of recommending it. Hence, collaborative recommendation engines suffer from the cold-start problem, an example of data sparsity when a new user or item enters the system, and there is not enough information to compute a reliable recommendation [Adomavicius and Tuzhilin, 2005; Yu et al., 2004]. When recommendation systems are used for an extensive number of products, the size of the user-item matrix becomes larger and sparser, making it even more challenging to provide recommendations and uphold effective performance in the prediction task.

Finally, Collaborative filtering algorithms are also known to exhibit popularity bias towards recommending items in the long tail. Since popular items bear more rating data that populate the matrix R, these algorithms are more prone to suggest them. As a result, a user with unique tastes compared to the rest of the population will not have other users who are particularly similar, leading to inadequate recommendations [Balabanović and Shoham, 1997]. Most of the techniques that strive to overcome these limitations, use hybrid recommendation approaches, which will be detailed in the following section.

2.4.3 Hybrid recommenders

Hybrid recommendation systems [Adomavicius and Tuzhilin, 2005; Burke, 2002], have been proposed to take advantage of both content-based and collaborative filtering strengths. At its core, hybrid approaches vary in the strategy used to combine recommendation methods from the two classes, for which there is a considerable range of approaches [Burke, 2002, 2007]. Burke [2002] identifies the following techniques:

- **Weighted -** A weighted recommender combines the scores of all algorithms to produce a weighted score. Each algorithm has an associated weight that may be adjusted.
- **Switching -** This technique decides which algorithm to use depending on some programmed criteria. For example, a system may use a fallback algorithm if it does not have the confidence to use the main algorithm.
- **Mixed -** A mixed recommender shows recommendations from multiple algorithms at the same time.

- **Cascade -** Cascade methods use first a recommender algorithm and then another to refine the recommendation.
- **Feature augmentation -** The output from the first recommendation technique is incorporated into the following as an input feature.
- **Meta-level -** Uses the learned model by one recommender as the input to another.

The principal motivation for using hybrid methods is to avoid the limitations of collaborative filtering and content-based recommendation techniques when used individually. As already discussed, a significant drawback of CF algorithms is known as the cold-start problem, so content-based can be applied in these early stages to counterbalance the lack of predictive ability in CF. Adomavicius and Tuzhilin [2005] classified the main trends of hybrid approaches as follows:

- *Combining separate recommendations:* the predictions of separate recommendation algorithms are combined to provide a single recommendation, adopting methods such as linear combinations [Claypool et al., 1999], or voting schemes [Pazzani, 1999].
- *Adding content-based characteristics to collaborative filtering:* Pazzani [1999] adjusted the user-based neighbors method to compute similarities based on content-based user profiles.
- *Adding collaborative characteristics to content-based methods:* latent factor models can be applied to content-based approaches for text recommendation, as in the research of Soboroff and Nicholas [1999].
- *Developing a single unifying recommendation model:* in Popescul et al. [2001] and Schein et al. [2002], is proffered a unified probabilistic method for combining content-based and collaborative recommendations.

2.5 Evaluation Methodologies

Currently, the evaluation of Recommender Systems still denotes a recurring subject of research, representing a vital role in the development of new proposals. Due to the number of factors and complexity involved in a recommendation engine, recommenders systems require assessment through distinct perspectives.

According to Shani and Gunawardana [2011], it is possible to distinguish between three main types for evaluating recommendation systems, corresponding to offline evaluations with historical data, online evaluation and user studies. In this section, we provide an overview of these principal axes in which we can classify the existing evaluation methodologies:

Offline Evaluation In the literature, offline evaluation continues to represent the most widely affirmed technique for validating recommender systems. These testing experiments denote a cost-effective and straightforward method for measuring the performance of several alternative algorithms. The main advantage of offline methods is that they only depend on historical data regarding the user's behavior with the system, so it does not require real participants. This previously collected data provides information about the users' preferences in a specific recommendation domain and can be used to compare various algorithms across a variety of settings. In this context, several recommendation datasets for different recommendation domains are publicly accessible. Amongst them, the MovieLens dataset is probably one

of the more popular ones for movie recommendation [Harper and Konstan, 2015]. Another well-known case of the use of historical data is the Netflix Prize [Bennett et al., 2007], where Netflix published a large-scale dataset to promote the research in the rating prediction problem. However, offline assessment has limited usefulness since it assumes that the user's past actions can model his future behavior. In this sense, it does not consider that data can change over time, disregarding variations in the user's preferences or the user's perception of the recommendations caused by new recommendation methods. Further, this approach is incapable of providing direct information on multiple aspects involved in user satisfaction with the recommendations. Despite all identified disadvantages, offline evaluation continues to characterize a statistically robust method that provides accessible means for assessing the quality of recommendations.

Online Evaluation Online experiments have been extensively used by several businesses and organizations to run controlled experiments on their systems. The purpose of these methods, also called A/B testing, is to measure the direct impact of the recommender system on the end-user. This evaluation is based on the implementation of the recommendation system in a running service and the assessment of the users' behavior. In this case, users are usually unaware that they are being used for testing, making online experiments the source of the most substantial evidence of the system's usefulness. However, conducting evaluations on real systems present some disadvantages for validating recommendation algorithms. First, evaluating systems with low performance can represent an adverse impact on the experience of real customers. Further, to acquire valid conclusions, it is essential to sustain the conditions that will ensure no impact of external factors on the results.

User Studies Performing user studies consists of a direct way of evaluating a recommender system's performance and overall user satisfaction. Test subjects are recruited to interact with the system, and these interactions are observed throughout the experiment. This evaluation methodology collects qualitative analysis and conducts qualitative questions through surveys before, during, or after the test. Compared to offline evaluations, user studies provide more granularity of data and feedback for each user. Nevertheless, they also present several limitations that can restrict their applicability. On the one hand, this kind of evaluation requires high costs in terms of time and resources, since it is necessary to recruit a broad set of participants, which frequently includes monetary rewards. On the other hand, the participant users in the studies must cover different population strata so that they are representative of the real universe of users in the system. Moreover, another shortcoming is that the results can be biased, as users are aware that they are involved in an experiment.

2.5.1 Evaluation Dimensions

In this section, we provide a brief introduction of the general goals in evaluating recommender systems. Most evaluation methodologies in recommendation systems have been oriented towards maximizing the accuracy of predictions, perceived as the recovery of the highest number of relevant items. Nevertheless, a more comprehensive perspective towards recommendation efficiency has revealed the need to improve user satisfaction and the usefulness of recommendations using more dimensions than just prediction accuracy. Properties such as diversity, serendipity, novelty, robustness, and scalability have a massive effect in the sense of a valuable and satisfying experience [Aggarwal et al., 2016]. Later, Chapter 5 describes the evaluation dimensions used in the validation of the proposed recommendation model in more detail.

2.5.1.1 Accuracy

An extensive part of the conducted research in the recommendation systems domain assumes that a successful recommendation algorithm must correctly predict user preferences [Herlocker et al., 2004]. Indeed, accuracy is one of the most fundamental measures through which recommender systems are evaluated. Regarding this context, the most common goal of a recommender system consists in providing predictions for the ratings in the test subset based on the ratings present in the training subset. However, various systems do not perform rating predictions, but instead, they output rankings of the top-N recommended items, which is particularly prevalent in implicit feedback data.

In this sense, the literature on recommender systems distinguishes between two broad categories of measuring recommendation accuracy: rating prediction and ranking [Steck, 2013]. For an extended period, the rating prediction problem dominated as the most used evaluation methodology in Recommendation Systems. More recently, this prevalence has been waning, and methodologies based on rankings have proved to correspond better with real effectiveness.

Rating Prediction: The interactions between users and items are described in the form of an matrix of ratings R, that includes information about the preferences of users U towards items I. This information is typically encoded in the form of numerical ratings, for example, 1 to 5 stars, as discussed in section 2.2. Usually, the evaluation of these predictions is based on error metrics, such as the mean absolute error (MAE), or the root mean squared error (RMSE), which was adopted as the standard metric for the Netflix Prize challenge [Bennett et al., 2007]. For a recommender system S and a test set R_test, the mean absolute error (MAE) and the root-mean-square error (RMSE) is given by:

$$MAE(S) = \frac{1}{|R_{test}|} \sum_{r_{ui} \in R_{test}} |r_{ui} - s(u,i)| \tag{2.17}$$

$$RMSE(S) = \sqrt{\frac{1}{|R_{test}|} \sum_{r_{ui} \in R_{test}} (r_{ui} - s(u,i))^2} \tag{2.18}$$

A lower MAE, or RMSE, means the algorithm produced predictions with less error than a higher value. Comparing these two evaluation measures, the RMSE penalizes large errors in the predictions, characterizing a better reflection of the accuracy in applications where the predictions' robustness for several classifications is essential. However, the RMSE does not show a true reflection of the average error and can increase ambiguity, leading to deceptive results. In opposite, MAE may represent a more appropriate measure in applications where the importance of outliers in the evaluation is limited, as it is less sensitive to distant outliers [Willmott and Matsuura, 2005]. Conclusively, the choice between the validating measures should depend on the system in question. Each technique owns its benefits and drawbacks, and only experimentation can reveal which method works best for a given dataset.

Ranking Task: In this case, the task of a recommender system is to produce a ranking, R_u, for all the candidate items according to the predicted relevance for the user u. This ranking is usually defined by the scores $s(u,i)$ provided for the set of items in descending order. This approach presents the sub-case of top-N recommendation task, in which the system makes a selection of the items in the catalog, meaning a cut-off of the N first ranked results [Cremonesi et al., 2010].

Frequently, ranking-based measures focus on the accuracy of the ranks of the top-N items. Following, we describe some of the most well-known metrics to conduct this evaluation.

Precision and recall at cutoff N represent standard metrics from information retrieval that measure the relevance of a set of ranked recommendations for the user. Given the size of the recommended list N, the set of recommended items is represented by $S(N)$. On the other hand, G represents the test set, denoting the actual set of relevant items consumed by the user. $Precision(N)$ is defined as the percentage of recommended items in the top-N set relevant to the user:

$$Precision(N) = 100 \times \frac{|S(N) \cap G|}{|S(N)|} \tag{2.19}$$

On the other hand, $Recall(N)$ represents the percentage of true positives, items that are eventually consumed, that have been recommended as positive for a list of size N:

$$Recall(N) = 100 \times \frac{|S(N) \cap G|}{|G|} \tag{2.20}$$

A complementary measure is the $F1 - score$ that summarizes both precision and recall by denoting the harmonic mean between them:

$$F1 - score(N) = \frac{2 \times Precision(N) \times Recall(N)}{Precision(N) + Recall(N)} \tag{2.21}$$

A different metric is the mean average precision, $MAP(N)$, which computes the fraction of relevant items in a recommended list of length N and provides information on how relevant the list of recommended items is for a given user. In practice, a higher value of MAP(N) indicates a better performance of the recommender system, given the ground-truth.

Nevertheless, the ranking task is not restricted to rating data, since item rankings can be generated either by explicit or implicit feedback, in contrast to the rating prediction. Moreover, rankings can be evaluated using additional dimensions of quality, such as diversity and novelty, that represent significant roles in evaluating recommendation lists, besides the traditional accuracy criteria.

2.5.1.2 Novelty

Novelty can be generally understood as the capacity of the recommender system to propose unexpected and novel content to the user. In this sense, the recommendation systems denote a more effective and favorable role when they are able to recommend items that the user has never seen before.

$$Novelty = \frac{1}{|U|} \sum_{\forall u \in U} \sum_{\forall i \in topN} \frac{log_2 \left(\frac{count(i)}{|U|} \right)}{|N|} \tag{2.22}$$

The number of users is represented by the absolute U, and $count(i)$ denotes the number of users that consumed the specific item i. Lastly, N is the length of the recommended list.

2.5.1.3 Coverage

The coverage measure is referred to as the fraction of elements for which predictions can be made. Even when a recommender system is highly accurate, it is essential to evaluate the proportion of the items a model is able to recommend.

$$Coverage = \frac{1}{N} \times 100 \tag{2.23}$$

In the equation above, I represents the total number of unique items present in the test data, while N is the number of unique items in the training data.

2.6 Hierarchical Structures for Recommendation

In the recommendation domain, collaborative filtering is the most popular approach. However, the majority of these algorithms work with static data. Li et al. [2007] suggest CVFDT (Concept-adapting Very Fast Decision Tree), a method of deriving decision trees [Hulten et al., 2001] to provide recommendations using collaborative filtering over an incoming data stream that contained user ratings for multiple items. The incoming collected data is used effectively through the development of a decision tree for every item as data arrives. The novelty of this system arises from the use of a hierarchy of items taxonomy, which allows improving the predicted ratings made by each decision tree and, this way, produce efficient recommendations in real time.

In real-world recommender systems, items exhibit certain hierarchical structures, analogously to user preferences. Recent studies in the field have shown that including hierarchies of items or user preferences can expand the performance of recommender systems. Items within the same genre or sub-genre of the hierarchical structure have identical properties, and consequently are likely to obtain similar ratings. This phenomenon has been proved helpful to improve the performance of recommender systems [Lu et al., 2012; Maleszka et al., 2013; Nikolakopoulos et al., 2015; Yang et al., 2016].

Maleszka et al. [2013] studied hierarchical structures of user-profiles for recommender systems. Nikolakopoulos et al. [2015] exploited the intrinsic hierarchical structure of the item space to tackle the data sparsity problem. Indeed, sparsity is an inherent aspect of most recommender systems, and it denotes one of the most challenging problems of collaborative filtering methods [Nikolakopoulos et al., 2013]. In the research of Nikolakopoulos et al. [2015], it was proposed a novel recommendation algorithm, Hierarchical Itemspace Rank (HIR), which exploits the intrinsic hierarchical structure of the item space to lighten the sparsity problem. Using the hierarchies of the underlying spaces, inter-item relations are described at a macroscopic level. Then, the item space is decomposed to define groups of similarly related elements to exploit the indirect proximity features hidden in the composition of the item space. This formulation precedes a corresponding stochastic matrix that can be used to enhance the direct inter-item relationships and reduces sparsity. The resulting model outperforms several state-of-the-art recommendation algorithms, as it revealed to be computationally efficient and of ease implementation, due to the properties of the hierarchical proximity matrix. Yang et al. [2016] proposed a recursive matrix factorization method that uses hierarchies as a regularizer. Also, recent investigations suggest that relationships between sibling nodes of hierarchies could also be valuable [Sun et al., 2017].

However, the approaches mentioned above assume that hierarchical structures are explicitly available in

the real world, and none of them considers implicit hierarchical structures, particularly notable in user preferences.

Wang et al. [2018] cross the problem of exploring the implicit hierarchical structures for recommender systems when they are not explicitly accessible. The work is based on the development of a recommendation framework that enables to explore the implicit hierarchies of both users and items simultaneously. Specifically, it proposes a novel framework IHSR, which incorporates implicit hierarchical structures of users and items for recommendation. A similar structure to hierarchies is ontologies [Middleton et al., 2004], which are very accurate and can improve the recommender system performance. Nevertheless, it is computationally hard, while hierarchies are easier to be used in recommender systems. For the accomplishment of this work, a weighted nonnegative matrix factorization (WNMF) was chosen as the basic model of the proposed framework. WNMF is one of the most popular models in recommendations and has been proven its effectiveness in handling large and sparse datasets [Zhang et al., 2006]. The experimental results on real-world datasets, namely MovieLens100K, the MovieLens1M, and the Yahoo!R2Music datasets, demonstrate the importance of the explicit and implicit hierarchical structures of items and users in the recommendation performance improvement.

Towards a different concept, Li et al. [2019] introduce a technique that learns the hidden hierarchical structure from the records of user-item rating, denoted as Hidden Hierarchical Matrix Factorization (HHMF). Contrarily to earlier proposed methods, this innovative approach does not require to know hierarchical structure beforehand. HHMF can be applied when this information is either explicit or implicit, as opposed to existing hierarchical MF methods. According to the experiments retrieved in the research, the discovery of latent hierarchical structures proved to enhance the quality of recommendations.

2.7 Summary

Increasingly, individuals are faced with large amounts of content from different online systems. In this way, recommendation systems arise from the need to help users search and discover new content from vast catalogs of items that may be relevant to them.

This chapter introduces the main concepts to the recommendation domain by providing an overview of the definitions and the available techniques. Then, we have made a description of the two main types of recommender systems, Content-based and Collaborative filtering. Further, the types of user feedback presented in recommendation algorithms are also described, namely positive-only feedback with typically containing implicit preferences of users. Considering the purpose of this dissertation, we also have provided an overview of the most recent research on algorithms that explicitly deal with hierarchical organizations. Finally, this chapter introduces the necessary concepts to understand all the work and used techniques.

Chapter 3

Data

Modern cable TV companies offer their users access to various services, expanding the different ways to watch television content. Therefore, it is essential to perceive how the users interact to recognize patterns and extract insights into each service, aiming to improve the television experience. In this chapter, we introduce a large-scale characterization of television data, namely the types of content and services available, and respective data sources. In the following sections, we present the steps for developing the dataset on which we evaluate our proposals. The rest of the chapter provides an in-depth exploratory data analysis on the cable TV data, and we proceed to give relevant details on the insights observed.

3.1 Television data

Due to the nature of this project, it is essential to introduce a formal explanation of the main concepts present in a cable TV system. Therefore, this section provides a complete characterization of audiovisual data present in these systems.

3.1.1 Types of content

Regarding television data, it is possible to distinguish between two major types of content: linear and non-linear. Concerning linear content, it corresponds to the programs that broadcast live, meaning they follow a timeline. On the other hand, non-linear TV allows users to select and watch content whenever they wish, which refers the case of VODs or catch-up TV services. The following subsections present extended details about each of the services to improve understanding.

Live TV Represents the traditional way of watching television, where a user can watch any program that is being broadcast live. In TV broadcasting, the viewer must tune a specific channel on a television at a scheduled time to watch a show. Considering that programs can only be watched during its scheduled broadcast, the catalog of items available to the user is highly dynamic. Therefore, recommendation systems for Live TV need to adapt to this condition and have to recommend programs being broadcast at the moment or to recommend scheduled programs.

Video On Demand (VOD) This services allows the clients to watch video content when they choose to, instead of waiting for a scheduled broadcast. This type of media distribution complements the television offer in which a client can watch any video content made available by the cable TV operator, usually a movie or a TV show. Typically, the telecommunications company provides Transactional video on demand (TVOD), where users pay to rent and watch a specific content, and Subscription video on demand (SVOD) packages, in which clients pay a recurring fee to have access to a subset of the content catalog.

Catch-up TV More recently, cable TV operators introduced Catch-up TV, a modern technology that enables users to watch any program that was broadcast live up to a few days before, usually up to seven days. This type of service provides clients a wide number of choices from which they can decide to watch.

Regarding recommender systems for the television domain, most researches focus on video on demand content. In this context, Netflix launched, in 2006, the Netflix Prize [Bennett et al., 2007], which had a significant impact on the state-of-the-art since then. However, the actual application of live-tv data in television recommender systems remains unexplored in the related literature, despite the prevalence of these services. Compared to VODS, live-TV presents several differences and challenges [Gonçalves et al., 2016b], as discussed in the following paragraphs.

First, the items catalog in VOD systems remains mostly static, with a small number of programs added or removed every day at most. Contrarily, in live-TV, hundreds of programs are made available and removed per hour, meaning the catalog is constantly changing due to broadcasting nature. Besides, this characteristic intensifies the cold-start problem considering many programs do not hold any user feedback. Additionally, programs in VOD systems are usually available for a very long time, unlike live-tv services, in which a program is only accessible during its broadcast. Thus, the amount of collected feedback is significantly reduced, and each program has a diminutive time window to be recommended.

Finally, an additional difference between these two services is the main type of feedback collected. Typically, in video on demand systems, users indicate their preferences by giving explicit ratings, such as using a like or dislike button. On the other hand, the portion of explicit feedback on live-TV is much smaller, making them impractical for learning user preferences. Nonetheless, implicit feedback is mainly available and used instead, particularly when a user watched the video content more than a certain percentage of visualization time.

3.1.2 Data Sources

The development of a television recommendation system requires the acquisition of an enormous amount of data. Hence, it is necessary to collect data from television content from several information sources [Gonçalves et al., 2016a]. The topics introduced below describe the different data sources and the information collected from each one.

Electronic Programming Guide (EPG) The EPG contains the catalog of items to recommend, including the scheduled programs to broadcast. Commonly, this sort of data represents an EPG for live TV and lists for VOD content. These items often have linked metadata, such as the corresponding category, broadcast date, a short description, or even the characters involved.

User Ratings Relevant television data can also be obtained through the ratings that users give to watched programs, referring to explicit feedback sources.

User Views The data about visualization times in television sessions represent the largest source of feedback from the users. This information denotes an implicit source of data since no additional action from the users is necessary besides their channel selections.

User Metadata Many distinct types of user metadata exist and can include information about the interests and demographics of the users that interacted implicitly or explicitly with the system.

3.2 Dataset

In this section, we present television data collected from a large Portuguese telecommunications and media company. This cable TV operator delivers a set of linear program television, Catch-up TV, and VOD services through Set-Top Boxes (STB) installed in the customers' homes. Users tune the corresponding broadcasting channel to watch live TV content. Catch-up TV content can be accessed by selecting a program through the Electronic Program Guide (EPG), present in the STB, up to seven days after it broadcasted live. Finally, to watch VOD, clients need to select a program from the VOD catalog.

In this case, the source of user feedback arises from the visualization data of the television sessions. This type of feedback consists of an implicit source of information since there is no need for additional action from the users besides their channel selections. A user viewing session corresponds to information about the periods they have been tuned to a specific channel. This knowledge associated with the items' broadcast date is used to determine how long each user watched a particular program.

Although the necessary data sources derive from the television domain, the available raw data was scattered and some fields were not documented, requiring much research to proceed to the extraction process, which became a lengthy, but imperative, stage of this dissertation.

For the remaining elements of this section, we provide details about the developed dataset used in the proposed recommendation system. Initially, we explain how data is collected and then processed. We then focus on its main characteristics: the number of users and items, the nature and quantity of interactions between them, and additional worth-mentioning properties.

3.2.1 Collecting Raw Data

As previously stated, recommending television content requires access to varied types of data. Therefore, in collaboration with the telecommunications company, it was necessary to obtain a vast amount of raw data that ensured the construction of a suitable and reliable dataset to input during the modeling phase.

In the first place, it was necessary to verify among several tables from different database management systems which data could be included in the dataset. The majority of data in the tables was incomplete, so we had to identify and combine different tables to obtain reliable content and ensure high-quality data.

Finally, all the information discovered in the course of this analysis stage was duly cataloged and documented. Indeed, it was built an organizational structure to make raw data understandable and

guarantee the television data organization, non-existent until then. Moreover, this data structure allows the reuse and integration of this type of data with any existing system that needs this information, just through the access of a single database.

After collecting all the available data, we then had relevant information about each type of content. In video on demand services, we had access to metadata containing pertinent information for the personalized recommendation of live programs in cable TV. This list includes information about the title, category, synopsis, release date, or even the IMDB rating for each program. On the other hand, Live and Catch-up TV hold the Electronic Program Guide (EPG) data, which corresponds to the description of what is displaying on each channel at any moment of the day. Added metadata includes the category, subcategory, channel, title, actors, directors, season, episode, and broadcast time.

Initially, the goal was to apply these three types of content in the input for modeling. However, after collecting all the available data, it was possible to verify that there was not enough data from VODs or catch-up TV. One of the reasons for this condition was that the data provided is obtained by a former set-up-box, meaning it presents a weaker interface and quality of interaction. Thus, as the available data from non-linear content would not be representative of the population, it was determined to use only live-tv data. On the one hand, linear television is illustrative of the majority of contents, becoming an enthralling field of study due to its singular characteristics, as previously stated in section 3.2.1. On the other hand, the non-linear recommendation handles a static catalog of items, representing a well-known investigation at the scientific level.

3.2.2 Data Preparation

Following the data collection phase, the information was aggregated and pre-processed in order to obtain an organized and structured dataset to implement in the recommendation model. This section presents our television dataset and explains the process that takes the raw data from the data sources and converts it to the format used by the recommender system, presented in Table 3.1.

Data Type	**Parameters**
User	Mac Id, List of Sessions, List of Programs
Program	Program Guide Id, List of Properties, List of Users
Session	Session Id, Mac Id, Start Date, End Date, Duration
Property	Property Id, Property Type, Property Value

Table 3.1: Structured data after preprocessing stage

Our cable TV dataset consists of television visualization data, limited by a time window corresponding to a period of around four weeks, from the 1st of October 2019 to the 31st of October 2019. The month's choice was due to the fact that it represents a more neutral phase compared to others, where seasonality can create inconvenient trends for this research. Each user in this dataset represents a MAC id from the set-top box, and each item denotes a television program. As previously discussed, the source of user feedback arises from the visualization data of the television program. Finally, items can have multiple properties, representing a piece of metadata associated with a program. The metadata for each program

corresponds to its title, channel, category, subcategory, and broadcast date.

For this project, it was necessary to collect information from the EPG and cross-check it with the information on what users are watching. This original data is polled for each MAC address every 60 seconds unless an interaction occurs (e.g. TV channel/TV program changes). Considering the massive quantity of information, it was essential to perform aggregations per day to make the process computationally feasible and in a reduced time. Thus, a monthly data table was obtained with information on the total number of seconds seen by each user throughout the day for each program broadcast live.

The final structure of the table to incorporate in the model contains one row per customer interaction, and a customer can have several interactions. This matrix allows the analysis of customer consumption for each variable, according to the viewing time or the number of programs seen. Finally, all the aggregate information per day implies a large volume of data with millions of lines and dozens of variables, as described in the subsequent section.

3.2.3 Data Analysis

In the present section, we start the characterization of the evaluation dataset by performing an extensive exploratory analysis on live television viewing data. For the first approach to exploratory and descriptive analysis of television data, we present below an overview of the linear content statistics.

During the collection period, we obtained TV session data from around 700,000 distinct anonymized Mac IDs. However, as it proved to be a computationally demanding process, we decided to use a representative sample of the number of users of the television service. Therefore, the data comprises 10,000 clients, randomly chosen with the only condition that they watched at least ten programs per week.

In total, there are 11489 programs in the live TV system. A program is categorized into one of the following 8 categories: Entertainment, News, Movies, Sports, TV Shows, Kids, Documentaries, or Adults. The Live TV catalog contains 118 subcategories in total. For instance, considering the Movies category, some possible subcategories are comedy, action, or drama. If we measure the programs available in a moment in time, Live TV has 178 programs since it is the number of channels available to the user. Each program has one or more episodes. For instance, a movie has only one episode, whereas television shows can have multiple episodes. Linear content that broadcasts daily, such as news, is also considered a television show with several episodes.

For the knowledge domain of this project and a clear perception of the analysis conducted further, it is fundamental to highlight the difference between two main terms: interactions and views. Interactions correspond to programs the user watched or passed through; in other words, the action of changing a television channel or program. On the other hand, a view is defined as a program watched consecutively for more than 5 minutes.

Finally, a summary of the data in this television dataset is available on Table 3.2.

Users	10000
Programs	11489
Categories	8
Subcategories	118
Channels	178
Interactions	4697080
Average user interactions (day)	17
Views	1963070
Average user views (day)	9

Table 3.2: Live-TV dataset statistics

Next, the performed analysis is distributed into different domains contributing to a more comprehensive and intelligible characterization of Live TV.

3.2.3.1 Usage Distribution

The current subsection describes service usage by measuring and comparing the usage of services on a typical Cable TV provider.

First, figure 3.1 shows the distribution of Mac IDs by aggregating daily viewing times, with bins corresponding to one hour of visualization. In this case, all program interactions with less than five minutes were removed.

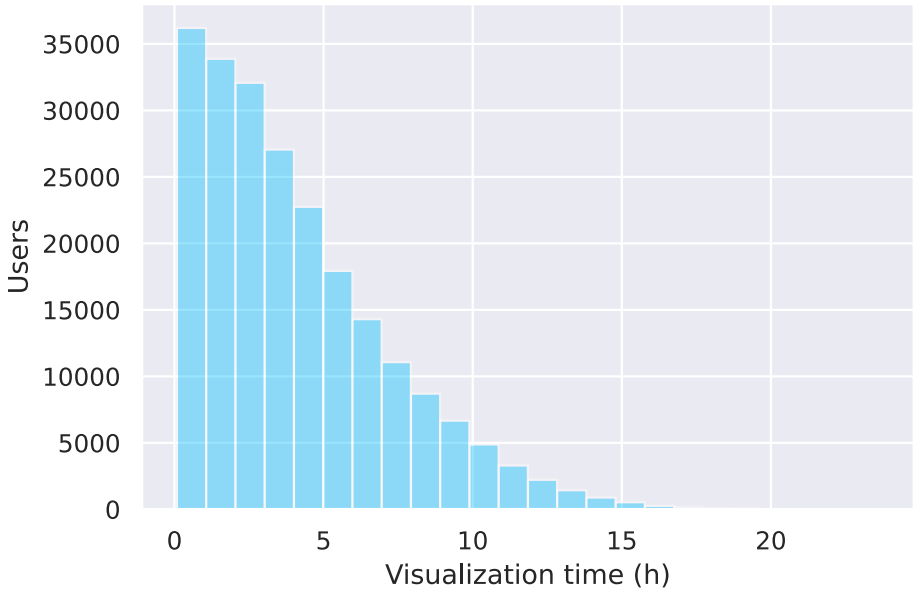


Figure 3.1: Distribution of users – Total Visualization Time

The graph displayed in Figure 3.2 show the number of watched hours per day in linear television service.

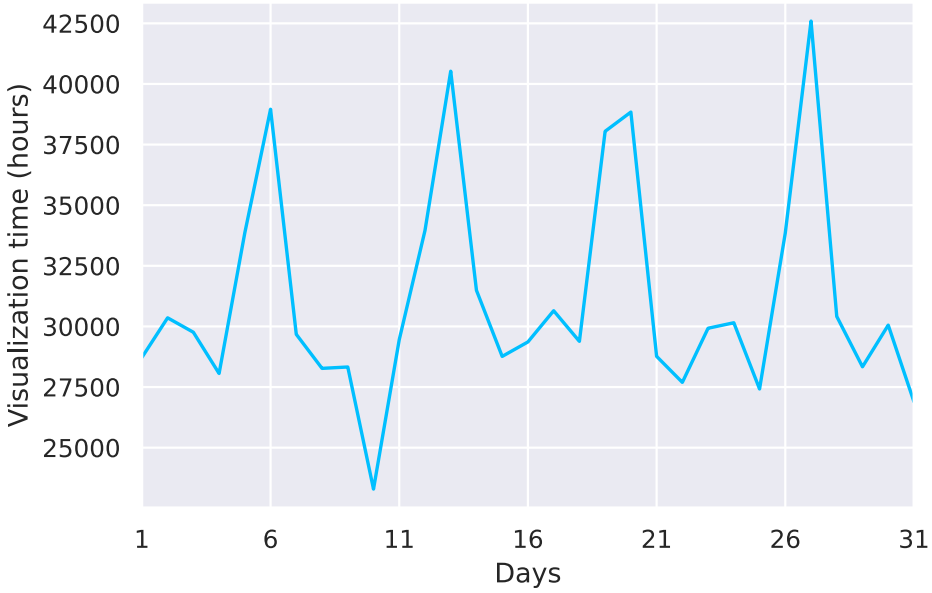


Figure 3.2: Number of hours watched per day in Live TV

According to the analysis, users spend an average of almost four hours a day watching television, which denotes a significant amount of time in people's daily lives. In this sense, the usefulness of recommendation systems in the field of traditional television is increasingly demonstrated.

Then, figure 3.3 displays the number of Live TV views per day over one month. The graphs' peaks occur at the weekends, suggesting that users consume more television content in those days. The following plot, figure 3.4, shows the number of interactions, which reveals similar conclusions.

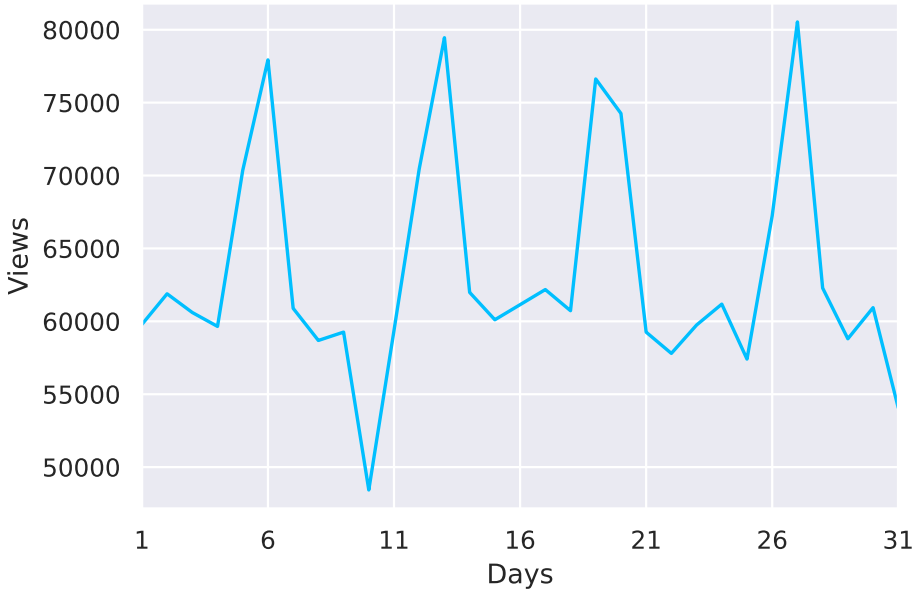


Figure 3.3: Number of views per day in Live TV

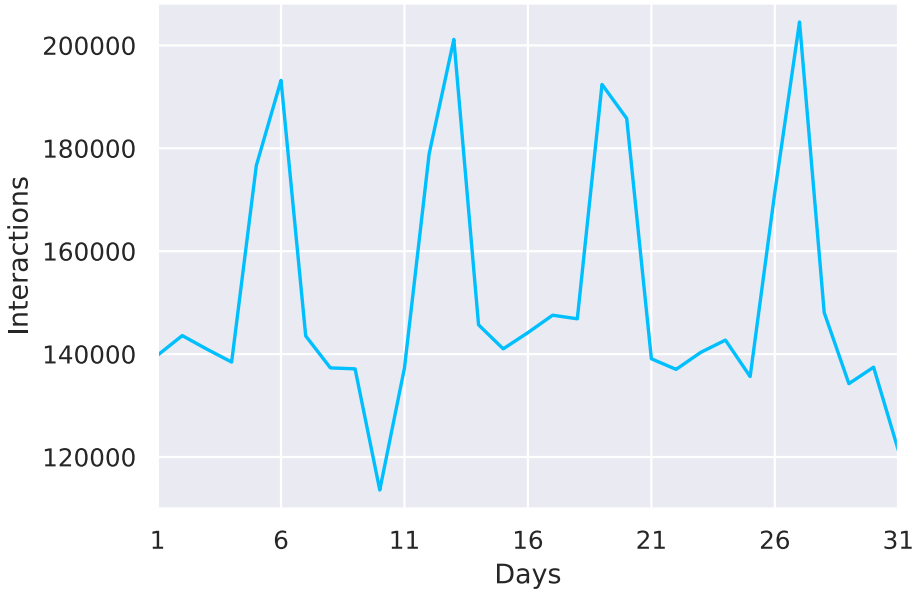


Figure 3.4: Number of interactions per day in Live TV

Lastly, figure 3.5 exhibits the distribution of usage activity per period of the day.

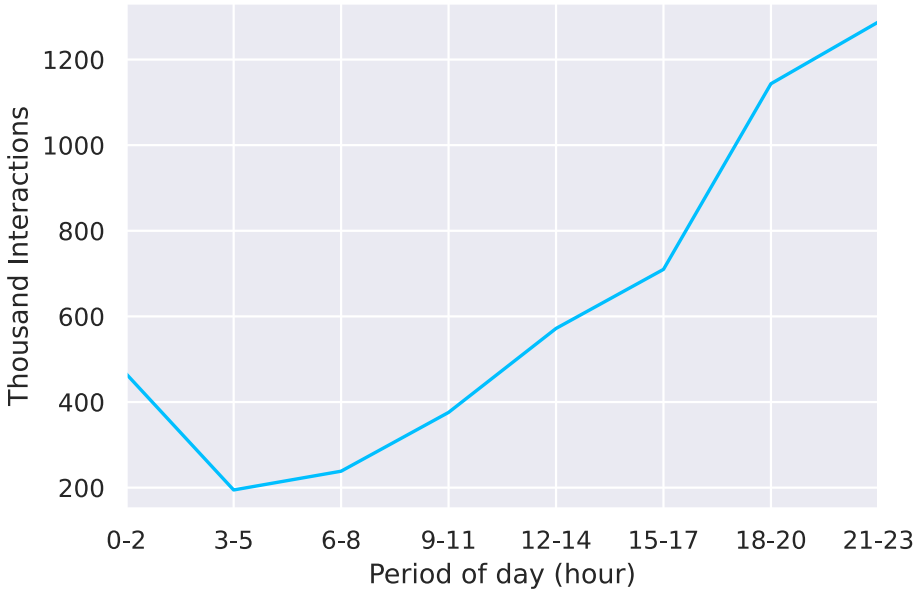


Figure 3.5: Number of interactions per period of day

The graph reflects the standard working and sleeping cycle of individuals. As shown, most views occur around 6:00 PM to 12:00 AM, usually when people are at home after work and not sleeping.

3.2.3.2 User Engagement

In this segment, we evaluate how the users watch television in live service. Figure 3.6 depicts the percentage of content visualization for live TV in this Cable TV provider. The x-axis represents the percentage of the program visualization time, and the y-axis shows the number of thousand interactions that had that percentage of visualization.

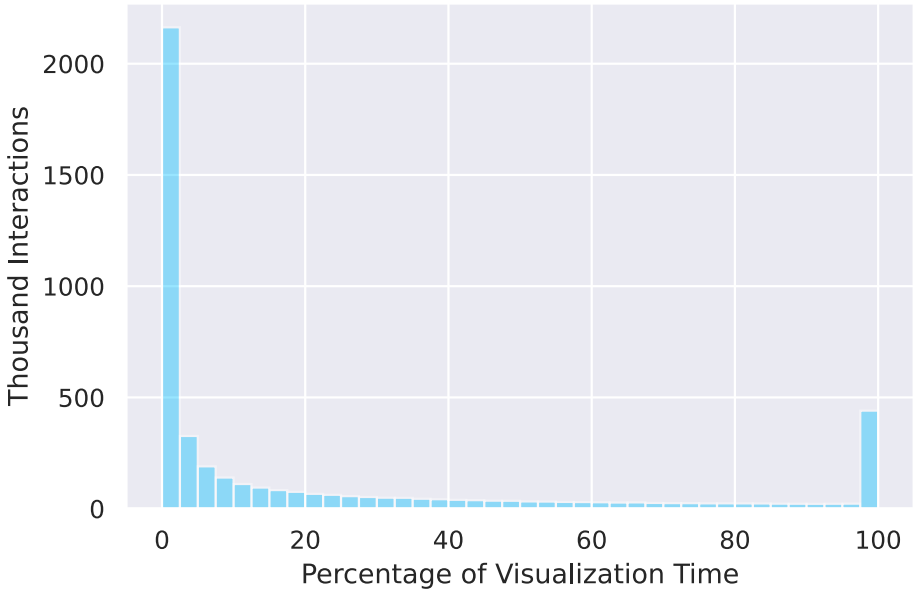


Figure 3.6: Distribution of percentage of visualization time

As demonstrated, Live TV has lower retention than other services, due to the possibility of channel surfing, also known as zapping. Considering that the main content of a television program ends at approximately 90% of its total duration, with the remaining part being advertisements or closing credits, the rate of complete program views in Live TV is about 10%.

Finally, figure 3.7 shows the percentage of content visualization by program category for Live TV. This kernel density estimate (KDE) graph describes the data using a continuous probability density curve.

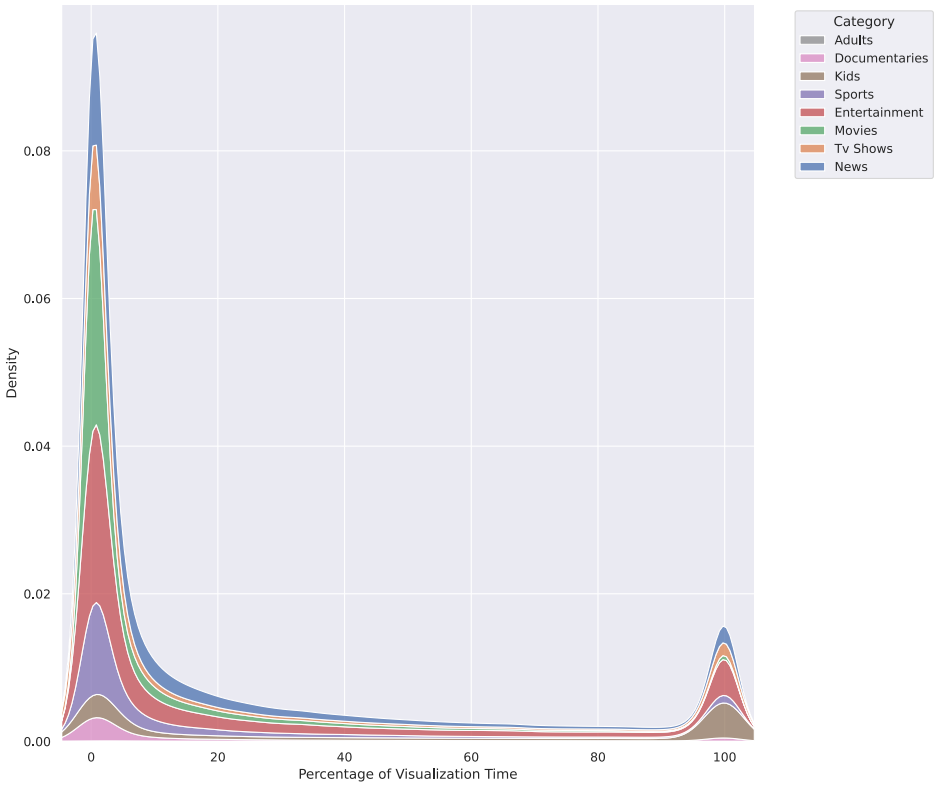


Figure 3.7: Percentage of visualization time per category for Live TV

Through this plot, several considerations can be interpreted. Kids' TV shows hold a lower dropout rate. For instance, we can also verify that Entertainment programs presenting the highest dropout rate, but also one of the highest full session rates.

3.2.3.3 Program Types

This section explains the available program types in linear content considering their dynamic nature. Also, we measure and discuss how large is the impact of the cold-start problem. First, it is possible to distinguish two main groups in the program context: new or repeated. Then, it is possible to characterize different contexts in live television programs [Gonçalves et al., 2016a]:

- New program: never broadcast before, meaning there is no information on user preferences about them (item cold-start problem);
- Program broadcast before, but not seen by a user: for these, we have information regarding other users that watched it;
- Finally, program broadcast before and watched by a user: corresponds to a new episode or a repeated program;

Figure 3.8 shows the distribution of main program types made available by the Cable TV operator in the catalog per day.

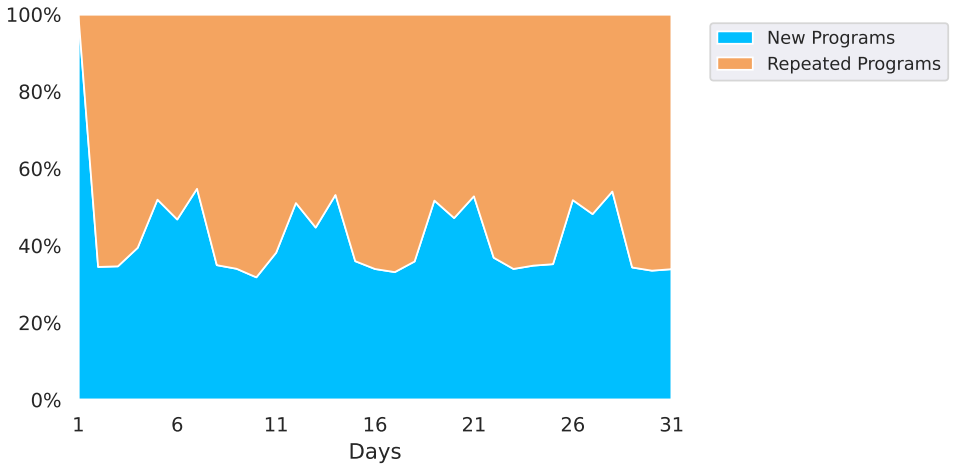


Figure 3.8: Distribution of program types per day

On average, 43% of programs broadcast live every day are new, and the remaining 57% consists of repeated programs or episodes. Due to the analysis beginning on the first day of the dataset, it is worth mentioning that the first week of data has content classified as new that otherwise would be identified as repeated. Therefore, the values obtained in the remaining weeks of data more accurately represent the typical situation found in a traditional TV operator.

3.2.3.4 Program Categories

In this section, we provide a comparison of user visualizations concerning different program categories.

As we can observe in figure 3.9, users consume approximately 94 hours of television content per month. Besides, entertainment corresponds to the leading category in this analysis, with users spending about 30 hours a month watching this type of content.

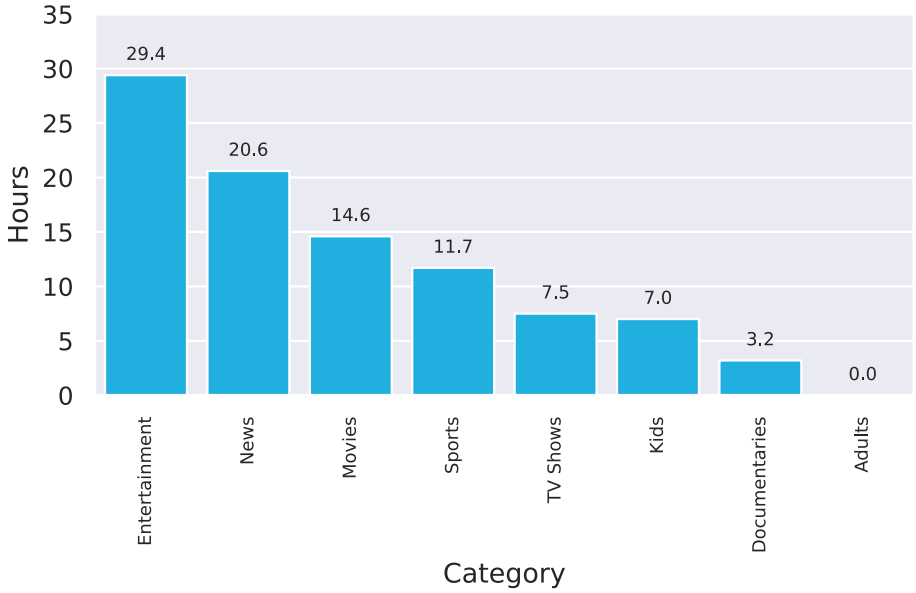


Figure 3.9: Number of hours watched per month in each category

In the same context, figure 3.10 shows the distribution of visualization programs according to categories by day in a month.

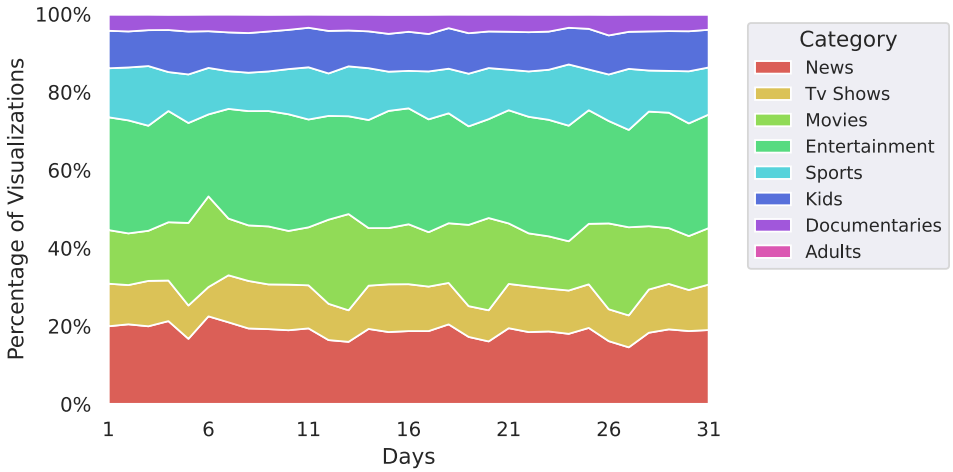


Figure 3.10: Program category distribution per day

Indeed, we can observe that most of the programs seen belong to the entertainment category, representing 28% of the total, followed by information and movies.

Finally, we also evaluated the impact of the cold-start problem on live television. The conducted analysis proved that the average percentage of new programs in the catalog per week is 25%. To complement this

assessment, we also decided to examine this problem for each live TV category, displayed in table 3.3.

News	29,6%
Tv Shows	27,7%
Movies	78,6%
Entertainment	35,8%
Sports	64,7%
Kids	26,4%
Documentaries	54,7%
Adults	69,8%

Table 3.3: Percentage of new programs by category

Table 3.3 shows the average percentage of new programs in the catalog per week by category. It reveals that a large percentage of Movies and Sports programs are new. We can also observe that most programs belonging to categories of News, Kids, and TV Shows are recurring, meaning most of the contents are new episodes or repeated ones.

3.2.3.5 Final Considerations

After the characterization of traditional cable TV and the throughout exploratory data analysis, it was possible to verify that live television data holds specific characteristics that need to be addressed before entering the modeling stage. As mentioned above, the source of user feedback corresponds to the visualization data of the television sessions, denoting implicit feedback. In regard to implicit feedback, it provides an indirect means of determining the user's preferences for the items. In the case of television data, the source of user feedback arises from the interactions extracted from the set-up box, which discriminate the programs' viewing times for each user. An implicit ranking can then be calculated using information about the user's interaction with the system, for example, how long users remain watching a particular program.

In the television visualization domain, two practices need to be strictly discarded from the dataset, as they represent neutral feedback: channel surfing and radio effect. On the one hand, channel surfing, also called zapping, consists of small viewing sessions and corresponds to the practice of quickly scanning through different television channels to find something interesting to watch. This type of feedback is discarded since it may represents a neutral interaction; the user quickly passed through the program, which may not mean negative feedback. On the other hand, the radio effect indicates extensive viewing sessions, usually on the same channel, which corresponds to the behavior when users leave the television on without actually watching. Consequently, an exhaustive study was carried out to make decisive choices for obtaining the final dataset.

First, to discard viewing sessions that represented channel surfing, we set a threshold of 10 seconds based on the analysis performed. Thus, very short full TV sessions (=<10 seconds) were removed. Considering the second enumerated problem, we assumed that continuous sessions on the same channel, without any interaction with the set-up box, would indicate that the user is not actually watching any program.

Therefore, we decided to remove users with more than 10^6 seconds in one month, equivalent to more than 9 hours per day.

After removing interactions that could negatively impact the quality of recommendations and limit the usefulness of the recommender system, the obtained dataset provides information that can be classified into unary or relevance feedback, as it will be discussed in the next chapter. To summarize, the final data culminates with 9872 users and is in the form $< user, program, relevance, metadata >$, where the relevance field indicates the user's preference level to the program.

To conclude this section, it is worth mentioning that the process conveyed for the development and characterization of the dataset was performed under appropriate privacy policies, without any personally identifiable information, as all the information was entirely anonymized.

3.3 Summary

Nowadays, cable TV services provide access to multiple sources of television content, introducing distinct experiences to their customers.

In this chapter, we start by describing the type of data present in television services, namely Live TV, Catch-up TV, and Video on Demand (VOD). Then, we describe the process of building a solid and reliable dataset consisting of television visualization information through different data sources. As a fundamental stage of this dissertation, we present a comprehensive TV usage characterization on a large-scale Portuguese telecommunications company, focusing on linear services. The in-depth analysis allowed an extensive study to understand the inherent characteristics of television data, namely how users interact with the television systems, the type of programs available, and user visualizations concerning different program categories. Besides, we also verified the massive impact of the cold start problem in the linear television domain. The information obtained consists of essential properties for the construction of a high-quality data set, aiming to further improved recommendations. Finally, we discuss some of the well-known characteristics of Live-TV, in particular, channel surfing or radio-effect, and the importance of discarding such interactions.

Chapter 4

Hierarchical Matrix Factorization

Chapter 2 presents a broad overview of the perspectives on hierarchical information in Recommender Systems. In recent years, there has been a significant expansion of research in the field of hierarchical recommendation. However, the research of hierarchical recommendation for television content remains largely unexplored. In this dissertation, we address such recommendation-specific problems and propose solutions for their enhancement, considering a different outlook on the problem of hierarchical organization.

This chapter provides a formal basis for introducing hierarchical information through a matrix factorization algorithm using different perspectives to increase quality measures, such as novelty of the recommendations. Thus, we propose a formal structure that unifies and generalizes the entry of several hierarchies, aiming to improve the current systems through advantageous properties not present in the previously reported algorithms. These characteristics are introduced by considering how users interact with the recommendations, without losing focus on the main items, but always keeping in mind the user's satisfaction and the expanding need to deliver new and diverse recommendations that the user also considers relevant. Specifically, the evaluation methodology of our proposals considers the ranking and relevance of the recommended items and follows a line of comparison of all systems through a reliable architecture. Our proposals intend to leverage hierarchical information as an essential component to increase the novelty and diversity of recommendations on a few ground concepts and formal models.

The structure of the rest of this chapter is as follows. Section 4.1 describes the formulation and implementation of the matrix factorization recommendation algorithm that served as a baseline for the evaluation and further improvement concerning the recommendation of novel and diverse content. In Section 4.2, we describe hierarchical knowledge in recommendations as the central concept that builds our formal model. Then, Section 4.3, details the integration of explicit hierarchical structures into the modeling phase. For Section 4.4, we propose the hierarchical model that addresses the cold start problem, which is the main component of our proposal to optimize the inherent characteristics of the collaborative filtering baseline and include novelty in recommendations.

4.1 Baseline Matrix Factorization

In particular, this section presents a baseline recommendation algorithm used in the common experimental design of our contributions. For the remainder of this project, we have centered our research on collaborative filtering approaches, due to their strong performance and efficiency in handling large and sparse datasets.

In this work, we choose matrix factorization (MF) as the basic model of the proposed experimental design, one of the most popular latent factor models to develop recommender systems [Koren et al., 2009]. Therefore, the remainder of this section describes the matrix factorization model used to evaluate the experiments proposed in the following sections.

As explained in more detail in Chapter 2, the basic idea of MF based models consists of a dimensionality reduction of the interaction matrix R into a two user and item matrices representing all the users and items in a k-dimensional latent vector space. Such user latent feature matrix (P), captures user preferences, while the item latent feature matrix (Q) captures the properties of the item. The matrices P and Q are obtained by minimizing the L2-regularized squared error for known values in matrix R, and the predicted ratings:

$$\min_{P.,Q.} = \sum_{(u,i)\in D} (R_{ui} - P_u \cdot Q_i^2) + \lambda_u \left\|P_u\right\|^2 + \lambda_i \left\|Q_i\right\|^2 \tag{4.1}$$

Using Stochastic Gradient Descent (SGD) optimization, the input of the MF algorithm is given by the tuple (u, i, r), that corresponds to a rating r given by a user u to an item i. In this case, matrix factorization is designed to work with positive-only data, by assuming that $r = 1$. In the end, the resulting estimate $\hat{R}_{ui} = P_u \cdot Q_i$ determines a user's preference level for an item. Since we are interested in the top-N recommendation, we need to retrieve an ordered list of items for each user. Therefore, we sort candidate items i for each user u using the function $f_{ui} = \left|1 - \hat{R}_{ui}\right|$, where $\hat{R}_{ui}$ is the predicted score, and then select the higher scores for each user.

For each dataset, the specific parameters for the recommendation algorithms were manually chosen to optimize the relevance of the recommendations, and are the following:

- number of latent features k: 100;
- regularization parameter λ: 0.1;
- number of iterations: 10.

Regarding the application of the most trivial baselines, we decided to choose a simple popularity-based model and a random recommender, arguably the simplest recommendation methods to overcome in a ranking problem [Amatriain, 2013].

From these general basis models, which provide non-personalized recommendations, it is possible to put into perspective the fundamental domains of novelty in the more sophisticated algorithms.

4.2 Hierarchical Information

Most items generally present a hierarchical organization structure that allows their representation in several categories. This section introduces hierarchical item categories, obtained from implicit feedback data, to enable relevant recommendations. Using this pivotal information, we propose a Hierarchical Matrix Factorization model (HMF) that incorporates the inherent hierarchies of television content and respective relations to model the item vectors.

Accordingly, the lowest level in the hierarchy means a group of items that form the most specific categories, while higher layers hold more generic classes, which are groups of these specific categories and so on. In this television content domain, items are classified into a hierarchical structure organized as category→

subcategory→program. In addition to the items' natural hierarchical structures, the preferences of users also present hierarchical structures, which have been used in decision-making problems [Moreno-Jimenez and Vargas, 1993]. According to the hierarchical structures of items, television content under the same hierarchical layer are likely to share similar features, so they are expected to be at the same level of the user preference. In this setting, the hierarchical structures are explicitly available, allowing more structured and robust information for guiding the process of learning latent features than implicit hierarchies [Wang et al., 2018].

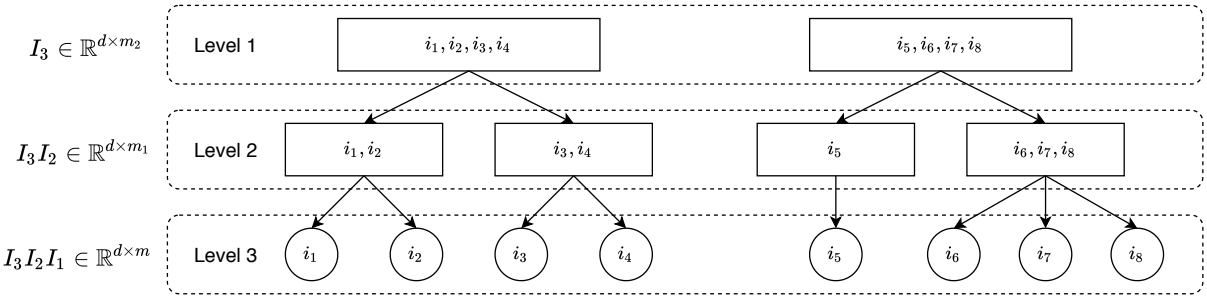


Figure 4.1: Illustration of Hierarchical Structures

In figure 4.1, we can observe an example of an explicit hierarchical structure present in a television data system. A hierarchical structure can be represented as a set of trees, with nodes at the same level in the tree denoting a group of items that possess similar properties. The set of items in a child node represents a subset of items in its parent node. Each parent node has one or more child nodes, and each leaf node represents a television program.

In the example illustrated in figure 4.1, two major categories in the live television content are represented in the two trees displayed, respectively 'Kids' and 'Sports'. On the left side, the hierarchy structure corresponding to the 'Kids' category (Level 1) has associated five specific television programs $\{i1, i2, i3, i4\}$, that belong to two subcategories (Level 2). From this observation, we can see that overall a hierarchical structure denotes a set of a parent-child relationship.

A hierarchical structure can be acknowledged as a collection of trees, where each node represents a group of objects that share particular features. The items included in a child node are a subset of the items presented in its parent node. Consequently, a child node captures more granular and in-depth properties for a smaller set, while a parent node captures more comprehensive properties for a broad set. As regard to leaf nodes, they capture the peculiar characteristics of each item.

Each node of the hierarchical trees is represented using the literature notation Wang et al. [2018]: we identify p as the number of levels of the hierarchical structure, then $I_p \cdots I_k \in \mathbb{R}^{d\times m_k}$, where $k \leq p$ represent the m_k nodes in the level $p-k+1$ of a hierarchical structure. Regarding the illustrated example in Figure 4.1, $I_3 \in \mathbb{R}^{d\times m_2}$ gives the latent representations of the two root nodes. On the other hand, $I_3 I_2 \in \mathbb{R}^{d\times m_1}$ with $m_1 = 4$ denotes the latent representations of the four nodes. Lastly, $I_3 I_2 I_1 \in \mathbb{R}^{d\times m}$ with $m = 8$ indicates the latent features of the eight leaf nodes.

4.3 Modeling Hierarchical Structures

As previously stated, the scope of this dissertation underlines on the integration of explicit hierarchical structures into modeling to achieve a more efficient and useful recommender system. This section describes the implementation of the proposed models that apply the hierarchical information of the items, including a formal comparison between all the different approaches. Regarding the data collected, it is possible to identify different hierarchies existing in the television service. In the course of the experiments, we decided to use a trivial and easily comprehensible hierarchy: category→subcategory→program.

In this case, the category level corresponds to the root of the tree, whereas the program denotes the lowest level in the hierarchy and, therefore, depicts the leaf nodes. The second layer of the tree corresponds to the program's subcategories, which indicates the intermediate level of the hierarchy. Thus, to provide recommendations at the level p of the hierarchy, we propose two distinct models, on the basis of collaborative filtering techniques. In Section 4.2, we have described the basic matrix factorization model as a representative and state of the art ranking based method. In this sense, it details a specific baseline model that only considers the p-level of the tree. Accordingly, to predict the level p of the hierarchy, the training dataset includes information only regarding the items in the p-level.

Considering the baseline algorithm above, we then developed a model that already includes a source of hierarchy information, albeit in a trivial procedure. Nonetheless, these steps are required to perform a robust evaluation of the actual effectiveness of the hierarchical organization. Figure 4.2 is an illustration of this model, referred to as $Model1$.

Thus, if we intend to recommend at the level p, this proposal includes in the training phase all the items positioned at the level below of the hierarchy, $p + 1$. In this case, there is an aggregation of the values of the child nodes to the respective parent nodes, meaning the user feedback is propagated upwards in the tree. As previously denoted, the implicit matrix factorization model provides scores to determine a ranking, revealing the user's affinity for a particular item. The central concept of this proposal is to apply the aggregator measures on the obtained scores at any given hierarchy level p, except in the case of the most specific items (end-nodes).

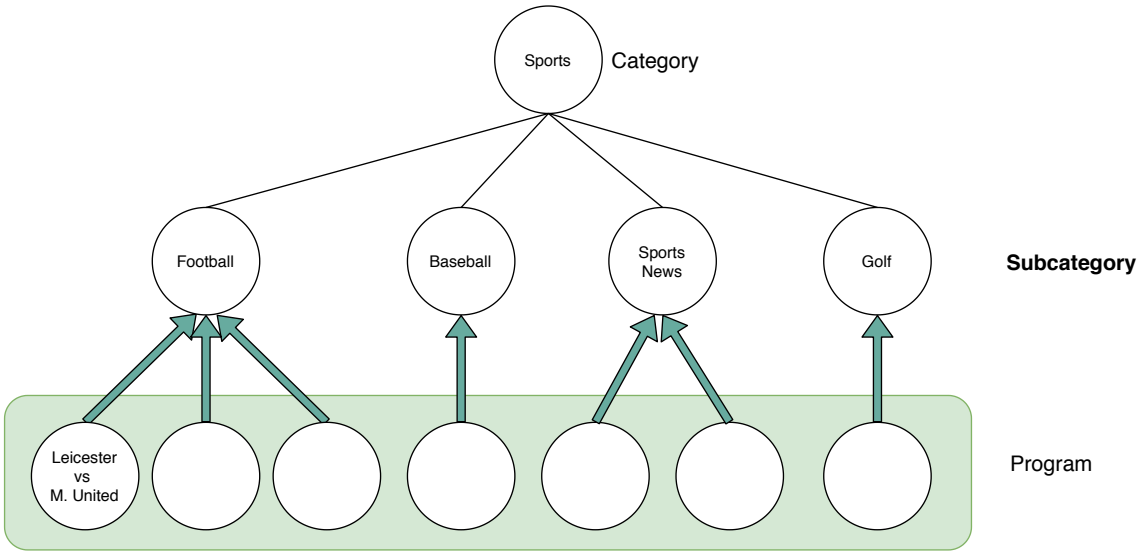


Figure 4.2: An Illustration of the Model 1, with level N = Subcategory

For example, suppose we want to determine the preference of a particular user towards the "Baseball" subcategory. The basic notion is to use the score of the levels below that subcategory, regarding all "Baseball" programs, and then aggregate to determine a score of the level above. The aggregators chosen to propagate the information of the child nodes to the corresponding parents on the hierarchy can be stated as follows:

- sum: represents the globality of what the user likes at level p, considering the entire universe at the level below;
- mean: determines the user's preference concerning the level p, summing up all the scores obtained in the levels below and then dividing by the number of child nodes (of items) of that level p;
- maximum: the idea of this aggregator is upon the assumption that the user likes the item at level p as much as the highest preference result of the item at the level below, $p + 1$. In other words, there is a particular item at the level of the child nodes that the user appreciates so much that this specific item represents the user's preference towards the p level;
- minimum: denotes the opposite concept of the maximum aggregator; In this case, it means that the user likes the item as much as the lowest preference score in relation to the child nodes. The underlying idea is that if the user had a bad experience concerning a particular item, it generates aversion to the remaining items of that level.

Finally, we developed the hierarchy propagation model, a more comprehensive proposal that unites the full information of the datasets and trains with all objects of the hierarchy simultaneously (child and parent nodes). Consequently, this proposed model allows queries to be made at all existing levels, across two distinct variants, in line with the models mentioned above: the possibility of recommending at the specific level p ($Model2a$), or performing aggregations at the level $p + 1$ ($Model2b$). The proposal for this

framework is based on the conception that it can take full advantage of the hierarchical structures, across a regularization of holding all the levels within the same model, which means that the latent features of child and parent nodes must be shared. This model is illustrated in Figure 4.3.

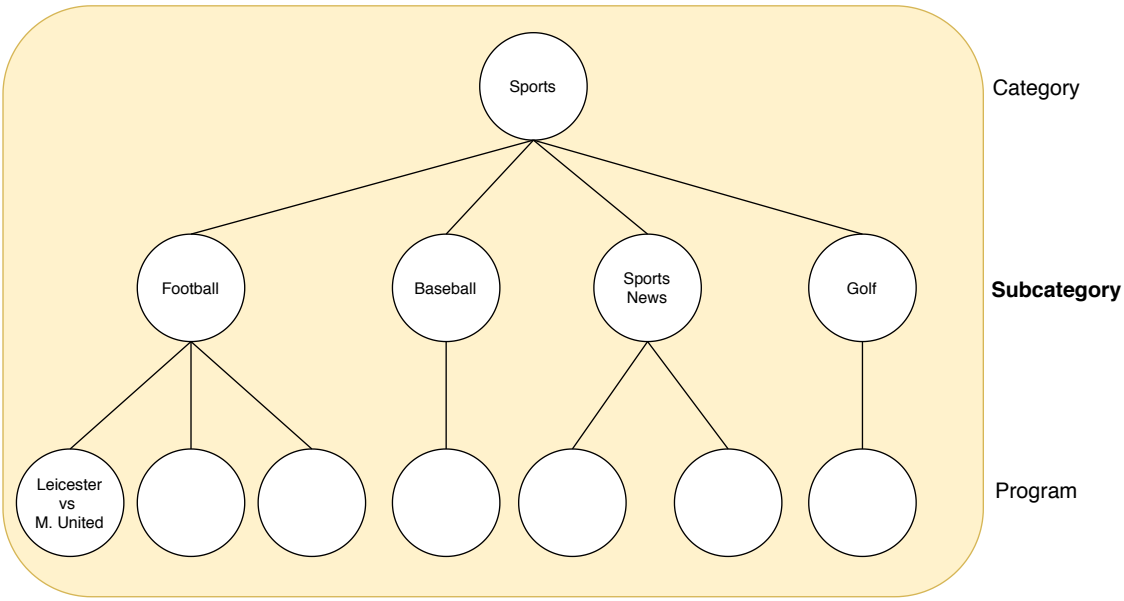


Figure 4.3: An Illustration of the Model 2, with level N = Subcategory

Specifically, it can perform as a regularizer to guide the learning process of item latent features. In the baseline model format, the learning process does not capture the intrinsic relations between the hierarchy levels. Contrarily, in this last model described, the latent features of all nodes must be shared, which suggests helping to learn with less sparsity, since it forms a denser matrix of interactions. Thus, this hierarchical proposal has some advantages considering it does not possess separate data information, but rather all the correlations between levels. With the course of this project, we present an in-depth comparison between distinct models and how hierarchical information can be involved in order to enrich the recommendations. The proposed framework for explicit hierarchical structures includes several modules that provide a straightforward infrastructure to include distinct hierarchies for different domains. This simple interface allows the prediction of the rankings directly with the information of level p of the hierarchy, or with the aggregation of the level below.

To summarize, it is possible to develop specific models, models of the child nodes and then propagate feedback, or we can have a combination of the two through a shared model for everything, which trains with all the data. In this sense, we concede a rich comparison between the different aspects. Aiming to understand the advantages to work with the model that sees with the full picture, which retains the entire hierarchy, comparing with the models that come to the specific level of the hierarchy, and also perceive the aggregations as a way to regularize predictions.

4.4 Hierarchical Model for Cold Start

In Chapter 2, we conducted a study of state of the art in the field of recommender systems. Throughout this revision, collaborative filtering techniques are often considered more accurate and effective than content-based algorithms. Nevertheless, CF approaches are mainly known to suffer from the cold start problem due to its inability to recommend items that are new to the system. Regarding the television field, specifically linear content, matrix factorization models can present some deficiencies when applied to this type of data. In this context, it is important to recall the available program types in live television [Gonçalves et al., 2016a], described in Section 3.2.3.3:

(a) new programs never broadcast before, which means that there is no information on user preferences about them. Denotes the new item cold start problem.

(b) programs broadcast before, but not watched by the user. For these programs we only have preferences from other users;

(c) programs broadcast before and watched by the user, namely new episodes or a repeated program.

Regarding this characterization of the program types, we expect a good performance and efficiency in collaborative-filtering techniques on points b) and c), due to their ability to find similarities between users according to what they watched. In contrast, these approaches will ultimately miscarry on point a) since these similarities are uncovered due to the absence of user preferences. Considering the dynamic nature of live television and the conducted analysis that verified the broad impact of the item cold start (Section 3.2.3.3), all the previously mentioned proposals based on matrix factorization techniques would retain a major shortcoming in this television context recommendation. Therefore, we extended the HMF model to deal with the cold start problem, which provides a novel component using hierarchical information - $Model2c$. The central notion is the possibility to verify the history of categories the user has consumed the most items from and, in this sense, recommend new items to the users based on the proportion of items the user consumed from each of these categories. Hence, the new hierarchical model proposal is based on the addition of information passing from the parent to child nodes in the hierarchy, in the sense of being able to recommend new items. Along the same lines as the preceding model, which shared hierarchical information from lower levels to higher ones, the concept of this variant is to propagate from the parents to respective child nodes of the available hierarchy. This denotes an especially useful condition, for instance, in programs that belong in categories or subcategories that are unpopular, and no one usually sees them. In this way, it becomes more robust to use the parent as a proxy for the child node. In conclusion, the top level of the hierarchy shares the information for the respective items in the level below, performing this spreading flow in a downward direction, as illustrated in Figure 4.4.

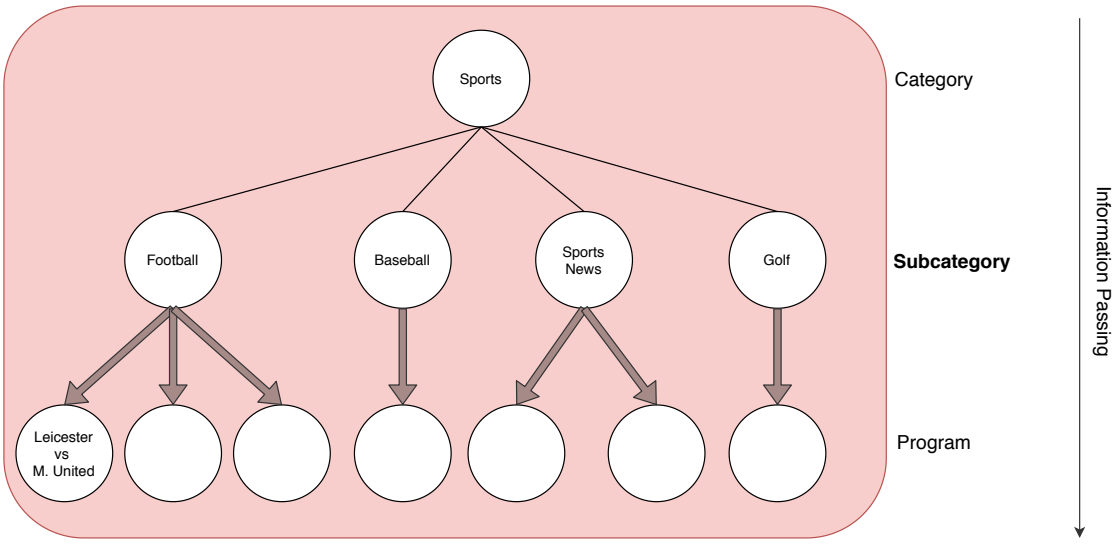


Figure 4.4: An Illustration of the Model 2c, with level N = Subcategory

On the other hand, this characteristic also denotes a valuable property in terms of recommending new items, which the user has never seen on television. Each user has a preference score towards a specific item in the hierarchy. Accordingly, the layer below of that particular item can have several child nodes, also with associated preference levels. In this sense, we assume that a new item that occurs under an individual subcategory corresponds to the average of all the items under that subcategory, multiplied by the user preference of that subcategory itself.

4.5 Summary

This chapter introduces a formal basis for explaining the different notions of hierarchical information. Then, we describe the process of introducing hierarchical information into a baseline algorithm using different perspectives to increase quality measures. Therefore, we have proposed a hierarchical matrix factorization algorithm, HMF, that incorporates the item hierarchies from implicit feedback data. Our proposal intends to address the cold-start problem through this pivotal information, potentially overcoming a significant shortcoming of collaborative filtering approaches to recommendation problems. This contribution is essential regarding its applicability in several environments, particularly in the television domain, due to the dynamic nature of linear content. Our proposal intends to address the cold-start problem through this pivotal information, potentially overcoming a significant shortcoming of collaborative filtering approaches to recommendation problems. Lastly, this contribution denotes an essential characteristic to recommendations regarding its applicability in several environments, particularly in the television domain, due to the dynamic nature of linear content.

Chapter 5

Evaluation

As a central element of the work carried out for this thesis, the design of the experiments of our contributions has been conducted. Thus, we provide a uniform description of the evaluation methodology used as a general basis in the experiments of this chapter, aiming for a straightforward interpretation and comparison of the multiple results obtained. Concerning the rest of the chapter, it is structured as follows. Section 5.1 outlines the type of implicit feedback employed in the datasets on which we evaluate our proposals. In section 5.2, we give details about the ranking-based evaluation methodology to validate the developed recommendation system. Then, the preliminary results obtained in the experimental design segment are reviewed to further comparison in Section 5.3. Finally, in Section 5.4, we provide a discussion on the results obtained and the contributions of the proposals.

5.1 Datasets

In this section, we describe the datasets considered to use in the proposed models using implicit feedback. As outlined in the previous chapter, the core of the data used in this research is based on the traditional cable television service. For our experiments, we have selected the dataset described in section 3.2, in accordance with the final considerations. However, considering the unique characteristics and the different variants encountered, we decided to split the data to form two datasets: LiveTVUnary and LiveTVRelevance. These datasets cover the television recommendation domain, using two distinct types of implicit feedback: positive-only or unary, meaning when we have only positive expressions of a preference assessment for the television program, defined by a threshold; and relevance feedback, in which case the user preference is estimated through the visualization time of the item.

As for the LiveTVUnary dataset, we had to define an appropriate threshold to select the interactions that would characterize only positive feedback. Then, we discarded all entries with elements r_{ui}, representing the interactions between user u and item i, smaller than 0.6. Excluding the advertising content, this value represents watching less than about half of a program, which is not a concrete indication that a user is interested in the program. The target variable is derived though r_{ui} values and designates the preference of the user u relatively to item i. Thus, if a user u consumed item i, meaning that $r_{ui} > 0.6$, it indicates that the user manifests a favorable opinion towards the item, and the target value is set to 1. Conclusively, the remaining information consists of a dataset with nearly 800,000 positive interactions in television sessions. This dataset has the advantage of being relatively small so that all modules of the

system can be executed in a short time while providing consistent and relevant results on account of the quality of the data. Moreover, it facilitates the comparison of results and reproducibility of the performed experiments. Consequently, LiveTVUnary denotes the primary dataset for the course of all the evaluation methodologies.

On the other hand, considering relevance feedback, we obtain an explicit ranking by determining the percentage of the program that was seen by the user. In the LiveTVRelevance dataset, elements r_{ui} represent the percentage of time user u watched the program i. For instance, a r_{ui} value equal to 0.4 means that the user watched 40% of the show duration. In this setting, higher r_{ui} values are proportional to a stronger indication that the user is interested in the item. The LiveTVRelevance dataset is the largest of the two datasets provided and consists of approximately 4 million interactions for 10,447 television programs by about 10,000 users.

The datasets mentioned above represent small data that simplify the reproducibility of our experiments but are able to provide further evidence of the generality of this project's proposals. In table 5.1, we provide a summary of some of the reported magnitudes to consolidate the information on both datasets. $|C_t|$ denotes the number of categories, while $|S_t|$ corresponds to the items' subcategories.

	$\|R\|$	$\|U\|$	$\|J\|$	$\|C_t\|$	$\|S_t\|$
LiveTVUnary	794162	9872	8239	8	114
LiveTVRelevance	3730938	9872	10447	8	117

Table 5.1: Characteristics of Live TV datasets of our experimental design

5.2 Evaluation Methodology

In the present section, we focus on describing the evaluation methodology performed on the recommendation algorithms, used as a general basis in the experiments, aiming for a straightforward interpretation and comparison of the multiple results obtained. We conduct a ranking-based evaluation to compare our proposals with the previous baseline recommenders on the datasets presented. In this sub-case of offline evaluation, it is necessary to divide the datasets into training and test subsets, and then select the items to rank. For the LiveTVUnary and LiveTVRelevance datasets, we have performed a standard division, in which three weeks of data are used for training and the remaining one for test. In other words, our system is trained using television viewing data from the first three weeks of October 2019 to generate predictions of what users will watch on the television during the ensuing week. In these circumstances, the evaluated recommendation algorithms are requested to generate a ranked list of items to present to all the users who have data in both training and test subsets. On the other hand, the items to rank for each user include all items that hold data in the training set. In consideration of the intrinsic characteristics of linear television content, the system only recommends the items that appear in the test set. In other words, if the television program is not broadcast in the evaluation week, is not part of the recommendations. Then, we present a prefix of the list to the user as the recommended television shows. Conclusively, the resulting recommendation lists are assessed using accuracy-based evaluation metrics, detailed in Section 2.5.1.1.

5.3 Results

Concerning the evaluation methodology presented previously, all the obtained results of the baseline algorithm, matrix factorization recommendation, and the proposed hierarchical models are displayed in the current section.

First, considering a ratings prediction task, we decided to use the LiveTVRelevance dataset in a preliminary evaluation phase. Therefore, LiveTVUnary consisted of the primary dataset for the development and assessment of this research. For the rating prediction task, the evaluation metrics described in previous Section 2.5.1.1, the mean absolute error (MAE) and the root mean squared error (RMSE), are adopted to validate the systems' performance. In this sense, smaller values for MAE and RMSE suggest improved rating prediction performance. From the Table 5.2, it is possible to observe the error metrics presented by the baseline model. As might be anticipated, the predictive error is higher at the more granular levels of the hierarchy, where the represented items are the most specific.

	Ratings Prediction		
	Categories	Subcategories	Programs
RMSE	0.18	0.22	0.35
MAE	0.13	0.19	0.27

Table 5.2: Rating prediction results (RMSE and MAE) on LinearTVRelevance

Given the LiveTVUnary dataset, we decided to test the baseline algorithm using a last-one-out approach. Besides the traditional division into train and test, this means including a "temporal" evaluation for assessing the baseline recommender system since data is chronological. The designated last-one-out evaluation approach includes the training set with all the user data except the last rated item, inserted for the testing set. In other words, for each user, we remove the last N rated item individually and then checked if it appeared on the newly generated top-N recommended items. Therefore, if the removed item is recommended within the first N positions, we count as a hit recommendation. In this case, the Table 5.3 show the obtained results with N = $\{1, 5\}$.

	Last-One-Out	
	Subcategories	Programs
Hit Ratio	0.27	0.07

	Last-Five-Out	
	Subcategories	Programs
Recall@5	0.61	0.21
Precision@5	0.36	0.18

Table 5.3: Last-one-out and Last-five-out evaluation on LinearTVUnary

In the next step, we evaluated all the proposed model's effectiveness using the top-N recommendation. Considering this ranking-based evaluation, we select precision@N, recall@N, F1-Score, and MAP@N as the accuracy-based evaluation metrics, where N varies as $\{5, 10, 15\}$.

This comparative assessment is conducted using the different levels of the hierarchy, except when its

application does not make sense, taking into account the characteristics of the data and models proposed. On the other hand, we also separate the models according to the different specialized aggregators. In this case, larger values of Precision@N, Recall@N, F1-Score and MAP@N means a better top-N recommendation performance.

The comparison results for top-N recommendation for all the levels in the hierarchy considered are summarized in Tables 5.4,5.5 and 5.6.

Evaluation Metrics (N=5)	MF			Model 1			Model 2							
							Model 2a			Model 2b			Model 2c	
	Categories	Subcategories	Programs	Aggregator	Categories	Subcategories	Categories	Subcategories	Programs	Aggregator	Categories	Subcategories	Subcategories	Programs
Recall@N	0.892	0.370	0.056	Sum:	0.820	0.348	0.889	0.390	0.036	Sum:	0.846	0.352	0.238	0.014
				Mean:	0.829	0.160				Mean:	0.834	0.156		
				Max:	0.844	0.341				Max:	0.875	0.294		
				Min:	0.744	0.005				Min:	0.763	0.006		
Precision@N	0.633	0.497	0.151	Sum:	0.584	0.470	0.631	0.522	0.097	Sum:	0.599	0.475	0.324	0.062
				Mean:	0.590	0.200				Mean:	0.594	0.193		
				Max:	0.598	0.461				Max:	0.620	0.391		
				Min:	0.538	0.009				Min:	0.546	0.012		
F1-Score	0.740	0.424	0.082	Sum:	0.655	0.366	0.738	0.446	0.053	Sum:	0.673	0.370	0.274	0.023
				Mean:	0.661	0.161				Mean:	0.666	0.156		
				Max:	0.672	0.359				Max:	0.697	0.306		
				Min:	0.600	0.006				Min:	0.611	0.008		
MAP@N	0.822	0.445	0.096	Sum:	0.721	0.360	0.818	0.481	0.054	Sum:	0.745	0.395	0.284	0.024
				Mean:	0.656	0.112				Mean:	0.726	0.105		
				Max:	0.735	0.402				Max:	0.794	0.325		
				Min:	0.511	0.005				Min:	0.521	0.006		

Color Key: 0 — Value — 1

Table 5.4: Top 5 Ranking on LinearTVUnary

Evaluation Metrics (N=10)	MF		Model 1		Model 2					
					Model 2a		Model 2b		Model 2c	
	Subcategories	Programs	Aggregator	Subcategories	Subcategories	Programs	Aggregator	Subcategories	Subcategories	Programs
Recal@N	0.53	0.102	Sum:	0.514	0.559	0.078	Sum:	0.519	0.346	0.043
			Mean:	0.232			Mean:	0.261		
			Max:	0.521			Max:	0.421		
			Min:	0.002			Min:	0.007		
Precision@N	0.37	0.140	Sum:	0.358	0.403	0.103	Sum:	0.349	0.231	0.076
			Mean:	0.159			Mean:	0.173		
			Max:	0.345			Max:	0.282		
			Min:	0.001			Min:	0.011		
F1-Score	0.44	0.118	Sum:	0.422	0.468	0.089	Sum:	0.417	0.277	0.055
			Mean:	0.189			Mean:	0.208		
			Max:	0.415			Max:	0.338		
			Min:	0.001			Min:	0.009		
MAP@N	0.468	0.123	Sum:	0.384	0.498	0.086	Sum:	0.418	0.251	0.052
			Mean:	0.142			Mean:	0.121		
			Max:	0.44			Max:	0.346		
			Min:	0.008			Min:	0.008		

Color Key: 0 — Value — 1

Table 5.5: Top 10 Ranking on LinearTVUnary

Evaluation Metrics (N=15)		MF		Model 1		Model 2					
						Model 2a		Model 2b		Model 2c	
		Subcategories	Programs	Aggregator	Subcategories	Subcategories	Programs	Aggregator	Subcategories	Subcategories	Programs
	Recal@N	0.65	0.138	Sum:	0.612	0.676	0.106	Sum:	0.617	0.502	0.065
				Mean:	0.308			Mean:	0.342		
				Max:	0.652			Max:	0.543		
				Min:	0.008			Min:	0.005		
	Precision@N	0.31	0.127	Sum:	0.283	0.347	0.074	Sum:	0.286	0.183	0.021
				Mean:	0.165			Mean:	0.134		
				Max:	0.269			Max:	0.211		
				Min:	0.010			Min:	0.009		
	F1-Score	0.42	0.132	Sum:	0.387	0.459	0.087	Sum:	0.391	0.268	0.032
				Mean:	0.215			Mean:	0.193		
				Max:	0.381			Max:	0.304		
				Min:	0.009			Min:	0.006		
	MAP@N	0.443	0.142	Sum:	0.345	0.479	0.082	Sum:	0.376	0.234	0.045
				Mean:	0.132			Mean:	0.129		
				Max:	0.421			Max:	0.318		
				Min:	0.004			Min:	0.007		

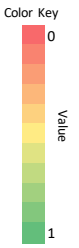


Table 5.6: Top 15 Ranking on LinearTVUnary

Specifically, the methods presented in these tables are defined as:

- MF: matrix factorization based collaborative filtering (baseline);
- Model 1: a model that trains at the N + 1 level of the hierarchy (aggregation);
- Model 2: a model that trains with all items in the hierarchy; It presents three variants:
  - Model 2a: direct queries at level N;
  - Model 2b: aggregates at N + 1 level;
  - Model 2c: aggregates at the N-1 level.

Regarding the personalized algorithms, we observe that the MF baseline gets the best results while the model 2c has the lowest performance according to all metrics and cutoffs. Considering the tested aggregators, the low predictive quality of the aggregator based on the minimum is evident. On the other hand, the sum and maximum aggregators present good results, denoting a favorable choice to perform the hierarchical aggregations. Regarding the subcategories recommendation, both the 2a and 2b models have better results than the baseline.

The obtained results show that the baseline model outperforms the proposed models, conferring better results in the categories and programs. Regarding the subcategories level, model 2a, that perceives at the direct level N, provides better results than the basic matrix factorization method.

5.4 Discussions

The comparison conducted between several algorithms suggests that the baseline matrix factorization performs better than all the algorithms based on the items hierarchies. Furthermore, it is possible to verify the accuracy-diversity trade-off using hierarchical organizations. As the hierarchy goes down, the number of items increases and becomes more specific. This results in increased diversity as the model is required to recommend more and specific categories. Therefore, this results in decreased accuracy as the user history is insufficient to estimate the preference for these lower level items.

Our main contribution – the algorithms – is therefore valuable for showing that the item categories can be used to make diverse and personalized recommendations given any base recommender, with little compromise in accuracy.

Hierarchical Matrix Factorization ensures a scalable and efficient structure for applying hierarchical item categories from implicit feedback. Moreover, since different users are interested in different categories, recommending using item hierarchies results in personalized recommendations. Therefore, the overall system diversity also increases.

In this scope, the last proposal, $Model2c$, reveals a decreased accuracy result, but offers more quality and usefulness for the recommender system since hierarchical item categories can be used to make diverse and personalized recommendations given any base recommender. In conclusion, the remaining results do not differ significantly and are identical through all the models. Moreover, even if the proposed models' predictive accuracy achieves lower results than the baseline, the idea is that we still have a more comprehensive model that allows all the advantages already mentioned throughout this dissertation, including the various forms of interaction, and numerous use cases.

Chapter 6

Conclusions

6.1 The Hierarchical Approach

Recommender systems have become a widespread application of machine learning technologies in an extensive spectrum of daily requests and a well-known engine to the general public. Specifically, the live television broadcast services present users with an overwhelming scope of possibilities from which they can choose. Regarding this domain, numerous recommender systems have been proposed, but most recommendation approaches do not consider the intrinsic hierarchical structure of audiovisual content. In recent years, the hierarchical organization has been establishing an effective way to guide users to their items of interest.

In this dissertation, we investigate the problem of recommending content at multiple granularity levels using explicit hierarchical structures of items in television services. The primary contribution of this proposal consisted of a comprehensive characterization of live television usage on a large Portuguese telecommunications company. Most research on recommendation systems for television is designed for non-linear content, denoting the VOD setting. Nevertheless, this type of content only represents a small portion of all views of the users of a Cable TV service. Therefore, we conducted a consistent characterization of linear content, that revealed a significant piece of this research since it unveiled particular characteristics that must be considered when building a recommendation system for a Cable TV provider.

On the other hand, we also provided the development of a reliable television dataset, a fundamental key for relevant and personalized recommendations. The contextual information, which we describe through insights from the previously mentioned characterization, enables to obtain better and more personalized recommendations.

Concerning the following contributions, we proposed the development of a hierarchical recommendation engine for television providers through matrix factorization techniques from collaborative filtering (CF). The core of this project consists of a unified framework to assess the importance of including hierarchical knowledge and enhance several perspectives through its employment in television recommender systems. This proposed hierarchical matrix factorization, HMF, manages the information available in these environments to predict the users' engagement with TV content through implicit feedback obtained from television viewing sessions.

Furthermore, we have addressed the large impact of the cold start problem on linear television by

developing an additional HMF approach that considers distinct forms of information passing through items' hierarchies. In this sense, our last proposal is able to recommend new items for which there is no user feedback, as opposed to collaborative filtering algorithms. Addressing these requirements, the last component of our framework, presented a new hierarchical model to optimize the usefulness of recommendations, particularizing the importance of including novelty measures.

The conducted research has revealed the universe of possibilities and the actual worth of hierarchical information for building personalized recommendations that are able to improve recommender systems' performance. Indeed, incorporating this type of structure denotes a relevant factor in providing relevant recommendations and maximizing recommendation systems' applicability in several use cases.

A set of conducted experiments on the developed datasets illustrate the importance of the explicit hierarchical structures of items in the recommendation performance improvement. The results of our experiments verify the efficacy of the contributions and allow further insights when applied to different environments. We demonstrate that this item hierarchy can be useful in making the recommendations more explainable and increasing the recommendations' novelty without compromising much on the accuracy. In conclusion, the final proposals achieve a better trade-off between accuracy and novelty of results than baseline approaches.

6.2 Future Work

The contributions of this dissertation introduce distinct ways of including and evaluating hierarchical information in Recommender Systems. However, the work presented here opens the way for several additional extensions.

Indeed, there are several interesting directions needing further investigation in the television domain. Accordingly, the theme of recommending programs never seen before is a much more challenging task that requires further investigation.

Besides, in this work, we choose the matrix factorization as our primary model to capture the implicit hierarchical structures of items, and we would like to investigate other basic models.

Moreover, user studies would also bring additional validation and further insights into the addressed questions and the proposed approaches, so far examined by theoretical analysis and offline experimentation.

Another line of future work consists in the extension of our work by means of including recommendations based on the content or features of the items in the recommendation domain.

Finally, the explicability of recommendations denotes a recurrent concept in many current recommendation systems, although television services do not include these settings. Hence, a different envisioned extension of our contributions that would include an explanation component with a focus on hierarchical descriptions is a possible path to conduct further work.

Bibliography

Emile Aarts and Jan Korst. Simulated annealing and boltzmann machines. 1988.

Jorge Abreu, Pedro Almeida, and Bruno Teles. Tv discovery & enjoy: a new approach to help users finding the right tv program to watch. In *Proceedings of the ACM International Conference on Interactive Experiences for TV and Online Video*, pages 63–70, 2014.

Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749, 2005.

Charu C Aggarwal et al. *Recommender systems*, volume 1. Springer, 2016.

Xavier Amatriain. Mining large streams of user data for personalized recommendations. *ACM SIGKDD Explorations Newsletter*, 14(2):37–48, 2013.

Ricardo Baeza-Yates, Berthier Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.

Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

Riccardo Bambini, Paolo Cremonesi, and Roberto Turrin. A recommender system for an iptv service provider: a real large-scale production environment. In *Recommender systems handbook*, pages 299–331. Springer, 2011.

Robert M Bell and Yehuda Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 43–52. IEEE, 2007.

James Bennett, Stan Lanning, et al. The netflix prize. In *Proceedings of KDD cup and workshop*, volume 2007, page 35. New York, 2007.

David C Blair and Melvin E Maron. An evaluation of retrieval effectiveness for a full-text document-retrieval system. *Communications of the ACM*, 28(3):289–299, 1985.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. *arXiv preprint arXiv:1301.7363*, 2013.

Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.

Robin Burke. Hybrid web recommender systems. In *The adaptive web*, pages 377–408. Springer, 2007.

Iván Cantador, Alejandro Bellogín, and Pablo Castells. Ontology-based personalised and context-aware recommendations of news items. In *2008 IEEE/WIC/ACM international conference on web intelligence and intelligent agent technology*, volume 1, pages 562–565. IEEE, 2008.

Iván Cantador, Alejandro Bellogín, and David Vallet. Content-based recommendation in social tagging systems. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 237–240, 2010.

Òscar Celma, Miquel Ramírez, and Perfecto Herrera. Foafing the music: A music recommendation system based on rss feeds and user preferences. In *ISMIR*, pages 464–467, 2005.

Tzung-Shi Chen, Cheng-Sian Chang, Jeng-Shian Lin, and Hui-Ling Yu. Context-aware writing in ubiquitous learning environments. *Research and Practice in Technology Enhanced Learning*, 4(01):61–82, 2009.

Mark Claypool, Anuja Gokhale, Tim Miranda, Paul Murnikov, Dmitry Netes, and Matthew Sartin. Combing content-based and collaborative filters in an online newspaper. 1999.

Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 39–46, 2010.

Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6): 391–407, 1990.

Simon Funk. MS Windows NT kernel description, 2006.

Thomas George and Srujana Merugu. A scalable collaborative filtering framework based on co-clustering. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, pages 4–pp. IEEE, 2005.

Jennifer Golbeck and Matthew Rothstein. Linking social networks on the web with foaf: A semantic web case study. In *AAAI*, volume 8, pages 1138–1143, 2008.

Carlos A. Gomez-Uribe and Neil Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Trans. Manage. Inf. Syst.*, 6(4), December 2016. ISSN: 2158-656X. doi:10.1145/2843948.

Diogo Gonçalves, Miguel Costa, and Francisco Couto. A large-scale characterization of user behaviour in cable tv. 09 2016a.

Diogo Gonçalves, Miguel Costa, and Francisco M Couto. A flexible recommendation system for cable tv. *arXiv preprint arXiv:1609.02451*, 08 2016b.

F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), December 2015. ISSN: 2160-6455. doi:10.1145/2827872.

David E Heckerman, Martin Luo, Guy Shani, and Mahbubul Alam Ali. Modifying advertisement scores based on advertisement response probabilities, May 6 2008. US Patent 7,370,002.

Jonathan L Herlocker, Joseph A Konstan, Loren G Terveen, and John T Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 194–201, 1995.

Geoffrey E Hinton, Terrence J Sejnowski, and David H Ackley. *Boltzmann machines: Constraint satisfaction networks that learn.* Carnegie-Mellon University, Department of Computer Science Pittsburgh, 1984.

Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, 22(1):89–115, 2004.

Shang H Hsu, Ming-Hui Wen, Hsin-Chieh Lin, Chun-Chia Lee, and Chia-Hoang Lee. Aimed-a personalized tv recommendation system. In *European conference on interactive television*, pages 166–174. Springer, 2007.

Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, pages 263–272. Ieee, 2008.

Zan Huang, Hsinchun Chen, and Daniel Zeng. Applying associative retrieval techniques to alleviate the sparsity problem in collaborative filtering. *ACM Transactions on Information Systems (TOIS)*, 22(1): 116–142, 2004.

Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 97–106, 2001.

Marius Kaminskas and Francesco Ricci. Location-adapted music recommendation using tags. In *International conference on user modeling, adaptation, and personalization*, pages 183–194. Springer, 2011.

Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434, 2008.

Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

Gai Li, Zhiqiang Zhang, Liyang Wang, Qiang Chen, and Jincai Pan. One-class collaborative filtering based on rating prediction and ranking prediction. *Knowledge-Based Systems*, 124, 03 2017. doi:10.1016/j.knosys.2017.02.034.

Hui Li, Yu Liu, Yuqiu Qian, Nikos Mamoulis, Wenting Tu, and David Cheung. Hhmf: hidden hierarchical matrix factorization for recommender systems. *Data Mining and Knowledge Discovery*, 33, 05 2019. doi:10.1007/s10618-019-00632-4.

Xue Li, Jorge M Barajas, and Yi Ding. Collaborative filtering on streaming data with interest-drifting. *Intelligent Data Analysis*, 11(1):75–87, 2007.

Blerina Lika, Kostas Kolomvatsos, and Stathes Hadjiefthymiades. Facing the cold start problem in recommender systems. *Expert Systems with Applications*, 41(4):2065–2073, 2014.

Greg Linden, Brent Smith, and Jeremy York. Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing*, 7(1):76–80, 2003.

Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. Personalized news recommendation based on click behavior. In *Proceedings of the 15th international conference on Intelligent user interfaces*, pages 31–40, 2010.

Tie-Yan Liu et al. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331, 2009.

Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer, 2011.

Kai Lu, Guanyuan Zhang, Rui Li, Shuai Zhang, and Bin Wang. Exploiting and exploring hierarchical structure in music recommendation. In *Asia Information Retrieval Symposium*, pages 211–225. Springer, 2012.

Harry Mak, Irena Koprinska, and Josiah Poon. Intimate: A web-based movie recommender using text categorization. In *Proceedings IEEE/WIC International Conference on Web Intelligence (WI 2003)*, pages 602–605. IEEE, 2003.

Marcin Maleszka, Bernadetta Mianowska, and Ngoc Thanh Nguyen. A method for collaborative recommendation using knowledge integration tools and hierarchical structure of user profiles. *Knowledge-Based Systems*, 47:1–13, 2013.

Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge university press, 2008.

Prem Melville, Raymond J Mooney, and Ramadass Nagarajan. Content-boosted collaborative filtering for improved recommendations. *Aaai/iaai*, 23:187–192, 2002.

Stuart E Middleton, David De Roure, and Nigel R Shadbolt. Ontology-based recommender systems. In *Handbook on ontologies*, pages 477–498. Springer, 2004.

Bamshad Mobasher, Honghua Dai, Tao Luo, and Miki Nakagawa. Effective personalization based on association rule discovery from web usage data. In *Proceedings of the 3rd international workshop on Web information and data management*, pages 9–15, 2001.

Raymond J Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 195–204, 2000.

Jose Maria Moreno-Jimenez and Luis G Vargas. A probabilistic study of preference structures in the analytic hierarchy process with interval judgments. *Mathematical and Computer Modelling*, 17(4-5): 73–81, 1993.

Athanasios N Nikolakopoulos, Marianna Kouneli, and John Garofalakis. A novel hierarchical approach to ranking-based collaborative filtering. In *International Conference on Engineering Applications of Neural Networks*, pages 50–59. Springer, 2013.

Athanasios N Nikolakopoulos, Marianna A Kouneli, and John D Garofalakis. Hierarchical itemspace rank: Exploiting hierarchy to alleviate sparsity in ranking-based recommendation. *Neurocomputing*, 163:126–136, 2015.

Vito Claudio Ostuni, Tommaso Di Noia, Eugenio Di Sciascio, and Roberto Mirizzi. Top-n recommendations from implicit feedback leveraging linked open data. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 85–92, 2013.

Rong Pan, Yunhong Zhou, Bin Cao, Nathan N Liu, Rajan Lukose, Martin Scholz, and Qiang Yang. One-class collaborative filtering. In *2008 Eighth IEEE International Conference on Data Mining*, pages 502–511. IEEE, 2008.

Ulrich Paquet and Noam Koenigstein. One-class collaborative filtering with random graphs. In *Proceedings of the 22nd international conference on World Wide Web*, pages 999–1008, 2013.

Alexandre Passant. dbrec—music recommendations using dbpedia. In *International Semantic Web Conference*, pages 209–224. Springer, 2010.

Arkadiusz Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD cup and workshop*, volume 2007, pages 5–8, 2007.

Michael J Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial intelligence review*, 13(5-6):393–408, 1999.

Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.

Alexandrin Popescul, Lyle Ungar, David Pennock, and Steve Lawrence. Probabilistic models for unified collaborative and content-based recommendation in sparse-data environments. 08 2001.

Anand Rajaraman and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, 2011.

Steffen Rendle and Christoph Freudenthaler. Improving pairwise learning for item recommendation from implicit feedback. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 273–282, 2014.

Paul Resnick and Hal R Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, 1994.

Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2011.

Yong Rui, Thomas S Huang, Michael Ortega, and Sharad Mehrotra. Relevance feedback: a power tool for interactive content-based image retrieval. *IEEE Transactions on circuits and systems for video technology*, 8(5):644–655, 1998.

Stuart Russell, John Binder, Daphne Koller, and Keiji Kanazawa. Local learning in probabilistic networks with hidden variables. In *IJCAI*, volume 95, pages 1146–1152, 1995.

Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*, pages 791–798, 2007.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Application of dimensionality reduction in recommender system-a case study. Technical report, Minnesota Univ Minneapolis Dept of Computer Science, 2000.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295, 2001.

Andrew Schein, Alexandrin Popescul, Lyle Ungar, and David Pennock. Methods and metrics for cold-start recommendations. pages 253–260, 08 2002. doi:10.1145/564376.564421.

Guy Shani and Asela Gunawardana. Evaluating recommendation systems. In *Recommender systems handbook*, pages 257–297. Springer, 2011.

Upendra Shardanand and Pattie Maes. Social information filtering: algorithms for automating "word of mouth". In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217, 1995.

Yue Shi, Martha Larson, and Alan Hanjalic. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Computing Surveys (CSUR)*, 47(1):1–45, 2014.

Peretz Shoval, Veronica Maidel, and Bracha Shapira. An ontology-content-based filtering method. 2008.

Ian Soboroff and Charles Nicholas. Combining content and collaboration in text filtering. In *Proceedings of the IJCAI*, volume 99, pages 86–91. sn, 1999.

Harald Steck. Evaluation of recommendations: rating-prediction and ranking. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 213–220, 2013.

Zhu Sun, Jie Yang, Jie Zhang, and Alessandro Bozzon. Exploiting both vertical and horizontal dimensions of feature hierarchy for effective recommendation. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Alvin Toffler. Future shock, 1970. *Sydney. Pan*, 1970.

Tran The Truyen, Dinh Q Phung, and Svetha Venkatesh. Ordinal boltzmann machines for collaborative filtering. *arXiv preprint arXiv:1205.2611*, 2012.

Lyle H Ungar and Dean P Foster. Clustering methods for collaborative filtering. In *AAAI workshop on recommendation systems*, volume 1, pages 114–129. Menlo Park, CA, 1998.

Koen Verstrepen and Bart Goethals. Unifying nearest neighbors collaborative filtering. In *Proceedings of the 8th ACM Conference on Recommender systems*, pages 177–184, 2014.

João Vinagre, Alípio Jorge, and João Gama. An overview on the exploitation of time in collaborative filtering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 5, 07 2015. doi:10.1002/widm.1160.

Maksims Volkovs and Guang Wei Yu. Effective latent models for binary feedback in recommender systems. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 313–322, 2015.

Suhang Wang, Jiliang Tang, Yilin Wang, and Huan Liu. Exploring hierarchical structures for recommender systems. *IEEE Transactions on Knowledge and Data Engineering*, 30(6):1022–1035, 2018.

Markus Weimer, Alexandros Karatzoglou, and Alex Smola. Improving maximum margin matrix factorization. *Machine Learning*, 72(3):263–276, 2008.

Cort J Willmott and Kenji Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate research*, 30(1):79–82, 2005.

Yi Wu, Yueting Zhuang, and Yunhe Pan. Content-based video similarity model. In *Proceedings of the eighth ACM international conference on Multimedia*, pages 465–467, 2000.

Jie Yang, Zhu Sun, Alessandro Bozzon, and Jie Zhang. Learning hierarchical feature influence for recommendation by recursive regularization. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 51–58, 2016.

Kai Yu, Anton Schwaighofer, Volker Tresp, Xiaowei Xu, and H-P Kriegel. Probabilistic memory-based collaborative filtering. *IEEE Transactions on Knowledge and Data Engineering*, 16(1):56–69, 2004.

Sheng Zhang, Weihong Wang, James Ford, and Fillia Makedon. Learning from incomplete ratings using non-negative matrix factorization. In *Proceedings of the 2006 SIAM international conference on data mining*, pages 549–553. SIAM, 2006.

Yi Zhang and Jonathan Koren. Efficient bayesian hierarchical user modeling for recommendation system. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 47–54, 2007.