FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Hybrid Approach to Content Recommendation

Fábio André da Cunha Almeida

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Maria Teresa Magalhães da Silva Pinto de Andrade (PhD)

July 16, 2012

Hybrid Approach to Content Recommendation

Fábio André da Cunha Almeida

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Doctor Maria Cristina de Carvalho Alves Ribeiro
External Examiner: Doctor José Manuel de Castro Torres
Supervisor: Doctor Maria Teresa Magalhães da Silva Pinto de Andrade

July 16, 2012

Abstract

Consumption of multimedia content, from TV programming, to music downloads through video streaming, has become common in our daily lives. However, the sheer size and diversity of currently available online resources, is turning choice and selection into a very difficult task. Systems that can assist the user in selecting useful information are thus becoming increasingly important. Such systems make use of recommendation engines – this is the main area of this dissertation.

This work has been conducted within the context of a research project being led by INESC Porto, to develop a recommender engine for the hospitality market. This engine is to be incorporated in the IPTV entertainment services offered to hotel guests, through which, in addition to watch TV programmes or rent movies, the guests can also access other types of services (e.g. book a tennis court).

The initial version of this recommendation engine follows a content-based approach and provides recommendations only for the television service. Three main limitations can be pointed to this initial implementation: 1) it is known that content-based approaches can rapidly become very focussed, providing the user always the same type of recommendations; 2) it only addresses TV content, neglecting other types of resources and services the guest has access to; 3) it is not capable of establishing relationships across both the users and the resources, to increase the performance and enrich the recommendations.

The main motivation of this dissertation was to investigate solutions to overcome these limitations. In particular, it aimed at exploring the use of hybrid approaches, by establishing a measure of proximity between guests and thus include in the recommendation list of one guest, items that would have been preferred by other guests with similar profile. By finding similarities between users, it is possible to make completely unexpected recommendations to them – this is the main advantage of collaborative approaches. Given that both content-based (CB) and collaborative filtering (CF) approaches have individually several limitations, an hybrid approach has the potential of delivering enhanced results by exploring the best of both.

This goal was pursued by initially performing a study, comparing the performances of different hybrid approaches when applied to the same type of resources as the ones present in our system. This procedure was important, as it has already been proved that the performance of different approaches can significantly vary depending on the domain they are applied to.

Supported by this initial study, a novel hybrid approach based on the kNN algorithm was proposed: an improved Pearson Correlation method for the user similarity weight computation was used to select the user's neighbourhood, which was then used to generate a collaborative "Predicted Profile". This profile is then used to generate recommendations applying the CB method that had been previously implemented (Naive Bayes). This novel method proposed in this dissertation adopts a different approach to address the sparsity problem, since there isn't a user-item rating matrix used in recommendations generation. This can be considered very important, especially when dealing with a large quantity and diversity of resources.

Resumo

O consumo de conteúdos multimédia desde programação televisiva, a *downloads* de música, até *streaming* de vídeo, tem vindo a tornar-se cada vez mais comum no nosso quotidiano. Contudo, com o aumento diário do número e diversidade de recursos *online* disponíveis, torna a escolha uma tarefa árdua. Sistemas capazes de auxiliar o utilizador a seleccionar informação útil têm vindo a tornar-se cada vez mais importantes. Estes sistemas recorrem a motores de recomendação - a área principal desta dissertação.

O trabalho de investigação conduzido neste projeto foi realizado no INESC Porto, com o principal objectivo de desenvolver um sistema de recomendação para o mercado hoteleiro. Este sistema foi incorporado no serviço de entretenimento IPTV oferecido aos hóspedes do hotel, através do qual para além de poderem ver programas televisivos ou alugar filmes, os hóspedes podem também aceder a outros tipos de serviços (p. ex. reservar um campo de ténis).

A versão inicial deste motor de recomendação segue uma abordagem baseada em conteúdo e fornece recomendações apenas para conteúdo televisivo. Três principais limitações podem ser apontadas a esta implementação inicial: 1) é sabido que abordagens baseadas em conteúdo podem rapidamente tornar-se muito especializadas, fornecendo sempre o mesmo tipo de recomendações ao utilizador; 2) apenas funciona com conteúdo televisivo, descartando outros tipos de recursos disponíveis; 3) despreza a possibilidade de estabelecer relações entre utilizadores e entre outros tipos de conteúdo, de forma a aumentar a performance e enriquecer o tipo de recomendações.

A principal motivação desta dissertação é investigar soluções para ultrapassar estas limitações. Visa explorar o uso de abordagens híbridas, estabelecendo medidas de proximidade entre hóspedes e, portanto, incluir na lista de recomendações de um hóspede itens que foram consumidos por outros hóspedes com perfis semelhantes. Através da procura de semelhança entre utilizadores, será possível fazer recomendações completamente inesperadas – esta é a principal vantagem das abordagens colaborativas. Visto que tanto as abordagens baseadas em conteúdo como as abordagens colaborativas têm diversas limitações quando são aplicadas separadamente, uma abordagem híbrida tem o potencial para produzir melhores resultados, explorando o melhor de cada técnica.

Para alcançar este objectivo, inicialmente foi feito um estudo comparativo das performances de diferentes abordagens híbridas quando aplicadas a conteúdos como estes. Este procedimento foi de grande importância, visto que já foi provado que a performance de diferentes abordagens pode variar significativamente dependendo da situação em que são aplicadas.

Suportada por este estudo inicial, uma nova abordagem híbrida baseada no algoritmo *kNN* foi proposta: um método aperfeiçoado baseado numa técnica da correlação de Pearson foi utilizado para seleccionar a vizinhança, que é posteriormente utilizada na geração de um perfil colaborativo. Este perfil é depois utilizado para gerar recomendações aplicando a técnica de filtragem colaborativa implementada anteriormente (Naive Bayes). Esta novo método proposto adota uma diferente abordagem para resolver o problema da esparsidade, visto que não é usada uma matriz de ratings utilizador-item na geração de recomendações. Isto é um factor muito importante, especialmente quando estamos a lidar com grandes quantidades de recursos de diferentes tipos.

Acknowledgements

I would like to thank to everyone who contributed to this dissertation in some manner. This would not have been possible without the kind support and help of many people around me, to only some of whom it is possible to give particular mention here.

First of all, to my supervisor Maria Teresa Andrade, that provided me valuable guidance and support during all phases of this dissertation.

A special thanks to my co-supervisor Pedro Morgado, for helping me since the beginning and for always follow my work and being supportive despite being far.

To everybody at Nonius Software that helped me to understand their system and made my work a little easier.

To my parents and sisters, thanks for your support during this period and apologize for my absence.

To my girlfriend, for her personal support, great patience and motivation at all times, which made all the difference.

To FEUP and especially to all the teachers that followed my path during the last 5 years, thank you for the transmitted knowledge and for always being so demanding.

Last but not least, I need to thank all my friends that were always ready to give me support and motivation during the most difficult phases of my work. Thank you also for your company and motivation during the sleepless nights, it was priceless.

Thank you everybody.

Fábio Almeida

*"If you can't explain it simply,
you don't understand it well enough"*

Albert Einstein

Contents

List of Figures

LIST OF FIGURES

List of Tables

Abbreviations

ANN	Artificial Neural Network
API	Application Programming Interface
CB	Content-Based Filtering
CF	Collaborative Filtering
CV	Cosine Vector
DB	Database
DT	Decision Tree
EPG	Electronic Program Guide
FEUP	Faculdade de Engenharia da Universidade do Porto
FR	Functional Requirement
HS	Hybrid System
IF	Information Filtering
IMDb	The Internet Movie Database
IPTV	Internet Protocol Television
IR	Information Retrieval
JSON	JavaScript Object Notation
kNN	k-Nearest-Neighbour
MAE	Mean Absolute Error
ML	Machine Learning
NB	Naive Bayes
NFR	Non-functional Requirement
PC	Pearson Correlation
RBC	Rule-based Classifier
RMSE	Root Mean Square Error
RS	Recommender System
SOA	Service-Oriented Architecture
STB	Set-top-box
TV	Television
UC	Use Case
UP	User Profiling
VoD	Video on Demand
XML	Extensible Markup Language

Chapter 1

Introduction

This chapter contextualizes the dissertation presented in this document, providing relevant background information. It presents the problem this dissertation aims to address, the motivation for solving it, the main goals to attain towards the envisaged solution and the methodologies to adopt in that process. It finalizes by describing the structure of the document, including a short resume of each chapter.

1.1 Context

Due to the increasing offer of different types of multimedia content, consumers are becoming more demanding. At the same time, with the sheer size and diversity of online resources growing daily, it's impossible for the users to analyse exhaustively all the available items to decide what is really relevant for them. Therefore, systems that are able to assist the user in selecting useful information, are becoming increasingly important towards meeting the users' expectations. To address this problem, it is necessary to make use of a recommender system – the main area of this dissertation. Recommender systems (also called recommendation systems/platforms/engines) help the users to choose items they can find useful or of their interest. According to some early definitions, the main task of a recommender system is to "select some objects according to the user's requirement" [Wan98].

These systems have become a very important research area since the appearance of the first papers on collaborative filtering in the mid-1990s [SM93, RIS+94, HSRF95, SM95] when researchers started focusing on recommendation problems that explicitly rely on the ratings structure. The first approaches aimed mainly at providing simple recommendations on news or music content to the users, based on similar items they consumed before or other similar users did. Since then, much work has been done both in the industry and academia on developing new approaches to the recommender systems.

This dissertation was developed within the framework of the project QREN Hotel 3.0. The research work conducted in this project was led by INESC Porto [1], having as main goal the development of a recommender engine for the hospitality market. This system is to be incorporated in the IPTV entertainment service offered to hotel guests, through which, in addition to watch TV programmes, rent movies, listen to radio or access Internet contents such as YouTube, the guests can also access to and buy other types of services (e.g., book a tennis court, rent a bicycle, etc.). The initial version of this recommendation engine follows a content-based approach and provides recommendations only for the television service.

1.2 Motivation and Goals

Recommendation systems greatly depend on the type of application they are devised for. The needs and characteristics of users will differ from one application area to the other and thus different types of information, relationships and rules will apply. Accordingly, different systems will follow distinct approaches and modes of operation to suit this diversity whilst providing optimal results.

As stated previously, the recommendation engine developed within the Hotel 3.0 project, has adopted a content-based approach and focussed on recommending television content. Nonetheless, three main limitations can be pointed to this initial implementation: 1) it is known from the literature that a content-based approach can rapidly become very focussed, providing the user always the same type of recommendations ("over-specialization" problem) (see section 2.3.8); 2) it only addresses TV content, neglecting other types of resources and services the guest has access to; 3) it neglects the possibility of establishing relationships both across the users and other resources, to increase the performance and enrich the type of recommendations.

In this context, the main motivation of this dissertation is to investigate solutions to overcome these limitations. In particular, it aims at exploring the use of hybrid approaches, by establishing a measure of proximity between guests and thus include in the recommendation list of one guest, items that have been consumed by other guests with similar profile. In the initial phase, the implemented system only made recommendations based on the characteristics of the items consumed by the users – content-based filtering. The content-based approaches only try to find items with similar characteristics to others that the user has consumed before. By establishing similarities between users, it is possible to make completely unexpected, yet eventually relevant, recommendations to users – this is the main advantage of collaborative approaches. Users that have similar preferences are designated as *neighbours*. Normally, neighbours share common preferences for particular types of items, but exhibit also differences among them. As such, incorporating collaboration between users in the recommendation process, transforming the content-based approach into a hybrid solution, has the potential of improving the performance of the system. This is the first major goal set for this dissertation.

[1] Institute for Systems and Computer Engineering of Porto - is a private non-profit association, recognised as a Public Interest Institution and an Associate Laboratory since 2002

One of the previously mentioned limitations of the system, is that it only addresses TV content. However, besides this type of content, users can also access and consume other types of resources, notably services offered by the hotel (e.g., book a tennis court or a spa session; rent a bicycle; etc.) and also watch a movie from the hotel video-on-demand (VoD) catalogue. Therefore, the recommendation engine would offer added-value, both to the users as well as to the hotel, if it would be able to generate recommendations combining all these different types of resources. So, the second major goal of this dissertation is to enhance the recommendation process by combining different kinds of resources, while ensuring the obtention of good results.

To achieve this goal, a study about different hybrid approaches was made and their results and performances compared, when applied to contents as the ones present in our system. This procedure led to the obtention of important scientific results, as it has already been proved that the performance of different approaches can significantly vary depending on the situation they are applied to.

To accomplish these goals the following methodology was followed. An initial phase was conducted to identifying the major scientific and technical challenges of the work. This step allowed to identify the necessity of: 1) specifying a set of tags and a structure to characterize and organize content as well as to represent user profiles; 2) defining the method for establishing similarities between users; and 3) investigating available techniques for combining content-based filtering with collaborative filtering, and selecting and implementing the one that could better meet the application's requirements. The major scientific approaches for these challenges are described in the next chapter of this manuscript.

1.3 Document Structure

This manuscript is structured in six chapters. Chapter 2 presents the state-of-the-art on topics that are relevant to this dissertation. It includes a general introduction about the relevance of recommender systems and possible forms of classifying them. It describes in more detail the major characteristics of each class, highlighting strengths and weaknesses. It also presents the most commonly used metrics for evaluating the performance of recommender systems and provides examples of existing frameworks for implementing collaborative filtering algorithms.

Chapter 3 shows the system specification and the followed scientific approach, providing an overview of the previously existing system, describing the aspects that had greater impact on the decisions taken and approaches adopted in the course of this dissertation to achieve the proposed goals. This allows to understand the limitations that can be pointed to the previous work, as well as the approach used to overtake them.

Chapter 4 describes the most important implementation decisions. It shows the architecture of the system to give an overview about all the interaction and overall behaviour, presenting the class models and database models. It also explains how the information processing is made, to understand the decisions on the approach. In this section is also shown how the features domain works and the changes made to it. The used data sources (Web services, etc.) are also presented in this chapter. User profiling changes are presented in this chapter as well. One of the most important sections presented here it's the recommendation method, that shows all the workflow for the proposed hybrid recommender engine.

Chapter 5 presents the solution validation. Here it's explained the integration of this approach into the existing system, the experimental setup and the result analysis to understand the benefits and drawbacks of this solution. Finally, it's presented a short summary and discussion about the results.

Chapter 6 presents some conclusions about the presented work, as well as some future work.

Chapter 2

State of the Art

In this chapter the state of the art about the themes that are relevant for this dissertation are described.

In section 2.1, it's given a general introduction about the relevance of recommender systems , and the general problem is formalized.

In section 2.2, recommender systems are divided into categories, describing each one and concluding with their advantages and drawbacks. Here a special emphasis is given to different approaches to create a hybrid system, as this is the main goal of this dissertation.

Section 2.3 describes the most know issues from recommender systems.

Section 2.4 presents an overview about some of the most important approaches that are widely used in the literature, especially those that were already been applied in systems similar to the proposed one.

Section 2.5 presents the most used metrics for evaluating the accuracy of recommender systems.

Finally, in section 2.6 some examples of frameworks that implement collaborative filtering algorithms are given.

2.1 Recommender Systems Overview

Recommender systems (also called recommendation systems/platforms/engines) help the users to choose objects they can find useful or of their interest. According to some early definitions, the main task of a recommender system is to "select some objects according to the user's requirement" [Wan98].

Recommender systems are developed with the aim to predict the 'rating' or 'preference' that a user would give to an item or social element they had not yet considered. The most two common approaches to solve this problem are using a model built from the characteristics of an item (content-based approaches) or the user's social environment (collaborative filtering approaches).

These predictions can be applied to very different objects, for example books, music, television programmes, services or even other people.

In general, recommender systems may serve two different purposes. On one hand, they can be used to stimulate users into doing something such as buying a specific book or watching a specific movie. On the other hand, recommender systems can also be seen as tools for dealing with *information overload*, as these systems aim to select the most interesting items from a larger set. Thus, recommender systems research is also strongly rooted in the fields of *information retrieval* and *information filtering*. In these areas, however, the focus lies mainly on the problem of discriminating between relevant and irrelevant items [JZFF10].

However, the provision of personalized recommendations requires that the system knows something about every user. Every recommender system must develop and maintain a *user model* or a *user profile* that, for example, contains the user's preferences. Although the existence of a user model is central to every recommender system, the way in which this information is acquired and exploited depends on the particular recommendation technique. User preferences can, for instance, be acquired *implicitly* by monitoring user behaviour, but the recommender system might also *explicitly* ask the user about his or her preferences. These preferences will be used by the recommender system to predict the user's evaluation of the object or the evaluations the user has done in the past. The more evaluations the users make, the better the results will be [CML+11].

These systems have become a very important research area since the appearance of the first papers on collaborative filtering in the mid-1990s [SM93, RIS+94, HSRF95, SM95] when researchers started focusing on recommendation problems that explicitly rely on the ratings structure. The first approaches aimed mainly to present simple recommendations on news or music content to the users based on similar items they consumed before or other similar users did. Since then, much work has been done both in the industry and academia on developing new approaches to the recommender systems.

Due to the overload of information that we have nowadays, the interest in this area still remains high so it still constitutes an interesting research area that allows providing personalized recommendations to users helping them to overcome this problem.

2.1.1 Formulation

As stated in [AT05], the recommendation problem can be reduced to the problem of estimating ratings for the items that have not already been seen by a user. This estimation is usually based on the ratings given by the user to other items (content-based methods) or the ratings that similar users gave to these items (collaborative methods).

Once we can estimate ratings for the yet unrated items, we can recommend item(s) with the highest estimated rating(s) to the user.

We can define the formulation:

- C: set of all users;

- S: set of all possible items that can be recommended;
- u: utility function that measures the usefulness of item s to user c, i.e., $u : C \times S \rightarrow R$, where R is a totally ordered set (e.g., nonnegative integers or real numbers within a certain range). Then, for each user $c \in C$, we want to choose such item $s' \in S$ that maximizes the user's utility. More formally:

$$\forall c \in C, \quad s' = arg \max_{s \in S}(c,s) \tag{2.1}$$

In recommender systems, the utility of an item is usually represented by a rating, which indicates how a particular user liked a particular item, e.g., John Doe gave the movie "Fight Club" the rating of 9 (out of 10). However, in general, utility can be an arbitrary function, including a profit function. Depending on the application, utility u can either be specified by the user, as is often done for the user-defined ratings, or can also be computed by the application, as can be the case for a profit-based utility function.

Each element of the user space C can be defined with a profile that includes various user characteristics, such as age, gender, etc. In the simplest case, the profile can contain only a single (unique) element, such as User ID.

Similarly, each element of the item space S is defined with a set of characteristics. For example, in a movie recommendation application, where S is a collection of movies, each movie can be represented not only by its ID, but also by its title, genre, director, year of release, leading actors, etc.

The central problem of recommender systems lies in that utility u is usually not defined on the whole $C \times S$ space, but only on some subset of it. This means that u needs to be extrapolated to the whole space $C \times S$. In recommender systems, utility is typically represented by ratings and is initially defined only on the items previously rated by the users. Extrapolations from known to unknown ratings are usually done by specifying *heuristics* that define the utility function and empirically validating its performance and *estimating* the utility function that optimizes certain performance criterion, such as the *mean squared error*. Once the unknown ratings are estimated, actual recommendations of items to a user are made by selecting the N items with the highest rating among all the estimated ratings for that user.

2.2 Recommender Systems Classification

Various dichotomies have been proposed that classify recommender systems according to the philosophies of the underlying domains, the proposers, and their parent communities. Researchers have identified some main dimensions to help in the study of recommender systems [PGF04]:

- how the recommender system is modelled (content-based or collaborative);
- how the recommender system is targeted (to an individual, group, or topic);

- how the recommender system is built;
- how the recommender system is maintained (online versus offline);

The most popular and commonly accepted classification distinguishes between content-based and collaborative recommender systems. Also, this is the more adequate classification to apply to the proposed problem. In the following subchapters the content-based and the collaborative recommenders will be described as well as the hybrid systems that are a combination of both.

2.2.1 Content-Based Filtering

Content-based (CB) filtering methods try to recommend items that are similar to those that a user liked in the past. In this method, candidate items are compared with items previously rated by the user and the best-matching items are recommended.

CB recommenders form profiles for each user independently. With this, each user will only get recommendations based on the items he already rated. This brings the primary drawback of CB filtering systems: their tendency to over-specialize the item selection and only very similar items to previous items consumed by the user are recommended. This can be a big problem, because as stated by [BMCMB+10], users find recommenders more useful when they recommend unexpected items.

Content-based recommendation has its roots in *information retrieval* and *information filtering*. According to [Bel92], *information filtering* (IF) and *information retrieval* (IR) are almost identical at an abstract level. IR, in general, is the process of searching for and extracting specific information from a collection of information items. IR uses queries as specifications of information needs, and is concerned with single uses of the system whereas information filtering employs user profiles to satisfy long-term goals of the users. While retrieval denotes “finding” the most relevant data, filtering carries a connotation of “removing” data that is irrelevant for the user [Mir01].

At its core, content-based recommendation is based on the availability of *item descriptions* (manually created or automatically extracted) and a *user profile* that describes the user’s interests, assigning the importance to item characteristics. As stated before, similarly to item descriptions, user profiles may also be automatically derived and “learned” either by analysing user behaviour and feedback or by asking explicitly about interests and preferences. The recommendation task then consists in comparing items to the user profile to determine which items match the user’s preferences best.

In conclusion, we can take the following advantages and drawbacks from these systems:

CB advantages:

- Do not require data on other users to achieve reasonable recommendation accuracy.
- Are not affected from the new user, cold-start and sparsity problems (see section 2.3).
- Are capable of recommending items to users with unique tastes.

- Do not suffer from first-rater problem, i.e., they are capable of recommending new and unpopular items to each and every user.
- Transparency: Explanations on how the recommender system works can be provided by explicitly listing content features or descriptions that caused an item to occur in the list of recommendations. Those features are indicators to consult in order to decide whether to trust a recommendation. On the other hand, in general, collaborative systems are black boxes, since the only explanation for an item recommendation is that unknown users with similar tastes liked that item.

CB drawbacks:

- Over-specialization: as these systems make recommendations only based on the users' habits, they have a tendency to specialize in a small group of items, making recommendations of items similar to the items already seen by the user with a limited degree of novelty.
- CB techniques have a natural limit in the number and type of features that are associated with the items they recommend. Some types of items that are not completely machine parsable (e.g. music, movies, TV programmes), need to have another source of information or human editors need to insert the features manually. This activity is highly subjective, expensive, time consuming and erroneous, so it can lead to bad recommendations.
- Polysemy (the situation when words have multiple meanings) and synonymy (when different words have the same meaning) of words further increase the difficulty of identifying relevant selections [RT08], and this can also make the system to ignore items that could be interesting for the user.

As mentioned before, this approach is already implemented on the developed system, and it has been proven that when used alone, its drawbacks are too much significant and the recommendations are not always relevant to the user. Therefore, on the next section other approaches will be explored.

2.2.2 Collaborative Filtering

Collaborative filtering (CF) approaches rely on the availability of user ratings information and make suggestions to a target user based on the items that similar users have liked in the past, without relying on any information about the items themselves other than their ratings.

Pure CF approaches do not exploit or require any knowledge about the items themselves. The obvious advantage of this strategy is that these data do not have to be entered into the system or maintained. On the other hand, using such characteristics to propose items that are actually similar to those the user liked in the past might be more effective [KRRS11].

As the name says, CF techniques are designed to model the social process of asking a friend for a recommendation (collaboration). There are two different approaches to make suggestions

[BMCMB+10, JZFF10]: (i) *user-based*: from the items that are liked by users similar to him/her; or (ii) *item-based*: from items that have received similar ratings to the items that the target user likes.

From a research perspective, these types of systems have been explored for many years, and their advantages, their performance, and their limitations are nowadays well understood. Over the years, various algorithms and techniques have been proposed and successfully evaluated on real-world and artificial test data.

The Tapestry system [GNOT92] introduced the idea (and terminology) of collaborative filtering and showed how both explicit annotation data and implicit behavioural data could be collected into a queryable database and tapped by users to produce personal filters [KRRS11]. Later, Resknik defended the term "recommender system" as a more generic terminology than collaborative filtering, because it can be possible to make a recommender system without any kind of collaboration among people.

Less than two years later, GroupLens [RIS+94], presented the *nearest-neighbour* based approach, that today still remains as one of the most important and used approaches. According to [JZFF10], a typical nearest-neighbour method can be separated into three steps: (i) decide a neighbourhood for the active user or item by similarity value, which is usually defined by the Pearson correlation; (ii) calculate a prediction from a weighted combination of selected neighbours' ratings for the active user or given item; (iii) a descending sorted list is made among these predications and a recommendation is generated by choosing the top N items in this list.

Collaborative recommendation techniques are often grouped into two classes: *memory-based* and *model-based* [BHK98, BMCMB+10, JZFF10].

Memory-based algorithms are *heuristics* that make ratings predictions based on the entire collection of items previously rated by users. These systems require all ratings, items, and users to be stored in memory.

Model-based algorithms use the collection of ratings to learn a model, which is then used to make ratings prediction. Models are developed using *data mining* and *machine learning* algorithms to find patterns based on the data. These systems periodically create a summary of ratings patterns offline.

The advantage of *memory-based* methods over *model-based* approaches is that they have less parameters to be tuned, while the disadvantage is that the approach cannot deal with data sparsity in a principled manner. Pure *memory-based* models do not scale well for real-world application. Thus, almost all practical algorithms use some form of pre-computation to reduce run-time complexity. As a result, current practical algorithms are either pure *model-based* algorithms or a hybrid of some pre-computation combined with some ratings data in memory. The most critical issue with *memory-based* algorithms is how to determine the similarity between two users. The two most popular approaches are the *correlation-based* approach and the *cosine-based* approach [BMCMB+10].

In conclusion, we can take the following advantages and drawbacks from these systems:

CF advantages:

- They can make unexpected recommendations to the users since they are not based on the users' preferences, but in items of users with similar tastes.
- They do not need a representation of items in terms of features; the only information that is necessary are the ratings given from the users to the items. Therefore, CF can be applied to virtually any kind of item.
- Scalability of the items database can be large since the technique does not require any human involvement for tagging descriptions or features.
- They can cope with cross-content recommendations such as making confident predictions of entirely different items to the user who have never rated such items in the past.
- The quality of the recommendations is improved over time.

CF drawbacks:

- Cold Start or First Raster Problem (see section 2.3): when a new item is added to the system, it cannot be recommended to any user until the item is rated for the first time.
- Sparsity: several applications contain millions of users and millions of items. In practice, even active users only rate a few items of the entire set and it results in a very sparse matrix in collaborative filtering, i.e., a CF matrix with a high percentage of empty cells. Because of this, CF systems may not locate successful neighbours and therefore generate weak recommendations.
- To have good recommendations with this type of systems, a very large number of users need to rate a very large number of items. In the beginning stages of the system this can be a problem, because the users do not feel motivated to express their preferences when the system cannot help them yet.
- Grey sheep: In a small or medium community of users, there are individuals whose opinions or tastes are unusual, so, they rarely receive accurate recommendations since they do not agree or disagree consistently with any of existing group of people.

2.2.3 Hybrid Systems

The term hybrid comes from the Latin word *hibrida* that means "the result of a mixed union". Hybrid systems combine multiple recommendation techniques together to produce its output. Usually, techniques of content-based filtering are combined with techniques of collaborative filtering with the main goal of using the advantages of both techniques and to overtake their drawbacks at the same time. As stated by [Mir01], hybrid recommender systems bring the advantages of different recommendation technologies together to improve accuracy of the predicted ratings.

For instance, CF methods suffer from new-item problems, i.e., they cannot recommend items that have no ratings. This does not limit content-based approaches since the prediction for new items is based on their description (features) that are typically easily available [Bur02]. Therefore, with the combination of these techniques into a hybrid system is possible to create a system that can make collaborative recommendations and at the same time does not suffer from the new-item problems.

Besides the combination of different types of techniques, it's also possible to combine different techniques of the same type, for example, two different content-based recommenders could work together. A number of projects have investigated this type of hybrid [Bur07], e.g.: NewsDude, which uses both naive Bayes and kNN classifiers in its news recommendations. However, this dissertation will be particularly focused on recommenders that combine different types of techniques (CB and CF), since these are the most commonly implemented ones and those that hold the most promise for resolving some important problems, e.g., cold-start problem.

In order to exploit the advantages of available recommendation methods, several hybrid approaches have been proposed, in their vast majority concerning combinations of collaborative filtering and content-based filtering. Robin Burke in [Bur02, Bur07] proposed a classification of hybrid techniques into seven classes: *weighted*, *switching*, *mixed*, *feature combination*, *cascade*, *feature augmentation* and *meta-level*. This classification was already adopted by several researchers [BMCMB+10] and more recently an abstraction of three base designs was proposed by [JZFF10] to organize these seven classes: *monolithic*, *parallelized* and *pipelined hybridization design*.

2.2.3.1 Monolithic hybridization design

Monolithic hybrid systems consist of a single recommender component that integrates multiple approaches by pre-processing and combining several knowledge sources, opposite to the other two designs that combine results from several components.

With this design the hybridization is achieved by a built-in modification of the algorithm behaviour to exploit different types of input data. As stated in [JZFF10], data-specific pre-processing steps are used to transform the input data into a representation that can be exploited by a specific algorithm paradigm.

According to Robin Burke's taxonomy [Bur02], it is possible to assign to this category *feature combination* and *feature augmentation* strategies. In figure 2.1 it is possible to see the basic design of a monolithic system.

Feature combination

In this kind of systems, a single recommendation algorithm is provided with features from different recommendation data sources.

Some examples of the application of this technique can be found on [BHCO98], where inductive rule learner Ripper was applied to the task of recommending movies using both user ratings

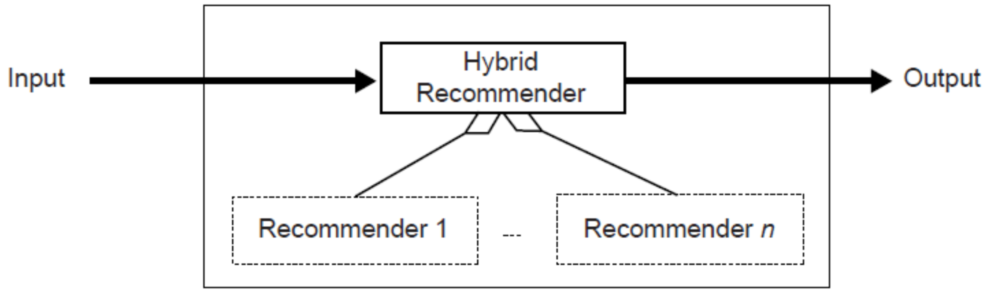

Figure 2.1: Monolithic hybridization design (obtained from [JZFF10])

and content features, and achieved significant improvements in precision over a purely collaborative approach. In [ZJ09b] different types of rating feedback based on their predictive accuracy and availability were exploited. In [Paz99] Pazzani used demographic user characteristics to bootstrap a collaborative recommender system when not enough item ratings were known.

Feature augmentation

In feature augmentation strategies, the output from one technique is used as an input feature to another. In contrast with feature combination, this hybrid does not simply combine and pre-process several types of input, but rather applies more complex transformation steps [JZFF10]. The output of a contributing recommender system augments the feature space of the actual recommender by pre-processing its knowledge sources. However, this must not be mistaken with a pipelined design that will be explained next.

Content-boosted collaborative filtering [MMN01] is an actual example of this variant. Libra system presented in [MR99] is also a good example of this approach.

2.2.3.2 Parallelized hybridization design

Parallelized hybridization designs employ several recommenders side by side and employ a specific hybridization mechanism to aggregate their outputs. *Switching*, *mixed* and *weighted* hybrid systems (HS) are differentiated from the remaining techniques in Burke's taxonomy by the fact that each of the individual (base) recommendation methods produces a prediction, independently from each other.

Figure 2.2 shows the basic design of a paralellized system.

Mixed

In mixed HS, predictions from each recommendation technique is presented to the user together. This is good in systems where is practical to make a large number of recommendations simultaneously.

The top-scoring items from each recommender are then displayed to the user next to each other, as in [BHY97]. However, when composing the different results into a single entity, such

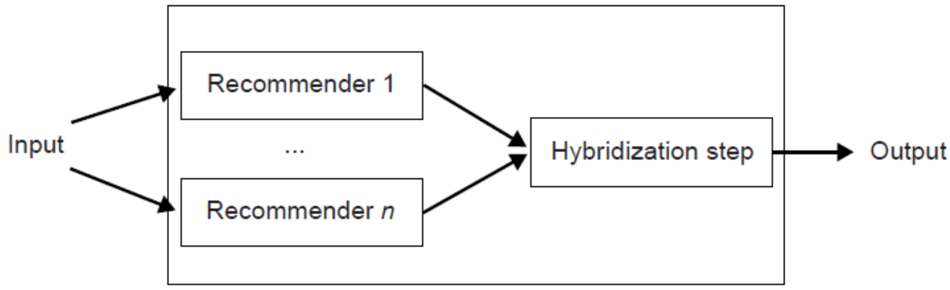

Figure 2.2: Parallelized hybridization design (obtained from [JZFF10])

as a television viewing schedule, some form of conflict resolution is required. In the personalized television application domain, [CS00] apply predefined precedence rules between different recommender functions.

Another form of a mixed HS was also presented [JZFF10], which merges the results of several recommendation systems: it proposes bundles of recommendations from different product categories in the tourism domain, in which for each category a separate recommender is employed.

Weighted

In this kind of systems, the scores (or ratings) of several recommendation techniques are combined together to produce a single recommendation.

For example, the simplest combined hybrid would be a linear combination of recommendation scores, e.g., the P-Tango system [CGM+99] applied to the news domain. It initially gives collaborative and content-based recommenders equal weight, but gradually adjusts it as the predictions about user ratings changes.

Switching

Switching hybrids decide which recommender should be used in a specific situation, depending on the user profile and/or the quality of recommendation results. For instance, to overcome the cold-start problem, a content-based and collaborative switching hybrid could initially make content-based recommendations until enough rating data are available. When the collaborative filtering component can deliver recommendations with sufficient confidence, the recommendation strategy could be switched.

Switching hybrids introduce additional complexity into the recommendation process, since the switching criteria must be determined and this introduces another level of parametrization. However, the benefit is that the system can be sensitive to the strengths and weaknesses of its constituent recommenders [Bur02].

Examples of switching HS are the NewsDude system [BP00] where two content-based variants and a collaborative approach are switched to recommend news, [ZJ09a] that switches between two hybrid variants of collaborative filtering and knowledge-based recommendation, and the DailyLearner system [Bur02] that uses a content/collaborative hybrid in which a content-based recommendation method is employed first.

2.2.3.3 Pipelined hybridization design

As stated in [JZFF10], pipelined hybrid systems implement a staged process in which several techniques sequentially build on each other before the final one produces the final recommendations for the user. The pipelined hybrid variants differentiate themselves mainly according to the type of output they produce for the next stage. In other words, a preceding component may either pre-process input data to build a model that is exploited by the subsequent stage or deliver a recommendation list for further refinement.

Figure 2.3 presents the basic design of a paralellized system.

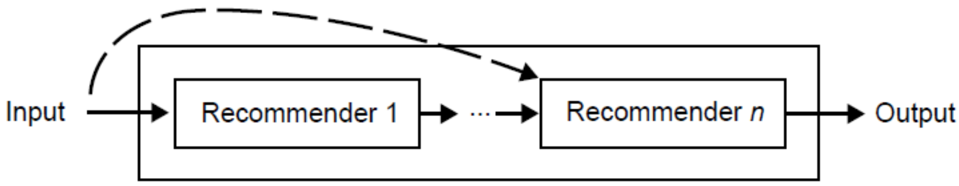

Figure 2.3: Pipelined hybridization design (obtained from [JZFF10])

Cascade

Unlike the previous methods, the cascade hybrid involves a staged process. In this method, several techniques sequentially build on each other before the final one produces recommendations for the user. Cascading strategies have the unfavourable property of potentially reducing the size of the recommendation set as each additional technique is applied, since each recommender cannot introduce new items on the list, it only can change the order of the list of recommended items.

One example of a cascading system is EntreeC [Bur02]: a knowledgebased restaurant recommender that is cascaded with a collaborative filtering algorithm to make recommendations based on the user's stated interests. The recommendations are placed in buckets of equal preference, and the collaborative technique is employed to break ties, further ranking the suggestions in each bucket.

Meta-level

In this type of approach, the entire model produced by one recommendation technique is used as input by another. As stated in [JZFF10], this differs from feature augmentation: in an augmentation hybrid, a learned model is used to generate features for input to a second algorithm; in a meta-level hybrid, the entire model becomes the input.

For instance, the Fab system [BS97] exploits a collaborative approach that builds on user models that have been built by a content-based recommender. LaboUr system [SKJ01] uses instance-based learning to create content-based user profiles which are then compared in a collaborative manner. More recently, Zanker [Zan08] evaluated a further variant of meta-level hybridization that combines collaborative filtering with knowledge-based recommendation.

2.2.3.4 Conclusions

In summary, no single hybridization variant is applicable in all circumstances, but it is well accepted that all base algorithms can be improved by being hybridized with other techniques.

The lack of appropriate datasets can explain the fact that little research has been conducted on comparing different recommendation strategies and especially their hybrids. In this aspect, the domain of application plays a decisive role. Whereas recommendation engines operating in areas such as movies and news consumption are comparably well researched, other application domains for recommender systems and algorithm paradigms have received less attention. Therefore, no empirically generalized conclusions about the advantages and disadvantages of different hybridization variants can be drawn. Depending on the application domain and requirements, different variants should be explored and compared.

Monolithic designs are advantageous if little additional knowledge is available for inclusion on the feature level. They typically require only some additional pre-processing steps or minor modifications in the principal algorithm and its data structures.

Parallelized designs are the least invasive to existing implementations, as they act as an additional post processing step. Nevertheless, they add some additional runtime complexity and require careful matching of the recommendation scores computed by the different parallelized algorithms.

Pipelined designs are the most ambitious hybridization designs, because they require deeper insight into algorithm's functioning to ensure efficient runtime computations. However, they typically perform well when combining two recommendation strategies which follow so dissimilar paradigms as is the case of collaborative filtering and content-based recommendation.

2.3 Recommender Systems Issues

2.3.1 Content quality

Content-based approaches depend on the characteristics of the items to make recommendations to the users. So, when the items do not have enough information to characterize themselves, the system cannot discriminate items the user likes from items that the user does not like and therefore cannot make good recommendations.

Recommender systems can also analyse the data, extracting info from a low level and inferring high level characteristics. However, this can be a very complex process that consumes many resources, and is an area still under research.

Additionally, the phenomena of polysemy (the situation when words have multiple meanings) and synonymy (when different words have the same meaning) further increase the difficulty of identifying relevant selections [RT08], preventing the system from considering items that could be interesting for the user.

2.3.2 Cold Start

The cold start problem occurs when a new object is presented to the system. This object can be an item to be recommended or a user. Depending on the type of the system, a new user or a new item can have a different impact.

In collaborative approaches, the *new item* problem (further detailed later in this section) occurs because when a new item comes to the system, is impossible to be recommended to the users before it is rated by someone.

In turn, the addition of a *new user* into the system is problematic in content-based approaches. When this approach is adopted, the system is able to produce a list of recommendations by matching the characteristics of candidate items to relevant characteristics in the user's profile. Normally, the user profile is built based on the history or previous consumptions of the user. A new user will not have any history and thus, unless the new user dedicates sufficient time and effort to insert information concerning his preferences in his profile, the system will not have the possibility of making any relevant recommendations to him [BO11]. Alternatively, it could switch to a default operating mode, in which it would recommend items exhibiting the most popular features, until the profile of the new user would have a minimum amount of information concerning his consumption preferences. This phenomenon is known as the *new user* problem and will be further detailed in this section.

2.3.3 New Item/First-Rater

The new item issue (also known as first-rater or early-rater problem) affects mostly collaborative-based systems. When a new item emerges, it cannot be recommended to a user before a user rates it [BO11]. This issue is clearly identified in collaborative filtering, since when a new item with no user assessment is inserted, it cannot be recommended. On the other hand, in content-based filtering, knowing the contents of an item is enough to enable a recommendation to a user, so these systems are not affected by this problem.

As stated in [Bur02], this problem is especially noted in domains such as news articles where there is a constant stream of new items and each user only rates a few. To try and solve this problem, recommender systems provide other incentives to encourage users to provide ratings or they implement methods to implicitly infer the user's rating without the need of an explicit rating.

2.3.4 New User

In content-based approaches, to be able to make accurate predictions, the system should be able to match the characteristics of a candidate item to relevant characteristics in the user's profile.

To accomplish this, a model with sufficiently detailed information on the user, including his/her tastes and preferences must first be built. Only when the system has a well-built model of the users' preferences, it can start making accurate predictions to the user. So, when a new user comes to the system and only rated few items, the system cannot understand his tastes and make accurate recommendations.

2.3.5 Grey Sheep

Grey sheep is the most common designation to the unusual user problem. This issue occurs on collaborative-based systems when a user has rare tastes, i.e., he does not agree or disagree consistently with any other users in the system. So, these individuals rarely receive accurate collaborative recommendations even when the critical mass of users is achieved [RT08].

In content-based filtering systems, even if the user has rare preferences, the recommendation of items is not a problem, since the system only makes recommendations based on his profile and does not depend on other users to make it.

2.3.6 Scalability

Different systems have different amounts of data. Some systems may reach millions of users and millions of items. When the quantity of users, items and evaluations is too large, systems that execute real-time calculations may provide a very long response time and may need computer resources that are simply not available. This is a common problem that affects both approaches. However, in collaborative filtering, this issue is more evident as the calculations are done using all users and all items. On the other hand, in content-based filtering, calculations are done using only one user and all related items, considering all attributes.

Linden et al. showed a comparison on [LSY03] about some existing methods to try to solve some scalability problems in very large data sets, but they show that these methods can fall short. They state that a scalable recommendation algorithm must perform the most expensive calculations offline. But this is not always possible, because generally, traditional collaborative filtering does little or no offline computation. Other methods like *dimensionality reduction*, *sampling*, *partitioning* or *clustering* can be used to solve some scalability issues, but they usually reduce the recommendation quality.

2.3.7 Sparsity

As stated before, sparsity is a problem that affects collaborative filtering systems. These systems make predictions based only on the rating that users assign to items. Some systems may have millions of users and items. In practise, users, even the active ones, only rate a few items from the entire set of items. This results in a very sparse matrix in a collaborative filtering approach, i.e., a CF matrix with a high percentage of empty cells. As a consequence, the system can have problems in locating neighbours to the user, thus generating weak recommendations.

As stated in [BO11], in content-based filtering the recommendations do not depend on the number of users and items, only on the users' preferences. Therefore this does not represent a problem in this kind of systems.

2.3.8 Over-Specialization

Over-specialization is a problem that can be clearly identified in content-based filtering. In this approach, when a user profile is defined, all the recommendations will be based on the items that the user liked in the past. So, only similar items to those he liked will be recommended, therefore is difficult to recommend really unexpected items to the user.

Related to the over-specialization, some authors use the term *serendipity* [BO11, JZFF10, Zan08]. Serendipity is a measure of how surprising the successful recommendations are [KRRS11]. In collaborative filtering, item recommendation is not based on the user's initial profile, but rather on his/her relation to other users. So, the surprise occurs more frequently, since similar users may have evaluated some items in a completely different manner or may have consumed different items than the original user.

2.4 Recommender Systems Techniques

Different methods are used to build recommender systems. This chapter presents an overview about some of the most important approaches that are widely used in the literature, especially those that were already been applied in systems similar to the proposed one.

These approaches are used in the classification process. A classifier is a mapping between a feature space and a label space, where the features represent characteristics of the elements to classify and the labels represent the classes [KRRS11]. A restaurant recommender system, for example, can be implemented by a classifier that classifies restaurants into one of two categories (good, bad) based on a number of features that describe it.

2.4.1 kNN

k-Nearest-Neighbour (kNN) is one of the most widely used approaches to build CF recommender engines. Given a point to be classified, the *kNN* classifier finds the *k* closest points (*nearest neighbours*) from the training records. It then assigns the class label according to the class labels of its nearest-neighbours. The underlying idea is that if a record falls in a particular neighbourhood where a class label is predominant it is because the record is likely to belong to that very same class.

kNN classifiers are amongst the simplest of all machine learning algorithms. Since kNN does not build models explicitly it is considered a lazy learner. Unlike eager learners (e.g. decision trees or rule-based systems), kNN classifiers leave many decisions to the classification step. Therefore, classifying unknown records is relatively expensive - this is one of the main drawbacks of this approach [KRRS11].

On the other hand, one of the advantages of this classifier is that it is conceptually very much related to the idea of CF: finding like-minded users (or similar items) is essentially equivalent to finding neighbours for a given user or an item. The other advantage is that, being the kNN classifier a lazy learner, it does not require to learn and maintain a given model. Therefore, the system can adapt to rapid changes in the user ratings matrix - this is a very big advantage of this kind of approaches, especially when they are applied to systems that change the number of users or items very quickly.

The kNN approach, although simple and intuitive, has shown good accuracy results and is very amenable to improvements. As a matter of fact, its supremacy as the *de facto* standard for CF recommendation has only been challenged recently by approaches based on dimensionality reduction [KRRS11].

2.4.2 Decision Trees

Decision trees (DT) [RM08] are classifiers on a target attribute (or class) in the form of a tree structure. The observations (or items) to classify are composed of attributes and their target value. The nodes of the tree can be: a) *decision nodes*, in these nodes a single attribute-value is tested to determine to which branch of the sub-tree applies; or b) *leaf nodes* which indicate the value of the target attribute.

The main advantages of building a classifier using a decision tree is that it is inexpensive to construct and it is extremely fast at classifying unknown instances [KRRS11].

Decision trees may be used in a model-based approach for a RS. One possibility is to use content features to build a decision tree that models all the variables involved in the user preferences. In [BRBG08] this idea is used to construct a DT using semantic information available for the items. The tree is built after the user has rated only two items. The features for each of the items are used to build a model that explains the user ratings. They use the information gain of every feature as the splitting criteria. It should be noted that although this approach is interesting from a theoretical perspective, the precision they report on their system is worse than that of recommending the average rating.

Besides the precision drawback, using DT it is very difficult and unpractical to build a decision tree that tries to explain all the variables involved in the decision making process.

2.4.3 Rule-based Classifiers

Rule-based classifiers (RBC) classify data by using a collection of "*if... then...*" rules. The rule antecedent or condition is an expression made of attribute conjunctions. The rule consequent is a positive or negative classification.

A rule r covers a given instance x if the attributes of the instance satisfy the rule condition. The *coverage* of a rule can be defined as the fraction of records that satisfies its antecedent. On the other hand, its *accuracy* is defined as the fraction of records that satisfies both the antecedent and the consequent. We say that a classifier contains mutually exclusive rules if the rules are independent of each other – i.e. every record is covered by at most one rule. Finally we say that the classifier has exhaustive rules if they account for every possible combination of attribute values – i.e. each record is covered by at least one rule [KRRS11].

The advantages of RBC are that they are extremely expressive since they are symbolic and operate with data attributes without any transformation. In a similar way to decision trees, RBC are easy to interpret, generate and they can classify new instances efficiently. How, it is also very difficult to build a complete recommender model based on rules. As a matter of fact, this method is not very popular in the context of RS because deriving a rule-based system means that we either have some explicit prior knowledge of the decision making process or that we derive the rules from another model such a decision tree.

So, instead of completely build recommender systems using RBC, they are used to improve the performance of a RS by injecting partial domain knowledge or business rules, as made in [LB03].

2.4.4 Artificial Neural Networks

An Artificial Neural Network (ANN) [Zur92] is an assembly of inter-connected nodes and weighted links that is inspired in the architecture of the biological brain. Nodes in an ANN are called neurons as an analogy with biological neurons. These simple functional units are composed into networks that have the ability to learn a classification problem after they are trained with sufficient data.

The main advantages of ANN are that, depending on the activation function, they can perform non-linear classification tasks, and that, due to their parallel nature, they can be efficient and even operate if part of the network fails. The main disadvantage is that it is hard to come up with the ideal network topology for a given problem and once the topology is decided this will act as a lower bound for the classification error. ANNs belong to the class of sub-symbolic classifiers, which means that they provide no semantics for inferring knowledge [KRRS11].

In [PB97] a comprehensive experimental study on the use of several machine learning algorithms for web site recommendation was made. Their main goal was to compare the simple naive Bayesian Classifier with computationally more expensive alternatives such as Decision Trees and Neural Networks. Their experimental results show that Decision Trees perform significantly worse. On the other hand, ANN and the Bayesian classifier performed similarly. They conclude that there does not seem to be a need for non-linear classifiers such as the ANN.

2.4.5 Matrix Factorization/Latent Factor Models

Matrix factorization methods can be used in recommender systems to derive a set of latent (hidden) factors from the rating patterns and characterize both users and items by such vectors of factors. In the movie domain, such automatically identified factors can correspond to obvious aspects of a movie such as genre or type (drama or action), but they can also be uninterpretable. A recommendation for an item i is made when the active user and the item i are similar with respect to these factors [JZFF10].

The idea of exploiting latent relationships in the data and using matrix factorization techniques such as *Singular Value Decomposition* (SVD) was relatively soon transferred to the domain of recommender systems [SKKR00]. They used SVD in RS to perform two different tasks: First, SVD captures latent relationships between customers and products that allow to compute the predicted likeliness of a certain product by a customer. Second, SVD produces a low-dimensional representation of the original customer-product space and then computes neighbourhood in the reduced space. Then they used that to generate a list of top-N product recommendations for customers.

The main advantage of these techniques is that they can increase the performance in systems with big datasets, since a reduced space to represent the user-item relationships is calculated, and then the computation is made using that matrix. They also are able to address the sparsity problem, finding these relationships in the data.

On the other hand, they have a drawback of loosing precision, since some items are grouped according to their relationships, what can lead to bad recommendations and even to discarded items during recommendation generation.

2.5 Metrics for Evaluation

Recommender system accuracy has been evaluated in the research literature since 1994 [RIS+94, HKTR04]. Since then, many metrics were proposed and used over the years. This section presents only the most popular metrics used by different authors, as well as those that have appeared more recently in the literature [JZFF10, BO11, KRRS11].

An efficient way to evaluate recommender systems is through the comparison of the generated predictions and the real evaluations made by the user. When evaluating the ability of a system to correctly predict a user's preference for a specific item, one of the most popular measures is the *Mean Absolute Error* (MAE) (2.2).

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|f_i - y_i| = \frac{1}{n}\sum_{i=1}^{n}|e_i| \tag{2.2}$$

As the name suggests, the mean absolute error is an average of the absolute errors $e_i = |f_i - y_i|$, where f_i is the computed prediction score and y_i the true rating value.

Another metric widely used is the *Root Mean Square Error* (RMSE) presented on equation (2.3).

$$x_{rms} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}x_i^2} = \sqrt{\frac{x_1^2 + x_2^2 + \ldots + x_N^2}{N}} \tag{2.3}$$

This metric was used on, for example, the Netflix prize competition[1], which offered a $1 000 000 reward for a reduction of 10% in the RMSE.

In the context of recommendation, the purpose of a classification task is to identify the n most relevant items for a given user. *Precision* and *Recall* are the two best-known classification metrics. These metrics are also used for measuring the quality of information retrieval tasks in general.

[1] See http://www.netflixprize.com/

Table 2.1: Classification of Items After Recommendation

	Recommended	Not Recommended
Used	True-positive (tp)	False-negative (fn)
Not used	False-positive (fp)	True-negative (tn)

Table (2.1) shows the possible classifications that user can give to items after recommendation. Precision and recall are calculated by asking the user to classify all the candidate items as being relevant or not to him. More precisely, the items that were included in the recommendation list and that the user has effectively considered relevant are marked as *true-positive*, whereas those that he has considered irrelevant are marked as *false-positive*. Likewise, items that were not included in the recommendation list but that the user has considered as relevant are marked as *false-negative*, whereas those that the user has effectively considered irrelevant are marked as *true-negative*.

Precision (2.4) is defined as the proportion of recommendations that were considered as relevant by the user, i.e., that are good recommendations, in relation to the total number of recommendations; *Recall* (2.5) is defined as the proportion of good recommendations that appeared in the list of recommendations, in relation to the total number of items that the user considered as relevant. Based on this definition and on table 2.1 it is possible to derive the formulas (2.4) and (2.5) to calculate these two measures.

$$Precision = \frac{tp}{tp + fp} \tag{2.4}$$

$$Recall = \frac{tp}{tp + fn} \tag{2.5}$$

A precision score of 1.0 indicates that all items included in the recommendation list were actually relevant to the user, thus they were all good recommendations (in this case, the number of false-positives would be 0). However, it does not provide any indication whether other relevant items among the universe of candidates were recommended or not. A recall score of 1.0 indicates that all items that were relevant to the user among the universe of candidates were actually included in the recommendation list (in this case the number of false-negatives would be 0). However, it does not provide any indication whether that list contained bad recommendations or not.

Combining both *Precision* and *Recall* there is the F_1 metric (2.6) [SKKR00, HKTR04].

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{2.6}$$

Comparative studies on commercial datasets have been conducted, for example, by [SKKR00]. Here, Bardrul Sarwar et al., use these metrics extensively to get a measure of accuracy in recommender systems obtaining good results.

2.5.1 Rating Normalization

Since each user has its own personal scale to assign ratings to items, in some cases it may be necessary to normalize the ratings. Especially with explicit ratings, some users can be reluctant to give low/high scores to items they liked/disliked.

There are two widely used methods to convert the ratings into a more universal scale: *mean-centering* and *Z-score*.

2.5.1.1 Mean-centering

This method determines weather a item is positive or negative by comparing it to the mean rating. In user-based recommendation, the rating r_{ui} given by the user is transformed to a mean-centered one $h(r_{ui})$ by subtracting to r_{ui} the average rating $\bar{r}_u$ of the items rated by the user (I_u):

$$h(r_{ui}) = r_{ui} - \bar{r}_u \tag{2.7}$$

2.5.1.2 Z-score

While mean-centering removes the offsets caused by the different perceptions of an average rating, Z-score normalization also considers the spread in the individual rating scales [KRRS11]. The normalization of a rating r_{ui} divides the user-mean-centered rating by the standard deviation σ_u of the ratings given by user u:

$$h(r_{ui}) = \frac{r_{ui} - \bar{r}_u}{\sigma_u} \tag{2.8}$$

2.6 Frameworks for Recommender Systems

A number of frameworks have been defined to assist the development and test of recommender systems. Some of these have been suggested and described in [Ram10]. In this section we provide a brief overview of the most popular ones.

2.6.1 CofiRank

CofiRank [2] is an approach to the collaborative filtering problem built upon the approach of Maximum Margin Matrix Factorization, that is able to run on the largest datasets available, to do structured prediction and can be easily parallelized to take advantage of multi core machines or clusters of workstations.

[2] See http://www.cofirank.org/

2.6.2 SUGGEST

SUGGEST [3] is a Top-N recommendation engine that implements two classes of collaborative filtering recommendation algorithms, called user-based and item-based. It can provide high quality recommendations and be applied to large-scale datasets.

2.6.3 C/Matlab Toolkit for Collaborative Filtering

This toolkit [4] is a set of C and Matlab functions implementing several methods of collaborative filtering. It also supplies functions for loading, handling and evaluating collaborative filtering methods.

[3] See http://glaros.dtc.umn.edu/gkhome/suggest/overview
[4] http://www.cs.cmu.edu/ lebanon/IR-lab.htm

Chapter 3

System Specification and Scientific Approach

As already explained, this dissertation was proposed and developed within the framework of the ongoing research project Hotel 3.0. Accordingly, the goals devised for the dissertation and its expected outcomes were in some way constrained by existing developments and approaches. The main objective set forth was to enhance the existing recommendation engine, both by modifying the adopted content-based approach, as well as by incorporating new functionalities to the system and increase its flexibility.

This chapter starts by providing an overview of the previously existing system, describing the aspects that had greater impact on the decisions taken and approaches adopted in the course of this dissertation to achieve the proposed goals. This allows to understand the limitations that can be pointed to the previous work, as well as the approach used to overtake them.

Section 3.1 starts by listing the requirements identified originally within the Hotel 3.0 project. It then describes the additional requirements that were defined specifically for the work to be conducted within the scope of this dissertation.

Section 3.2 presents some of the most important use cases that were defined to the system, to understand the main expected functionalities.

Section 3.3 provides an overview of the original system architecture, emphasizing the aspects more directly related to this dissertation, whereas section 3.4 describes problems that were identified in the original system.

Finally section 3.5 presents the approaches that were adopted in this dissertation to solve the listed problems and accomplish the proposed goals.

3.1 Requirements

The requirements for this system are described in this section. There are two kinds of requirements: functional and non-functional. In [Som07], functional requirements are defined as "statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations". Non-functional requirements are "constraints on the services or functions offered by the system. They include timing constraints, constraints on the development process and standards".

These requirements are presented in two groups: first, the global requirements for the project QREN Hotel 3.0. After, the specific requirements within the scope of this dissertation are presented.

3.1.1 Global Requirements

Table 3.1 presents the functional requirements defined on a earlier phase to this dissertation.

Table 3.1: Functional Requirements - Global System

Identifier	**Description**
FRG01	Content-based recommender system
FRG02	Read and characterize TV programmes data
FRG03	Generate recommendations for TV programmes
FRG04	User profile generation based on the user's history
FRG05	Recommender system get users' history from back-end
FRG06	Reorder menus according to the user's preferences
FRG07	Configuration file for system definitions and database configuration
FRG08	Create database to store users' information

Table 3.2 shows the non-functional requirements defined on a earlier phase to this dissertation.

Table 3.2: Non-functional Requirements - Global System

Identifier	**Description**
NFRG01	Extract information from different content-providers online
NFRG02	24/7 availability
NFRG03	Automatic and periodic operation
NFRG04	Automatic information acquisition and treatment

3.1.2 Specific Requirements

Table 3.3: Functional Requirements - Specific

Identifier	**Description**
FRS01	Implement a filtering technique that takes advantage of similarity between users
FRS02	Always try to provide serendipitous recommendations to users
FRS03	Implement a hybridization technique to combine CF with CB
FRS04	Allow the system to treat other types of information (VoD and Services)
FRS05	Generate recommendations for different content types

Table 3.3 presents the functional requirements within the scope of this dissertation. These requirements were defined to overtake some limitations already pointed in 1.2: for example, the functional requirement FRG01 defined on table 3.1, has the flaw of rapidly become very focussed, providing the user always the same type of recommendations ("over-specialization" problem). To solve this problem, the functional requirements FRS01, FRS02 and FRS03 were defined. As stated before in 2.3.8, serendipity is a measure of how surprising the successful recommendations are [KRRS11]. As this recommender system is to be applied in the hospitality market, is necessary to always provide new and surprising recommendations to users, to make them satisfied and at the same time try to increase the profit.

On the other hand, FRS04 and FRS05 were addressed to overcome one of the main limitations of QREN Hotel 3.0 framework, as result of the defined in FRG02 and FRG03.

Table 3.3 specifies the non-functional requirements for this dissertation.

Table 3.4: Non-functional Requirements - Specific

Identifier	**Description**
NFRS01	Implement a CF technique using the existing user profile structure with as minimum changes as possible
NFRS02	Implement a CF technique with low complexity to avoid loss of performance
NFRS03	Allow adding new content types to the system at any time more simply
NFRS04	Allow the generation of recommendations for new content types using the same technique easily

The non-functional requirement NFRS01 is presented in addiction to FRS01. As the Hotel 3.0 recommender system was originally designed to work only with a content-based approach, all developments were made thinking on this. To improve the already working system without having a big impact and cause several architectural and functional changes, NFRS01 was defined to ensure that the chosen CF approach can work together with the existing CB technique without the need of major changes.

As this recommender system can be running in a platform with low computational resources, is necessary to ensure that the implemented hybrid approach has a good performance in comparison with the original CB approach. NFR03 was defined to ensure this.

In the IT world, every day there is an emerging need to adapt the systems to new realities and requirements. Besides adding to the system the ability to deal with new types of resources, as defined in FRS03 and FRS04, the requirement NFRS04 was also identified to ensure that a new resource (e.g., Radio Channels) can be added to the system with minimum effort and be used in the same way as the previously existing resources when generating recommendations.

3.2 Use Cases

This section presents some of the most important use cases (UC) that were defined to the system, to understand the main expected functionalities.

The UC model is a catalogue of system functionality described using UML. Each UC represents a single, repeatable interaction that a user or "actor" experiences when using the system. *Actors* are the users of the system. Each Actor will have a well-defined role, and in the context of that role have useful interactions with the system. A person may perform the role of more than one Actor, although they will only assume one role during one use case interaction. An Actor role may be performed by a non-human system, such as another computer program or module. Use Cases may include other Use Cases as part of a larger pattern of interaction and may also be extended by other use cases to handle exceptional conditions.

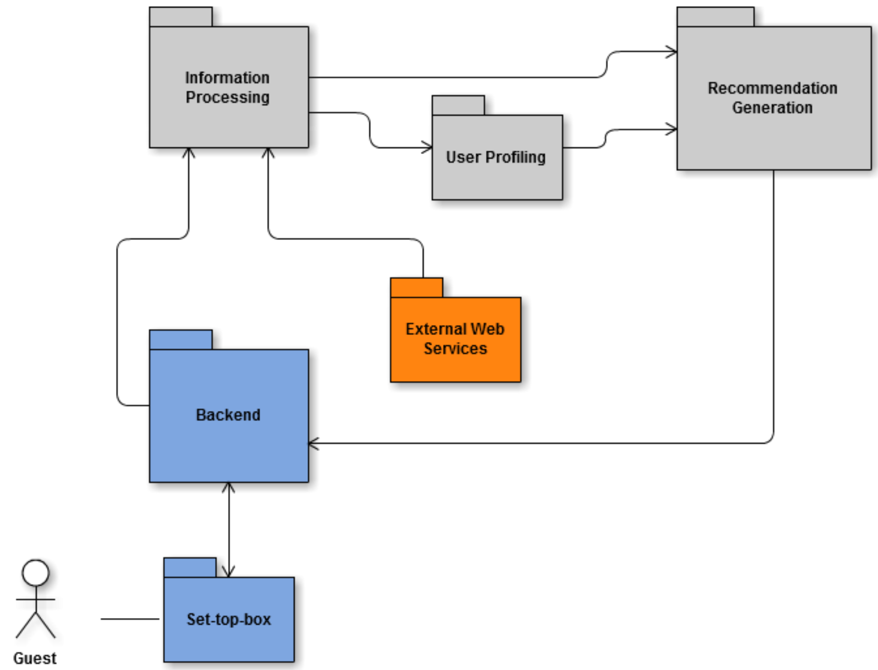

Figure 3.1: High Level Architecture

Figure 3.1 illustrates an overview of the system high level architecture. Here are represented the main modules of the system. These modules are going to present the role of "actors" in the UC model, since their main actions are going to be described.

These actors and the main use cases are illustrated in figure 3.2.

Table 3.5 presents a brief description about the most important use cases previously shown in figure 3.2. These use cases presented here are relative to the whole system, not only the recommender engine. Therefore, not every use cases are executed by the recommendation engine. For example, the functionality correspondent to the use case *sendInteractionBackend* shown on table 3.5 is executed by the set-top-box, so it was not developed within the scope of this dissertation.

The use cases that represent functionalities developed to the recommender engine have as Primary Actor *Information Processing*, *User Profiling* and *Recommendation Generation*, which represent the main modules of this recommender.

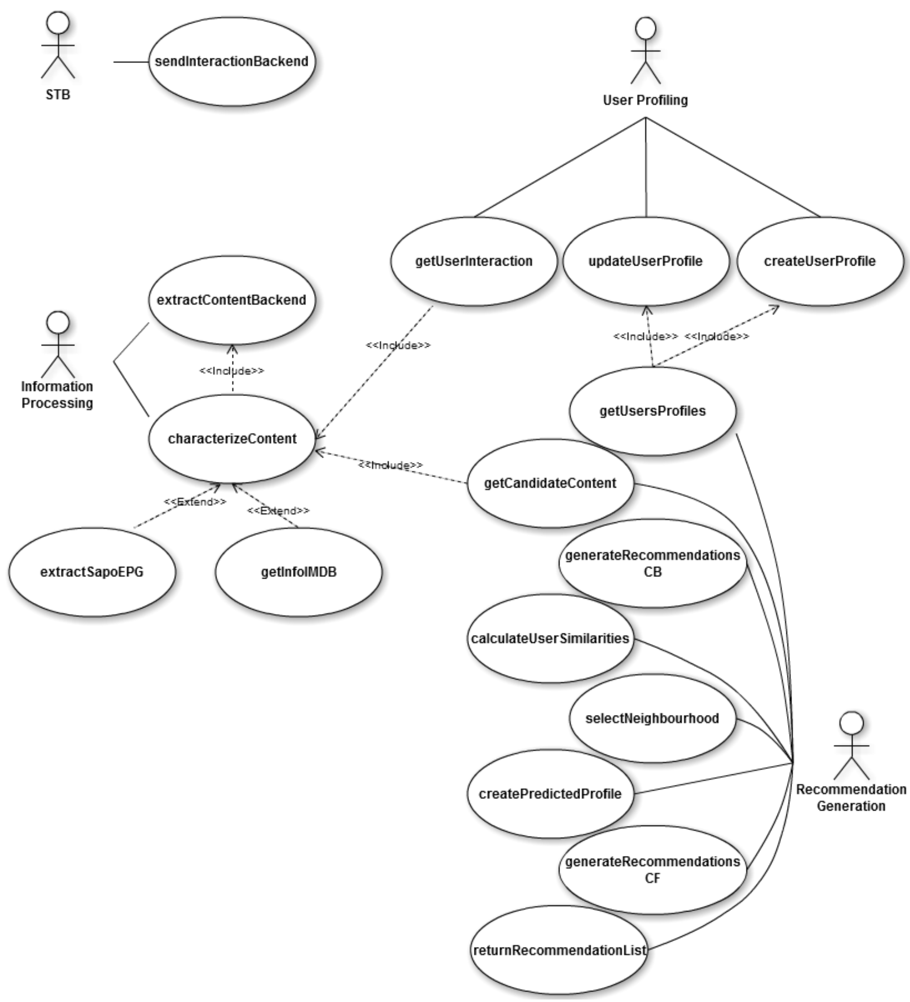

Figure 3.2: Use Cases Diagram

Table 3.5: Brief Description for Most Important Use Cases

UC Name	**Impact in the system**	**Primary Actor**	**Secondary Actor**
sendInteractionBackend	Set-top-box sends users' interaction history to the backend, so that it can later be sent to the "Information Processing" module.	STB	Backend
extractContentBackend	Request all data from backend and parse it so that it can be analysed by the recommender. This information is about *users*, *available channels*, *VoD*, *services* and *user interaction*; it is provided in the XML format.	Information Processing	Backend
characterizeContent	Characterized the previously extracted content from backend, adding features from the domain to it.	Information Processing	
extractSapoEPG	Extracts the TV schedule from Sapo EPG external Web service.	Information Processing	External Web Services
getInfoIMDB	Extracts info about a specific title from IMDb API.	Information Processing	External Web Services
getUserInteraction	Requests the user interaction data from "Information Processing".	User Profiling	Information Processing
createUserProfile	Creates a new user profile based on his interaction previously obtained.	User Profiling	
updateUserProfile	Updates an existing profile based on the new interaction.	User Profiling	
getCandidateContent	Request the characterized items from "Information Processing" to use them as candidate content in recommendation generation.	Recommendation Generation	Information Processing
getUsersProfiles	Requests users' profiles from "User Profiling" to use them on recommendation process.	Recommendation Generation	User Profiling
generateRecommendationsCB	Generates recommendations using content-based approach (Naive Bayes) for the previously obtained user profiles and candidate content.	Recommendation Generation	
calculateUserSimilarities	Executes the similarity weight computation to find the neighbours for all users in the list received from "User Profiling" module.	Recommendation Generation	
selectNeighbourhood	For each user filters the previously calculated neighbours, selecting only the most similar to each target user.	Recommendation Generation	
createPredictedProfile	Creates a new collaborative predicted profile based on the user's neighbourhood.	Recommendation Generation	
generateRecommendationsCF	Generates recommendations using content-based approach (Naive Bayes) for the previously calculated "Predicted Profile".	Recommendation Generation	
returnRecommendationList	Returns both recommendations (CB and CF) after the hybridization step to backend.	Recommendation Generation	Backend

3.3 Previous Work

3.3.1 Overview

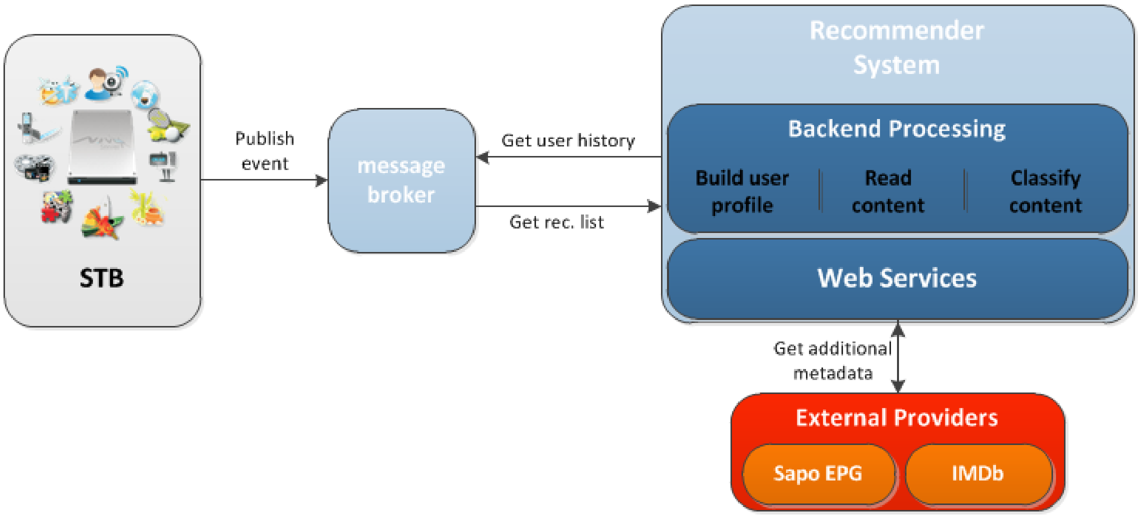

Figure 3.3: System Overview

In figure 3.3 it's possible to see the system overview. This only illustrates the system operation in a global point of view; more detail about the system architecture will be given in chapter 4.

The set-top-box (STB) provides all available services to the end user (watch TV, buy hotel services, etc.) and keeps track of his interactions. Then, when needed, this STB sends and event to a message broker. A message broker is an intermediary program that translates a message from the formal messaging protocol of the sender to the formal messaging protocol of the receiver. When this request is "get recommendation list", the message broker sends it to the recommender system, that retrieves the recommendation list.

The recommender system is responsible for all the information treatment. For example, it parses data related with users and TV channels from Web services to update its own database. Besides that, it obtains daily all the data from the external Electronic Program Guide (EPG) retrieved from the Sapo EPG. Then it characterizes the obtained data, to associate key features to each program. When there is not enough information to characterize a program, it also uses information obtained from IMDb API, for example to characterize a movie.

To build the users' profiles, the system obtains the user history from the provided Web service and gets the program features according to the viewing history (this step will be explained in more detail in 4.2.4).

After having the generated user profile, the system can generate recommendations for other programmes with similar content to the previously viewed by the user, and retrieve that list of recommendations to the STB.

3.3.2 User Profiling

The user profiling technique is only implicit, since there isn't any kind of explicit feedback given by the user. This technique was based in the implicit profiling presented in [YZ04]. This method stores for each user some features previously obtained from TV program characterization, each one associated with a weight. When a user has watched a TV program for a period of time and switched to another one, the user profile will be refined and revised based on the implicit feedback automatically observed by the system. The algorithm is the following:

1) If a term already exists in the profile, its weight is modified as the following:

$$w_i' = (1-\alpha) \times w_i + \alpha \times \Delta w_i \tag{3.1}$$

$$\Delta w_i = \beta \times f(i) \tag{3.2}$$

$$\beta = \frac{T_r}{T_t} \in [0,1] \tag{3.3}$$

$$f(i) = \frac{I_{max} - i}{I_{max}} \quad 1 \leq i \leq I_{max} \tag{3.4}$$

2) If a term does not exist in the profile:

$$w_i = \alpha \times \beta \times f(i) \tag{3.5}$$

Where:

- w_i' if the weight of term t_i after update;
- w_i if the weight of term t_i before update;
- $\alpha \quad (0 \leq \alpha \leq 1)$ is the learning rate that determines how quickly the user profile forgets old preferences and tracks new ones;
- β is the ratio of user's real watching time (T_r) to the program's total duration (T_t);
- i is the order of the term t_i;
- $f(i)$ reflects the influence of the order in user's viewing history to the weight update process; $f(i)$ should decrease with increasing order of the term. If the term ti is not in the profile before, $f(i)$ as a default value ε;
- I_{max} is the maximum of i, i.e., the total number of terms in the user's profile.

Using this algorithm, the approach was to create two different profiles for each user: the "Accepted Profile", which registers the weights of the features that the user likes and; the "Rejected

Profile", which registers weights for the features that the user dislikes. It should be noted that it is possible that the same feature appears in both profiles.

The two profiles are updated taking into consideration the ratio of viewing time to the total duration of the program, adopting the following rule:

- If $\beta \geq 70\%$ the feature and the associated weight will be added to the "Accepted Profile";
- If $\beta < 70\%$ the feature and the associated weight will be added to the "Rejected Profile".

3.3.3 Content-based Recommendation

The CB technique used in the system is based on the Naive Bayes (NB) classifier. This method is well known from the literature and widely used in several applications [FGG97].

Some examples are given bellow:

- [MR99] in a text classifier;
- [MK08] in a keyphrase extraction algorithm;
- [SKJ01] to calculate the probability of user interest for a given document;
- [GCP03] to calculate the probability of words being in the same document;
- [Gha10] to create a switching hybrid recommender system using NB and CF;
- [Paz07] as a text classification algorithm.

The Naive Bayes classifier that was implemented makes use of the user profiles described in 3.3.2. When a candidate program is analysed by the system, its features are compared with those contained in the user profile. Then, the user preference for that program is predicted according to the user's accepted and rejected features.

3.4 System Problems

Prior to initiate the developments within the context of this dissertation, the already existing system was analysed to detect eventual problems or deficiencies that could jeopardize the successful achievement of this dissertation. As it was already explained, the outcomes of this work were meant to be integrated into the existing system, with the goal of enhancing its operation. This section describes some of the major problems found in the existing system, notably those that were considered as having more impact within the scope of this dissertation.

3.4.1 User Profiling Problems

The problems listed here below, were identified as the ones that could potentially have a stronger negative impact on the development of this dissertation:

- Feature weight update process not appropriate for the data available in the system;
- Threshold definition (70%) that defines update in "Accepted Profile" or "Rejected Profile" causes big problems in the recommendation when the profile starts being updated with features that already existed.

The first problem is mainly related with the use of the factor $f(i)$. As stated before, $f(i)$ reflects the influence of the order in user's viewing history to the weight update process; $f(i)$ should decrease with increasing order of the term. The terms that are evaluated in the system correspond with the features that were associated to a program during the characterization process. During this process, all these features are related with a program giving the same importance to all the features. So, during the user profiling, there is no need to give different weight to a feature depending in its order on the feature list.

Table 3.6: User Viewing Example

Program	**Feature list**	**Watching ratio (%)**	**Date**
P1	Comedy	100%	Jan 2011
P2	Comedy	100%	Feb 2011
P3	Comedy	100%	Mar 2011
P4	Comedy	100%	Apr 2011
P5	Comedy	100%	May 2011
P6	Comedy	100%	Jun 2011
P7	Comedy	20%	Jul 2011
...	..	...	...
Px	Comedy	0.1%	Nov 2011
Py	Comedy	0.1%	Dec 2011

Table 3.7: User Accepted Profile

User	**Accepted Feature**	**Ratio**
User 1	Comedy	1

Table 3.8: User Rejected Profile

User	**Rejected Feature**	**Ratio**
User 1	Comedy	0.001

The second identified problem is related to the inability of the system to adequately follow changes in time in the user's preferences. This can be better explained by looking at the example given in tables 3.6 to 3.8. The user whose viewings and profiles are listed in those tables, used to have a clear preference for "Comedy" programmes as the system registered several times, a 100% (full-time) visioning of programmes of that genre during the period Jan-Jun 2011. As a consequence, the corresponding feature in the Accepted user profile reached a high value. However, after that period, the user seemed to have lost interest in that genre, as the system during the period Jul-Dec 2011, only detected related viewings of a very limited amount of time (or simply did not detect at all any viewing of that genre). When the amount of viewing time is less than

70% of the complete duration of the program, the system does not consider it as a preference for updating the user profile as an accepted feature. Instead it uses it to update the "Rejected Profile", which then sees the value of the corresponding feature decaying to near 0. However the "Accepted Profile" is not updated and thus the "Comedy" feature still maintain its high value. This leads the system to believe that the user still has preferences for that genre and thus to probably recommend comedies to the user.

The solution adopted to solve this problem is explained in section 4.3.

3.4.2 Other Problems

Besides the user profiling problem, there were other kinds of problems found in the system that needed to be solved to make the system work properly.

One of these problems affected indirectly the user profiling. The processing of user behavioural data (explained in more detail in section 4.2.4) parses the user channel interaction, i.e. the time interval for each channel viewed by the user, to get the watching ratio (β) for each program. During the system testing (explained in chapter 5) some problems were detected in this step. Some β values did not correspond to the actual watching ratio, which translated in wrong weight values for some features.

During the system testing other kinds of problems were detected. Initially the system was collecting data from the Sapo EPG corresponding to only 6 channels. When the volume of information increased and the system started to collect daily data from 107 channels, new problems arose. Those problems were mainly related to memory leaks and the use of wrong containers for the data, but also related to the characterization process.

The system robustness was also improved, notably in the connection to external Web services, allowing the system to continue to work properly even in cases when those Web services were not available.

Although these problems were not directly related with the goals of this dissertation, they were all solved.

3.5 Approach

This section describes the main contributions of this dissertation. It explains the scientific approach that was followed to address the identified challenges, briefly describing the solutions adopted to fulfilling the defined requirements. The following list provides an indication of the main scientific topics addressed and the contribution provided by this dissertation to each one.

1. **Features domain**
   The first identified challenge was related to the obtention of a compact, yet complete and flexible, structure and set of tags to characterize and represent content. Given that the area of application covered a broad range of resources (from TV programmes to hotel services,

through Internet Radio and Hi-Def Youtube content), an extended feature domain was defined with the capacity to cover any type of item that the system would need to process. This extended feature domain applied to any type of resources, allows the system to deal with a large variety of items and make cross-content recommendations.

2. **Other types of resources**
   One of the limitations pointed to the previously existing system, regarded the fact that it only addressed TV content, neglecting other types of resources and services the guest had access to and thus preventing the possibility of establishing relationships across them to increase the performance and enrich the type of recommendations. To solve this, the system was enriched with content from video-on-demand and hotel services, establishing relationships across them using the features domain defined previously.

3. **User profiling**
   To solve the user profiling problems that were already detailed in section 3.4.1, the user profiling technique was changed to work better in this domain and to try to solve the *New User problem* (these changes are explained in more detail in section 4.3). Additionally, instead of creating two profiles, the "Accepted Profile" and the "Rejected Profile", it was decided to use only one profile containing all the domain features with an associated weight in the range $[0;1]$.

4. **Recommendation method**
   Although there are different techniques described in the literature to implement the CF recommendation approach, this dissertation proposes a novel CF approach based on k-Nearest-Neighbour (kNN). Instead of recommending exactly the same items that similar users (neighbours) to the target user had consumed before, it creates a new "Predicted Profile" for the target user based on his neighbours' profiles, subsequently applying the already adopted content-based method (Naive Bayes). It should be observed that in our system, user profiles do not contain ratings for the specific items consumed by the users. Instead they contain features and associated weights, describing in this way the users' preferences. This proposed method uses an improved Pearson Correlation metric for the measurement of user similarities, to work better in this domain. Furthermore, this proposed approach that generates a collaborative "Predicted Profile" with features from a well defined structure and set of tags, also tries to overcome the *Sparsity* problem, since there isn't a user-item rating matrix used in recommendation generation.

5. **Recommendation domain**
   The developed recommendation engine can be considered as an enhanced version of the previously implemented system. Not only it extends the scope of recommendations from the TV domain to include also other types of resources, such as VoD or hotel services, as it incorporates in the process a CF recommendation approach to improve the results of the CB approach. Moreover, it presents an architecture flexible enough to allow the operator

to decide: 1) which domains the recommendation process should apply to (by including or excluding domains out of the complete universe of candidate content - the applied design pattern enables to easily fetch new types of candidate content using apposite Web services) and; 2) whether the system should use the combination of the two recommendation techniques (CF and CB) or just one of them.

The implementation will be described in more detail in the next chapter 4.

Chapter 4

Implementation

In this chapter the main implementation decisions will be described. Whenever it's necessary, all the studied methods are shown, as well as the reasons that leaded to the chosen one.

Section 4.1 shows the architecture of the system to give an overview about all the interaction and overall behaviour. Here are shown the class models and database models.

In section 4.2 it is shown how the information processing is made, to understand the decisions on the approach. In this section is also showed how the features domain works and the changes to it. The used data sources (Web services, etc.) are also shown in this chapter.

To solve the user profiling problems that were already detailed in the section 3.4.1, the user profiling was changed. Section 4.3 presents all these changes, for instance the use of a single profile and the method change for calculation and update.

Section 4.4 is one of the most important sections of this chapter. Here is shown the collaborative-filtering technique applied, the user similarity method and the neighbourhood selection, the collaborative profile prediction and the recommendation based in this profile. The implemented recommendation method, besides of improving the recommendations in the TV domain (previously implemented), now is also able of generating recommendations for other resources like VoD and hotel services. The hybridization technique applied is also explained in this section.

Finally, section 4.5 shows other work done in the system, generally improvements that were made besides the goals within the scope of this dissertation.

4.1 Architecture

Here is shown the system architecture. To make everything more readable and understandable, the diagrams are shown in parts and only the main classes are presented. Subsection 4.1.1 gives an overview about all the interaction and overall behaviour. Subsection 4.1.2 explains the class model and subsection 4.1.3 presents the database model.

4.1.1 Overview

First of all, it's important to describe the implemented system from a technical point of view. This system was implemented using C++, a middle-level language, as it comprises a combination of both high-level and low-level language features. C++ is one of the most popular programming languages and is implemented on a wide variety of hardware and operating system platforms [Sch98]. So, several classes were built, using the object-oriented features that C++ provides.

In addition of using an object-oriented design, this system also uses a service-oriented architecture (SOA). SOA is a set of principles and methodologies for designing and developing software in the form of interoperable services, building software components that can be reused for different purposes [Erl05]. Another advantage of this architecture is that clients can access to SOA services using a well-defined interface, usually provided in XML (Extensible Markup Language) or JSON (JavaScript Object Notation). In this system, services are provided in XML. This will be described in further detail in this section.

Figure 4.1 illustrates an overview of the system architecture.

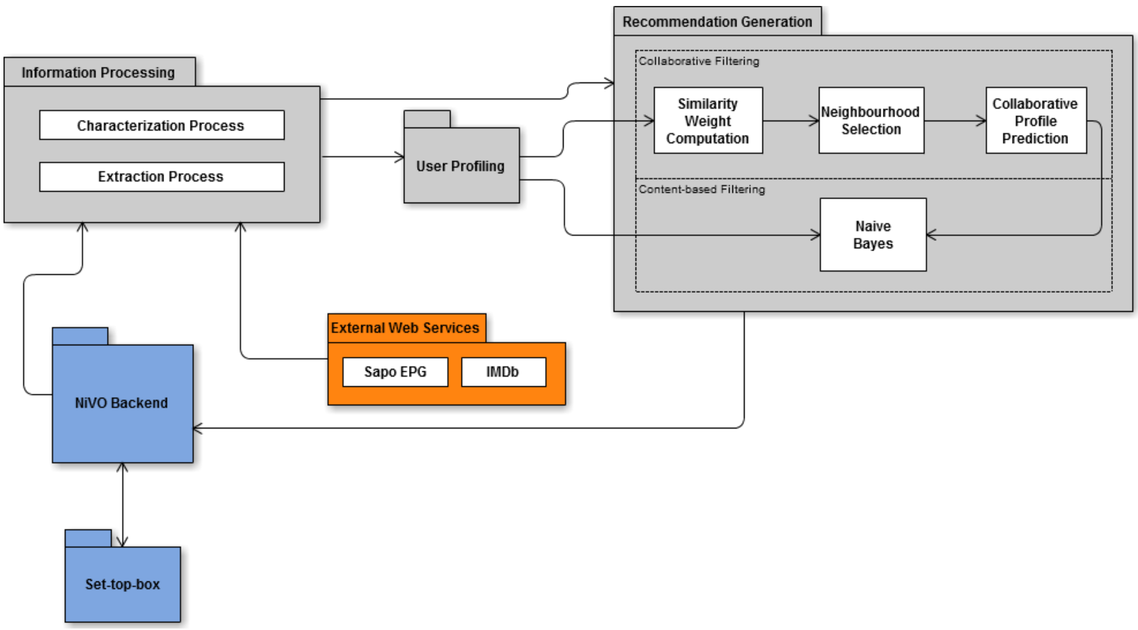

Figure 4.1: Architecture Overview

All modules are grouped by colours according to their roles. On the bottom left of the figure, blue modules represent the NiVO system[1] backend and set-top-box. The set-top-box provides all available services to the end user (e.g. watch TV, buy hotel services, etc.). It also keeps track of all his interactions, that are sent to backend. This module is responsible for gathering interaction data from all the set-top-boxes, and also to manage all data that are available to the users.

[1]NiVO is a hospitality system. For more information about it, see http://www.noniussoftware.com/pdfs/nivo_br_en.pdf

On the top, are presented the main modules of the recommender engine. The backend module provides all data to "Information Processing" module. This module has two main functionalities: "Extraction Process" and "Characterization Process". "Extraction Process" is responsible for request all data from backend and parse it so that it can be analysed by the recommender. This information is about *users*, *available channels*, *VoD*, *services* and *user interaction*; it is provided in the XML format, and then parsed by this class and inserted in the correspondent containers within the recommender.

"Characterization Process" is responsible for, as the name suggests, characterize the content previously obtained by the extraction process. This process associates features from the recommender feature domain to each item (e.g, tv program or hotel service). Usually, Web services provide enough metadata to characterize the items. When these data is unavailable or is not enough, the characterization process can also try to extract features for the items using the items' description: this often occurs in TV content. The adopted approach to do this is "bag of words" [Wal06], that uses a dictionary that associates words or expressions to features in the domain.

When it is not possible to characterize the content by neither of these methods, the characterization process can get additional information from external Web services. These are represented by the orange module "External Web Services". These Web services are Sapo EPG, that is used to get the daily TV schedule for the available channels, and also IMDb API, that is used to get additional information for TV content, especially movies and TV series. It's important to say that this information is always kept updated in a local database, so, if the system fails for any reason, at any point all data can be restored from this database without the need of read and characterize everything again.

The next step is to provide the user interaction data obtained during the extraction process to "User Profiling" module. This module is responsible to create and update the users' profiles in the system. Then, these profiles are analysed by the "Recommendation Generation" module, as well as the candidate items to be recommended, obtained in "Information Processing" module. Content-based filtering approach uses Naive Bayes algorithm to compare these profiles with the candidate items, to classify the items with a *recommendation weight*, to decide if they can be recommended or not. This algorithm compares the features contained in the profiles with the features associated with candidate items (obtained in the characterization process), trying to find similarities between them.

On the other hand, collaborative filtering approach uses these profiles in a different way: the first step is to make the similarity weight computation, which basically is a degree of similarity between each user in the system. Next, for each user, is made a neighbourhood selection, where the most similar users to the target user are selected as "neighbours". Then, using these neighbours, is made the collaborative profile prediction, where for each user is generated a "collaborative profile", which is a profile that contains features that are present in his neighbours' profiles. The final step of CF approach is to feed these profiles to the Naive Bayes algorithm, applying the same algorithm used before in the "users' profiles", but now to "collaborative profiles".

Finally, after the hybridization step, the full list of recommendations is returned to the backend.

4.1.2 Class Model

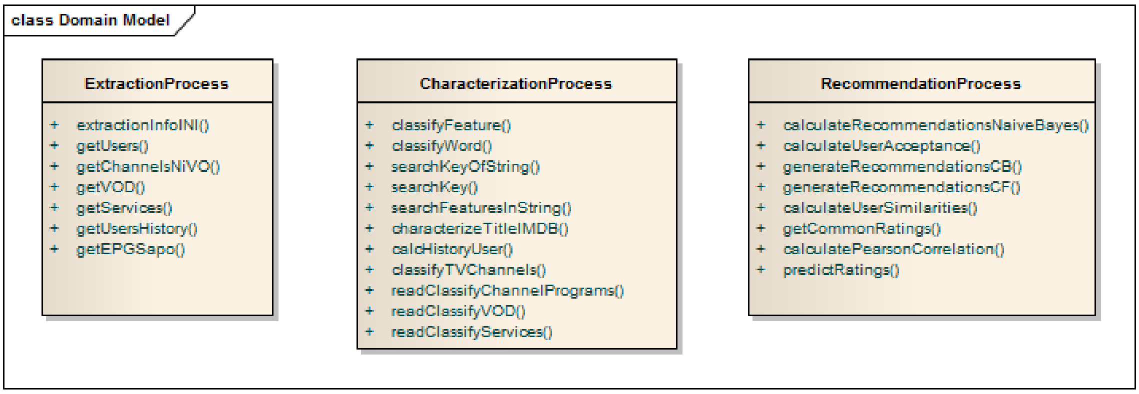

Figure 4.2: Class Model - Main classes

Figure 4.2 shows the three main classes of the system. The class *ExtractionProcess* is responsible for all the data gathering operations. Some of its most important methods are listed. The method *extractionInfoINI()* gets data from the local database, updating the engine with all data read before from the NiVO backend. This method is executed only once when the recommender engine starts running, in order to gather data from users, including profiles and interaction, TV channels, previously characterized items and also data to feed the dictionary for the classification process.

The methods *getUsers()*, *getChannelsNiVO()*, *getVOD()*, *getServices()* and *getUsersHistory()* communicate with the NiVO backend, allowing to retrieve data internally available in the system, They enable to obtain respectively: 1) data related to the users; 2) TV channels available to the hotel guests; 3) the VoD catalogue available in the hotel; 4) the additional services offered by the hotel to its guests; and 5) the users' interaction with the set-top-boxes.

The method *getEPGSapo()* implements functionality to retrieve data external to the system. This method, as well as the method *characterizeTitleIMDB()* in the class *CharacterizationProcess*, use the Web services listed in module "External Web Services" on figure 4.1. *getVOD()*, *getServices()* and *getEPGSapo()* collectively allow the system to obtain data about the list of candidate content, essential for the global operation of the recommendation engine. The process of gathering data concerning the Electronic Programming Guide (EPG) (accessing an online service made available by Sapo, the Sapo EPG API) needs to be performed daily, since the EPG only provides information concerning the TV programming for one day.

The class *CharacterizationProcess* is responsible for analysing the candidate content obtained by the previously mentioned methods and for characterizing it. This is, this class analyses the metadata that accompanies the candidate content and obtains a set of tags that allow to identify the type of content. As explained before, this characterization process is performed adopting an approach "bag of words" [Wal06]. When there is not sufficient metadata to fully characterize the content, as it often occurs with the TV programming content obtained from the EPG, the system uses the internal dictionary or accesses external sources, such as the IMDb API, to obtain additional tags.

Video-on-demand and hotel services resources usually came with metadata within the features domain in the system. With this kind of content, the characterization process only needs to parse the data from Web services and associate it with the content, organizing it in the system.

The class *RecommendationProcess* is responsible for the recommendation process. This class uses all the data pre-processed previously by the other classes and generates recommendation for each user. This process will be explained in further detail in section 4.4.

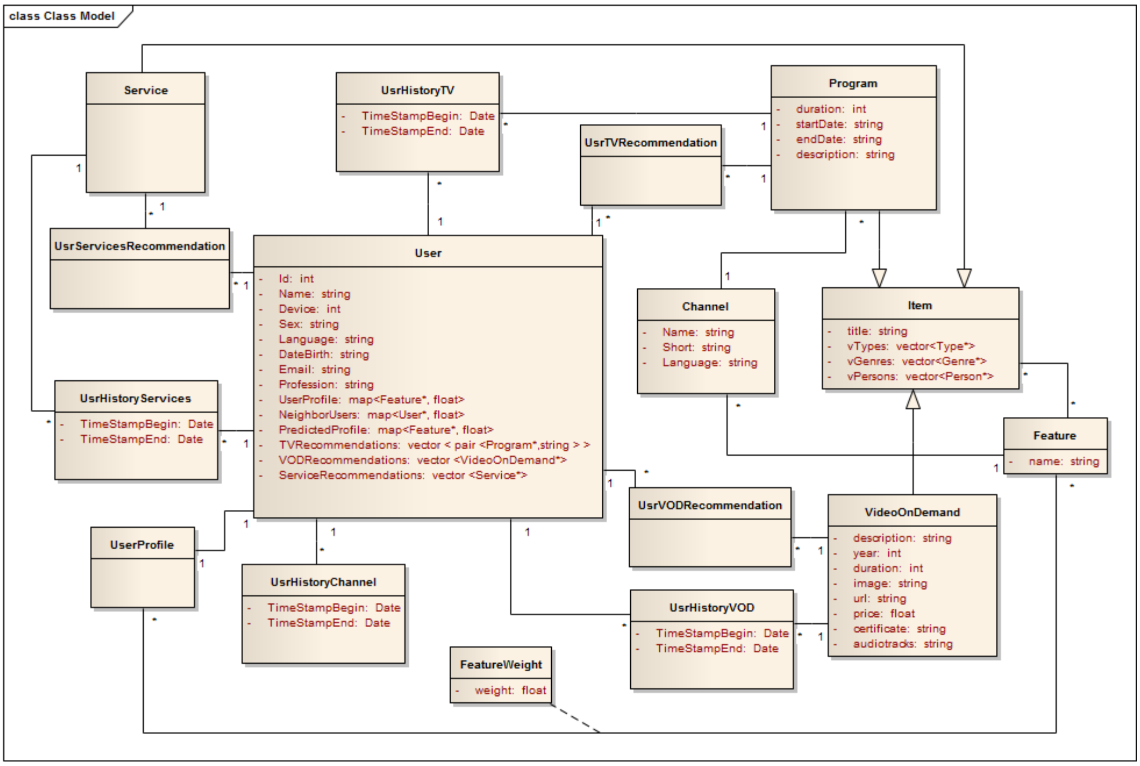

Figure 4.3: Class Model - User and Resources

Figure 4.3 shows the user and his resources. As is possible to see, in the centre of the image there is the user class with all his attributes. Besides this, the user has associated with him *Recommendations* and a *History* for each kind of resource (TV, VoD and services). This history represents his interaction parsed from the previous steps.

Each user has also a "User Profile". This profile is a set of features, with an associated weight to each one. In this case, the used container is a *map<Feature*,float>*, that associates to each feature the correspondent weight. This is a very important detail, since initially the user profile was stored in a *vector< pair<Feature*, float> >*, which means that for checking if a feature was in the user's profile, it was necessary to search in all the elements of this vector. With this change became possible to find a feature with only one operation (using *map.find()*).

It's important to note that all resources (*Program*, *VoD* and *Service*) are all subclasses of a base class *Item*. Initially, the system only had the class *Program* to represent the candidate content. But, with the necessity of incorporating other types of resources in the system, it was necessary to create this base class, that contains the common attributes to each item: *title*, that identifies the object, and the vectors of features (*Type*, *Genre* and *Person*), that are resultant from characterization process already explained previously. In this way, when there is the need to add a new resource to the system, it is only necessary to make it a subclass of *Item* and implement the necessary methods.

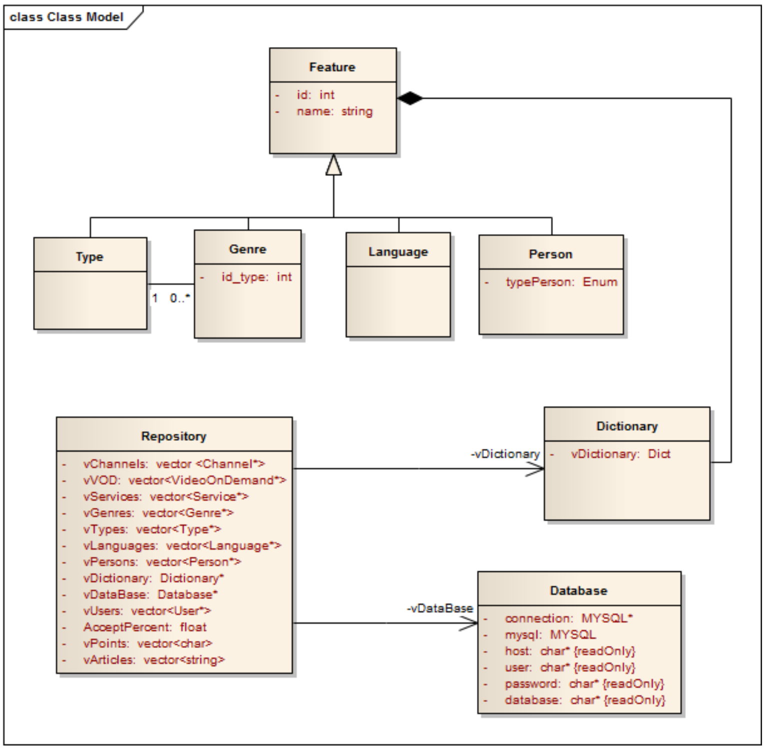

Figure 4.4: Class Model

On the bottom left corner of figure 4.4 it is possible to see the class *Repository*. This class is responsible to store all the information that is available in the recommender engine, that was obtained from extraction process or from the local database.

This class is connected to the class *Database*, that is responsible to manage all connections to read and write data to DB. The class *Repository* is available to *ExtractionProcess* and *CharacterizationProcess*, which allows to the data that is obtained from the backend and characterized to be sent to local database.

This repository has also the dictionary, where all the data that is used in the "bag of words" method is stored.

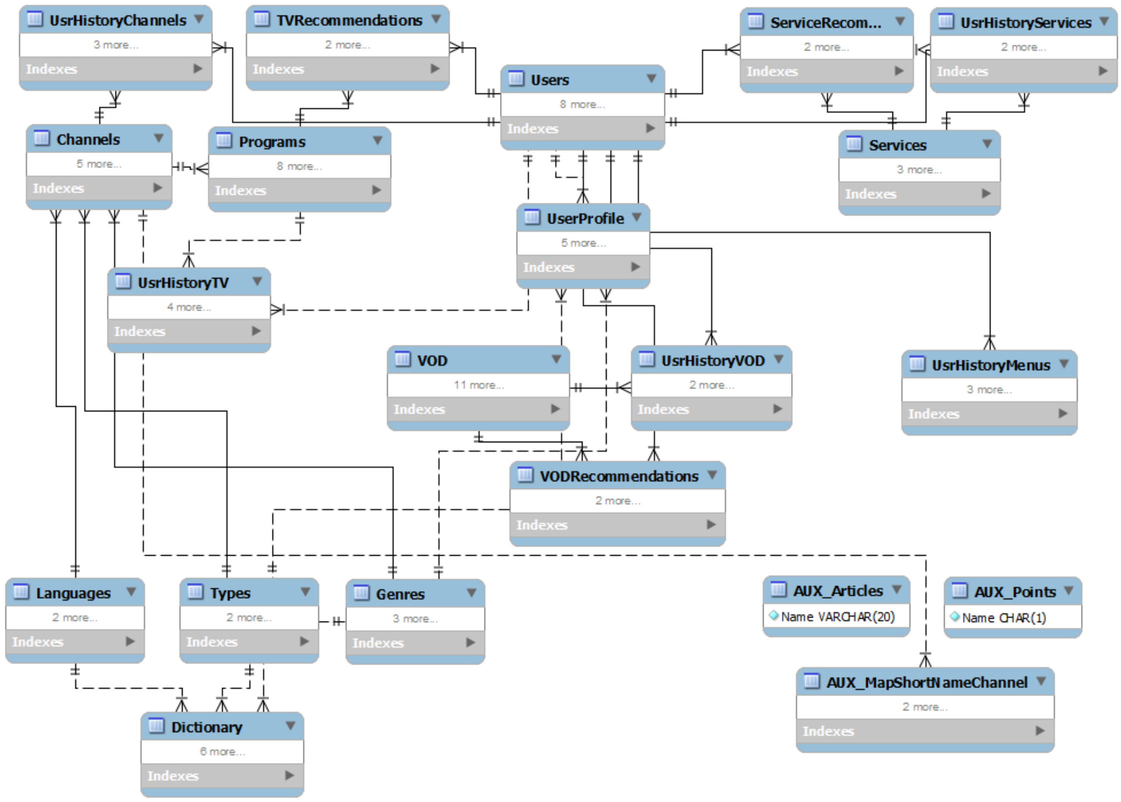

Figure 4.5: Database Structure

4.1.3 Data Model

Figure 4.5 shows the global database structure with the corresponding relationships. This model was designed to store all the information that was already mentioned before in the previous section. Is important to mention that, for example, compared with the model from the earlier developed system (previous to this dissertation), the "User Profile" is now only in one table (*UserProfile*) instead of being in two (*AcceptedProfile* and *RejectedProfile*).

Besides that, now there are tables to store all the information related do video-on-demand and services (VOD, *UsrHistoryVOD*, *VODRecommendations*, Services, *UsrHistoryServices* and *ServiceRecommendations*).

All the tables shown in this model can easily be associated with the classes presented in the previous section (Class Model). They are all used to store processed information by the "Information Processing" module. The only tables that are not related with any of those classes are *AUX_Articles*, *AUX_Points*, and *AUX_MapShortNameChannel*. The first two are auxiliary tables that contain information to be used in the characterization process, more specifically in "bag of words" approach. *AUX_MapShortNameChannel* is an auxiliary table to *Channels* that contain some short names associated with channels, that are used to access Sapo EPG API.

4.2 Information Processing

This section gives an overview about the information processing. This is very important to understand better the data sources and how the information is processed to extract the key features that are used in recommendation.

Here are also described the changes that were made to the feature domain, as well as the processing of behavioural data, that is used in the user profiling.

4.2.1 Data Sources

The developed recommender system uses two different kinds of data sources. On one hand we have the internal Web services that usually provide new available data to the engine, that are located in the backend. On the other hand we have external data that is used to add new information to the data provided by the internal Web services. Hereupon, we have the following data sources:

Internal Web services:

- **TV Channels**
  Provides information about all the available TV channels in the system. This is important to know from which channels to get the programme schedule from the EPG.

- **Users**
  Provides the user list. If a new user is inserted into the system, it will be also added to the recommender engine to create his profile based on his interaction and start generating recommendations.

- **Video-on-demand**
  Provides information about all available VoD in the hotel catalogue, including important metadata to characterize the content.

- **Services**
  Provides information about all available hotel services, including important metadata to characterize the content.

- **User Interaction**
  Provides information about all user interaction. This Web service contains information about the periods of time that the user is viewing a channel and consumed VoD and services.

External providers:

- **Sapo Electronic Program Guide**
  This external provider gives information about which programmes are passing on each channel and the corresponding timestamps.

- **IMDb API**
  The Internet Movie Database provides information about movies, e.g. genre, duration and actors, and also TV series.

Listings 4.1, 4.2, 4.3, 4.4, 4.5, 4.6 and 4.7 present some examples of information extracted from these Web services.


```xml
<?xml version="1.0" encoding="utf-8"?>
<TV>
  <class name= 'STB100'>
    <playlist lang="default">
      <channel number="1">
        <name>Eurosport</name>
        <logo>eurosport.png</logo>
        <url>udp://@239.192.1.36:1234</url>
        <nationality>DE</nationality>
        <description>Eurosport - Desporto Alemao</description>
      </channel>
    </playlist>
  </class>
</TV>
```


Listing 4.1: Web Service Example - TV Channels


```xml
<?xml version="1.0" encoding="utf-8"?>
<Users>
  <User>
    <Name>Patricia Almeida</Name>
    <Identification>4</Identification>
    <Language>PT</Language>
    <DateBirth>1989-03-12</DateBirth>
    <Country>Portugal</Country>
    <Timezone>GMT +0000</Timezone>
    <Email>patricia.almeida@gmail.com</Email>
    <Profession>Estudante</Profession>
    <Sex>M</Sex>
  </User>
</Users>
```


Listing 4.2: Web Service Example - Users


```xml
<?xml version="1.0" encoding="UTF-8"?>
<vod>
  <genre name="Action">
    <metadata id="129245">
```

```
      <name>Death Race</name>
      <year>2008</year>
      <image>/minimyth/playlists/vod/deathrace-large.jpg</image>
      <duration>105</duration>
      <description>Ex-con Jensen Ames is forced by the warden of a notorious prison
           to compete in our post-industrial world\'s most popular   sport: a car
          race in which inmates must brutalize and kill one another on the road to
          victory.</description>
      <price>7,00</price>
      <url>rtsp://10.0.30.142:554/NiVo/deathrace_en_fr_de_it_es_ts.mpg</url>
      <cast>Paul Anderson (Director), Jason Statham, Joan Allen, Ian McShane,
          Tyrese Gibson</cast>
      <certificate>15</certificate>
      <audiotracks>es, it, de, fr, en</audiotracks>
    </metadata>
  </genre>
</vod>
```

Listing 4.3: Web Service Example - VoD

```
<?xml version="1.0" encoding="UTF-8"?>
<services>
  <category name="Cycling">
    <metadata>
      <id>13572</id>
      <name>Aluguer de bicicleta</name>
      <price>12,0</price>
    </metadata>
  </category>
</services>
```

Listing 4.4: Web Service Example - Service

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Interaction SYSTEM "userInteraction.dtd">
<Interaction>
  <User>
    <Name>Luis Maricio</Name>
    <DateBirth>15-11-1980</DateBirth>
    <Sex>M</Sex>
    <Language>PT</Language>
    <Country>Portugal</Country>
    <Timezone>GMT +0000</Timezone>
    <Identification>1115544</Identification>
    <Actions>
      <MENU>
```

```
      <TimeStampBegin>2010-12-11 10:54:36</TimeStampBegin>
      <TimeStampEnd>2010-12-11 10:55:36</TimeStampEnd>
      <Equipment>STB100</Equipment>
      <Name>Menu Principal</Name>
      <Menudepth>1</Menudepth>
    </MENU>
    <TV>
      <Name>RTP1</Name>
      <TimeStampBegin>2010-12-11 10:57:36</TimeStampBegin>
      <TimeStampEnd>2010-12-11 16:56:36</TimeStampEnd>
      <Equipment>STB100</Equipment>
      <Country>EN</Country>
      <Menudepth>3</Menudepth>
    </TV>
  </Actions>
 </User>
</Interaction>
```

Listing 4.5: Web Service Example - User Interaction

```
<GetProgramListByChannelDateIntervalResponse xmlns="http://services.sapo.pt/
    Metadata/EPG">
  <GetProgramListByChannelDateIntervalResult>
    <Program>
      <Id>4344795</Id>
      <Title>Britcom</Title>
      <Description>Inclui: Dois Mil E Doze e Absolutamente Fabulosas.</Description>
      <StartTime>2012-06-17 23:28:00</StartTime>
      <Duration>3840</Duration>
      <ChannelName>RTP 2</ChannelName>
      <ChannelSigla>RTP2</ChannelSigla>
      <EndTime>2012-06-18 00:32:00</EndTime>
      <EpisodeTitle/>
      <ShortDescription>Britcom</ShortDescription>
    </Program>
  </GetProgramListByChannelDateIntervalResult>
</GetProgramListByChannelDateIntervalResponse>
```

Listing 4.6: External Provider Example - Sapo EPG

```
<root response="True">
  <movie title="Fight Club" year="1999" rated="R" released="15 Oct 1999" runtime="2
      h 19 min" genre="Drama" director="David Fincher" writer="Chuck Palahniuk,
     Jim Uhls" actors="Brad Pitt, Edward Norton, Helena Bonham Carter, Meat Loaf"
     plot="An insomniac office worker and a devil-may-care soap maker form an
     underground fight club that transforms into a violent revolution." poster="
```

```
    http://ia.media-imdb.com/images/M/
    MV5BMTc1ODg1NTMxNV5BMl5BanBnXkFtZTYwOTg4Njk4._V1_SX640.jpg" imdbRating="8.8"
    imdbVotes="573,043" imdbID="tt0137523"/>
</root>
```

Listing 4.7: External Provider Example - IMDb API

4.2.2 Characterization Process

The characterization process is a very important step in this recommender system. The main idea of this process is to associate a set of features from the domain to the provided resources. This allows the system to read the data and deal with it, constructing user profiles based on that information and generating recommendations.

This characterization process is applied to 3 types of content in the system: TV programmes, VoD and hotel services. The characterization process for the TV programmes was already implemented in the first version of the system. The selected method was to apply a “bag of words” approach [Wal06] to the program descriptions, since that the provided program metadata from Sapo EPG does not have enough information to characterize the programmes. When the programmes are movies or series, additional information can also be obtained using an external provider (e.g. IMDb API).

To characterize video-on-demand and services from the hotel the process is much easier. As it can be seen in listings 4.3 and 4.4, both VoD and services already come organized by feature from Web services. So, during the characterization process, it’s only necessary to parse the XML and associate the features to the content.

When the characterization process is finished, every items have associated features from the features domain. This will be explained in the next section 4.2.3.

4.2.3 Features Domain

In figure 4.6 it is shown the features organization. The system has two types of features: Types and Genres. Each one is only characterized by its name and an associated ID. Besides that, a Genre can be associated (or not) with a Type so that an hierarchy (with the maximum of two levels) can be created.

Figure 4.7 presents the Types. In figure 4.8 are shown some Genres that have as parent the type "Sport". There are other Genres in the system that are not connected with any Type in specific. For example, the Genre "Crime" is not associated with any type, because it can be present in items of different types, e.g. films or series.

Blanco-Fernández et al. designed a system presented in [BFLNPA+11] with the main goal of offering tourism recommendations based on the TV viewing history. They managed to do this defining an ontology where they associated key features of TV Contents to features of Tourist Resources.

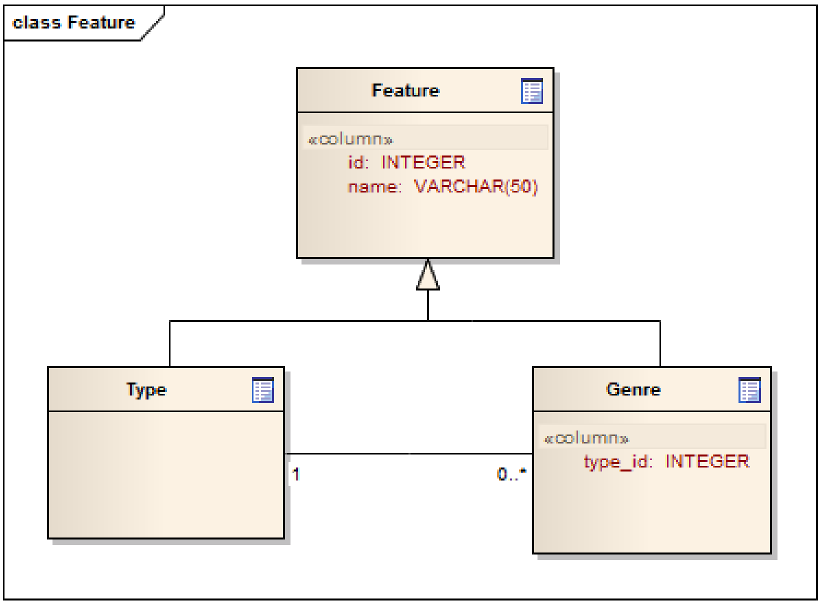

Figure 4.6: Features Organization

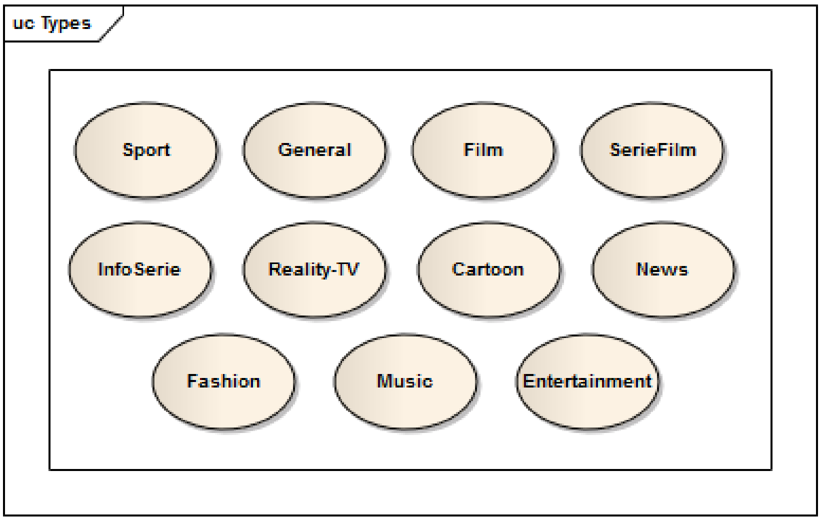

Figure 4.7: Types

After the evaluation of all the possible kinds of data available in the system, it was decided that it was possible to create a compact, yet complete and flexible, structure and set of tags, that is able to cover all the items that the system may have. To understand better the advantages of having only one domain for all the features, an example is shown on table 4.1.

Is possible to see from the user interaction in the example on table 4.1 that the user likes Comedies, Football and Tennis. Table 4.2 shows some items that can be evaluated for recommendation to the user. Although the user has watched a tennis program on the television and did not actually played, we can infer that the user likes tennis. As this feature is going to be added to the

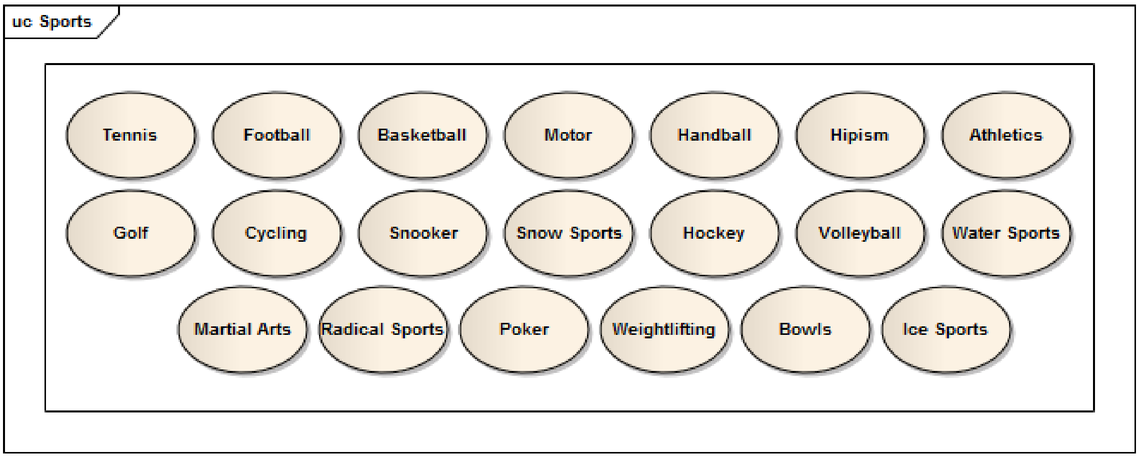

Figure 4.8: Genres Associated with the Type Sports

Table 4.1: User Interaction History Example

Item	Feature list	Acceptance ratio (%)
[TV Program] Euro 2012 - Denmark vs. Portugal	Football	100%
[VoD] American Reunion	Comedy	100%
[TV Program] Queens - AEGON Championships	Tennis	98%

Table 4.2: Content for Recommendation

Item	Feature list
[TV Program] The Big Bang Theory S03E10	Comedy
[Service] Rent a tennis court	Tennis

user's profile, the service "Rent a tennis court" can be recommended to the user, as it has a feature "Tennis" associated with it. The same happens with the TV program "The Big Bang Theory S03E10" because it is a "Comedy", and the user has requested a video-on-demand that was also a "Comedy".

Adding features that cover all the domain that can be present in the system, it's possible to add different resources to the system and make cross-content recommendations since all data shares the same features.

With this, it is also possible to respect the non-functional requirement NFRS01 that says to make the minimum changes as possible in the previously developed system. With this, it's possible to keep the current characterization method for TV programmes and the same feature structure, only enriching the available data. NFRS03 is also respected with this approach, since at any time is possible to add a new data source to the system, and to do that is only necessary to review the features domain.

4.2.4 Behavioural Data

As was shown before in figure 4.1, the set-top-box (STB) keeps a record of the users' interaction with the system, that later is sent to the recommender engine. This user interaction can also be called behavioural data, since it pictures the users' behaviour. The processing of this information is used to make the user profiling that will be explained in the next section 4.3.

In listing 4.8 it's shown a typical example of a user interaction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Interaction SYSTEM "userInteraction.dtd">
<Interaction>
  <User>
    <Name>Luis Maricio</Name>
    <DateBirth>15-11-1980</DateBirth>
    <Sex>M</Sex>
    <Language>PT</Language>
    <Country>Portugal</Country>
    <Timezone>GMT +0000</Timezone>
    <Identification>1115544</Identification>
    <Actions>
      <TV>
        <Name>RTP1</Name>
        <TimeStampBegin>2010-12-11 10:57:36</TimeStampBegin>
        <TimeStampEnd>2010-12-11 16:56:36</TimeStampEnd>
        <Equipment>STB100</Equipment>
        <Country>EN</Country>
        <Menudepth>3</Menudepth>
      </TV>
      <TV>
        <Name>SIC</Name>
        <TimeStampBegin>2010-12-11 16:56:37</TimeStampBegin>
        <TimeStampEnd>2010-12-11 17:24:12</TimeStampEnd>
        <Equipment>STB100</Equipment>
        <Country>EN</Country>
        <Menudepth>3</Menudepth>
      </TV>
      <VOD>
        <Id>26833795</Id>
        <TimeStampBegin>2010-12-11 10:03:51</TimeStampBegin>
      </VOD>
      <Service>
        <Id>17905436</Id>
        <TimeStampBegin>2010-12-11 07:59:47</TimeStampBegin>
      </Service>
    </Actions>
  </User>
</Interaction>
```

Listing 4.8: User Interaction Example

The parsing of this user interaction is very important to understand the users' tastes. The most difficult part of this information processing is related to the TV interaction. To treat this information, it's necessary to follow some steps: the first step is, for each viewing period in a channel, get the programmes during that time. Then, for each program is necessary to calculate the ratio of user's real watching time to the program's total duration, since this ration will have an impact during the user profiling.

This ratio corresponds to the user appreciation for a certain item. When a user action is associated to the purchase of a VoD or a service, this ratio will be equal to 100%.

4.3 User Profiling

To solve the user profiling problems that were already detailed in section 3.4.1 the following approach was followed.

1) If a term already exists in the profile, its weight is modified as the following:

$$w_i' = (1-\alpha) \times w_i + \alpha \times \beta \tag{4.1}$$

$$\beta = \frac{T_r}{T_t} \in [0,1] \tag{4.2}$$

2) If a term does not exist in the profile:

$$w_i = \beta \tag{4.3}$$

Where:

- w_i' if the weight of term t_i after update;
- w_i if the weight of term t_i before update;
- $\alpha \quad (0 \leq \alpha \leq 1)$ is the learning rate that determines how quickly the user profile forgets old preferences and tracks new ones;
- β is the user's rating to an item; in the case of TV programmes, is the ratio of user's real watching time (T_r) to the program's total duration (T_t).

This user profiling technique was used on the creation of a single profile, thus solving one of the main problems detected with the previous approach. This profile is composed by the features associated with each resource consumed by the user. Each feature is then associated to a weight within the limits $[0,1]$, i.e. if a feature has a weight equal to 0 the user completely rejects it, and if is 1 completely accepts it.

Besides this, is possible to see that equations (3.1) (w_i') and (3.5) (w_i) suffered some modifications. Analysing the equation (4.1) it's possible to see that the factor $f(i)$ was removed, since that as stated before, in this domain there is no need to give a different weight to a feature depending in its order on the feature list. On the same manner, $f(i)$ was also removed from w_i calculation.

Its possible to see another change on equation (4.3). The learning rate α was removed from the weight calculation when a feature is new to a user profile. The feature weight will only be influenced by this factor during the update process. This decision was made to try to avoid the "New User Problem" (see section 2.3.4) and start generating recommendations for new users in the system as soon as possible. This is a very important aspect in the domain of hospitality services, since a significant portion of users can use the system very sporadically, so the system needs to respond quickly to their needs.

4.4 Recommendation Method

Collaborative recommender systems depend on overlap of ratings across users. This brings a difficulty when the space of ratings is sparse: few users have rated the same items [Bur02]. A big variety of systems with different goals that use CF methods try to find similarity between users analysing common items rated by both users [RIS$^+$94, HKTR04, Zan08, RT08, ZJ09b]. In these techniques if user A is similar to user B, the list of items recommended to user A will include some items that user B has rated and user A didn't.

The proposed CF approach is based on k-Nearest-Neighbour (kNN). As explained before in 2.4, this kind of approaches despite being relatively simple to implement compared with other approached, provide very good accuracy results. In the 2008 solution to the Netflix Prize, [BKV08] makes an extensive comparison between the results of different methods using the root mean squared error (RMSE) metric. Their final solution provided a RMSE=0.8643 doing a linear blend of over 100 results of different approaches. On the other hand, their neighbourhood-based model could provide a RMSE=0.8953, that despite of not being so accurate as their final solution, is also a very good result, especially if we take in mind that their final solution uses more than 100 approaches. So, based on their tests, it's possible to conclude that kNN approaches can also provide very good accuracy results.

Another important aspect of this kind of approaches is that they can adapt to rapid changes in the user-ratings matrix. this is a very big advantage of this approaches, especially when they are applied to systems that change the number of users or items very quickly. In a system used by hotel guests like the one proposed in this dissertation, it's very important that the system can rapidly adapt to this kind of changes, starting to generate recommendations of new items and to new users as soon as possible. So, this is important aspect that lead to the choice of kNN approach to be applied in this system.

The fundamental algorithm of this recommendation method can be divided into three steps: 1) measurement of similarities between the target and the remaining users in the database, using an improved Pearson Correlation method; 2) selection of the most similar users who will serve as recommender, applying a top-N filtering (to chose the N most similar users) and threshold filtering (ignoring neighbours with a similarity degree less than 0); and 3) generate a "Predicted Profile" based on the weighted average of the neighbours' profile features, weighted by their similarity to the target user [LG07].

The adopted approach creates a new profile to the target user based on his neighbours' profiles, the "Predicted Profile", instead of recommending particular items previously consumed by those neighbours. Then, it applies the content-based method to find candidate content similar to the preferences registered in this new "Predicted Profile".

This novel method proposed in this dissertation adopts a different approach to address the sparsity problem (see section 2.3.7), since there isn't a user-item rating matrix used in recommendations generation. This can be considered very important, especially when dealing with a large quantity and diversity of resources. The user profiling technique explained in the previous section that influences this recommendation method also tries to avoid the new user problem 2.3.4.

4.4.1 Rating Prediction

There are two kinds of neighbourhood-based recommendations approaches: user-based and item-based. Item-based approaches (such as [LSY03]) predict the rating user u would assign to item i based on ratings u has given to items similar to i (similar items are items rated by several users in a similar way). On the other hand, user-based approaches (such as [RIS+94]) evaluate the interest of user u for item i using the ratings assigned to this item by other users with similar rating patterns (neighbours) [KRRS11]. This work adopts an approach similar to the latter, with the difference that it does not evaluate ratings for actual items but rather for features.

The approach more suitable to this problem, as stated before, is the user-based. In user-based approaches there are two ways for predicting ratings: using regression or using classification. In [KRRS11] is stated that the choice between implementing a neighbourhood-based (NB) regression or classification method depends on the system's rating scale: if the rating scale is continuous, then a regression method is more appropriate. On the contrary, if the rating scale has only discrete values, then a classification method might be preferable.

Since that with the defined user profiling method we only have continuous values, we'll use the regression technique for rating prediction.

The following equation will be applied:

$$\hat{r}_{ui} = \frac{\sum_{v \in N_i(u)} w_{uv} r_{vi}}{\sum_{v \in N_i(u)} |w_{uv}|} \tag{4.4}$$

Where in the formulation (4.4):

- u target user;
- v neighbour user;
- i item to be compared;
- $N(u)$ nearest-neighbours to u;
- $N_i(u)$ nearest-neighbours to u that have rated i;
- r_{ui} rating of a user u to a new item i;

- w_{uv} similarity between users u and v;

As stated previously, the rating prediction does not aim at obtaining ratings for individual items that could be recommended to the user, e.g. a movie. It rather aims at predicting ratings or weights that would be assigned to features that characterize those items. The user profiles used in this work do not register ratings for individual items consumed by the user. Instead, they register the features that characterize the items that the user has consumed, assigning them weights or ratings. This means that the user profile does not follow the traditional $n \times m$ "user $\times$ item" matrix model, but rather a "user $\times$ feature" matrix.

Accordingly, the rating prediction phase does not produce a set of predicted ratings to items that could be recommended to the target user, as it usually happens in this kind of systems. It rather produces a new "Predicted Profile" with a set of "predicted features" based on the profiles of users similar to the target user.

4.4.2 Rating Normalization

For the rating normalization both formulas presented in chapter 2.5.1 were tested.

Using the mean-centering approach presented in equation (2.7) the user-based prediction of a rating r_{ui} is obtained by using:

$$\hat{r}_{ui} = \bar{r}_u + \frac{\sum_{v \in N_i(u)} w_{uv}(r_{vi} - \bar{r}_v)}{\sum_{v \in N_i(u)} |w_{uv}|} \tag{4.5}$$

On the same way, using z-score (2.8), replacing the normalized rating on the equation to calculate the predicted rating:

$$\hat{r}_{ui} = \bar{r}_u + \sigma_u \frac{\sum_{v \in N_i(u)} w_{uv}(r_{vi} - \bar{r}_v)/\sigma_u}{\sum_{v \in N_i(u)} |w_{uv}|} \tag{4.6}$$

Based on some testing in the rating prediction using both normalization formulas, we reached the conclusion that in our test environment, neither of these normalization methods has any effect in the results.

This is probably because the user profiling is only based in implicit profiling, i.e., the users do not explicitly give their rating, so as the rating is given by the TV viewing time or the purchase (or not) of VoD or a service, there is no relevant variation on the ratings scale. So, in the final solution, no rating normalization method was applied.

4.4.3 Similarity Weight Computation

The similarity weight computation is one of the most critical aspect of building a neighbourhood-based (NB) recommender system, since it has a significant impact on both its accuracy and its performance. This has two important roles on a NB system: 1) allows the selection of trusted neighbours whose ratings can be used in the prediction, and 2) provides the means to give more or less importance to these neighbours in prediction, since the similarity degree between neighbours counts on the rating prediction as seen on 4.4.1

4.4.3.1 Different Metrics

Several similarity measures were studied. One of them often used in information retrieval [Bel92, GR05, WTLH06] to measure the similarity between two objects a and b consists in representing them in the form of two vectors x_a and x_b and computing the Cosine Vector (CV) similarity between these vector:

$$cos(x_a, x_b) = \frac{x_a \top x_b}{\|x_a\| \, \|x_b\|} \tag{4.7}$$

It's possible to apply this measure to the context of item recommendation by considering a user u as a vector $x_u \in \mathbb{R}^{|I|}$, where $x_u = r_{ui}$ (if user u has rated i, otherwise 0). The similarity would be computed as:

$$CV(u,v) = cos(x_a, x_b) = \frac{\sum_{i \in I_{uv}} r_{ui} r_{vi}}{\sqrt{\sum_{i \in I_u} r_{ui}^2 \sum_{j \in I_v} r_{vj}^2}} \tag{4.8}$$

where I_{uv} denotes the items rated by both users. A problem with this measure is that is does not consider the differences in the mean and variance of the ratings made by users u and v. A popular measure that compares ratings where the effects of mean and variance have been removed is the Pearson Correlation (PC) similarity:

$$PC(u,v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - \bar{r_u})(r_{vi} - \bar{r_v})}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \bar{r_u})^2 \sum_{i \in I_{uv}} (r_{vi} - \bar{r_v})^2}} \tag{4.9}$$

Another widely used metric is the *Mean Squared Difference* (MSD) [SM95], which evaluates similarity between two users u and v as the inverse of the average squared difference between the ratings given by both on the same items [KRRS11]:

$$MSD(u,v) = \frac{|I_{uv}|}{\sum_{i \in I_{uv}} (r_{ui} - r_{vi})^2} \tag{4.10}$$

4.4.3.2 Chosen Metric

Several studies compare the use of different metrics in different systems. The metric that has slightly better performance and commonly accepted as one of the best choices is Pearson Correlation [HKR02, ZJ09b, KRRS11].

The proposed metric to the similarity weight computation is the Pearson Correlation with some improvements to produce better results in this domain. The formulation is the following:

$$PC(u,v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - \bar{r_u})(r_{vi} - \bar{r_v})}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \bar{r_u})^2 \sum_{i \in I_{uv}} (r_{vi} - \bar{r_v})^2}} \times \frac{N_{r_{uv}}}{N_{r_u} + N_{r_v} - N_{r_{uv}}} \tag{4.11}$$

Where in equation (4.11):

- N_{r_u} is the number of rated items by the user u;
- N_{r_v} is the number of rated items by the user v;
- $N_{r_{uv}}$ is the number of common ratings by the users u and v;

This factor was added because, for example, when a new user only had 1 feature in the profile, if he was compared with another user that had the same rating to the same feature, their similarity degree will be 100%, despite the users could be very different. By adding this factor, the number of common ratings now if a significant aspect in similarity degree calculation.

4.4.4 Neighbourhood Selection

When designing an hybrid recommender system with a collaborative approach based on the k-Nearest-Neighbour method , the number of nearest-neighbours to select and the criteria used for this selection can also have a serious impact on the quality of recommendations. This selection is usually done in two steps: 1) a global filtering step is applied to keep only the most likely candidates, and 2) choose the best candidates before the prediction [KRRS11].

This is very important because in systems that can have thousands or even millions of users and items, usually it's not possible to store all the similarities between all users and/or items due to memory limitations. This filtering step is very important to reduce the amount of data to store and at the same time to limit the number of candidate neighbours to consider in predictions.

There are three main ways to apply this filtering:

- **Top-N filtering**
  For each user or item only the best N neighbours are kept, as well as the corresponding similarity weight. N should be chosen carefully to avoid problems with efficiency or accuracy. For example, if N is too large, an excessive amount of memory will be required and the rating prediction process will be slow. On the other hand, if it is too small it can reduce the coverage of the recommendation and which may lead to some items never be recommended.

- **Threshold filtering**
  Instead of choosing a fixed number of NN, this filtering technique keeps all neighbours whose similarity degree is greater than a minimum given threshold (w_{min}). The drawback of this approach is that the right value for w_{min} may be hard to find, i.e., defining only a minimum threshold value can lead to the selection of too many or too few neighbours.

- **Negative filtering**
  In general, negative rating correlations are less reliable than positive ones. Intuitively, this is because strong positive correlation between two users is a good indicator of their belonging to a common group [KRRS11]. However, although negative correlation may indicate membership to different groups, it does not tell how different these groups are, or whether

these groups are compatible for other categories of items. So, this filtering type maybe can be good for some systems, although there are no conclusive studies that can prove their efficiency.

Instead of adopt a single neighbourhood selection technique and be limited to a single approach, it was decided to use both threshold filtering and Top-N filtering. The threshold filtering was implemented with $w_{min} = 0$ in order to ignore the neighbours with a similarity degree less or equal to 0. These neighbours are from different groups or do not have any kind of relation, so there is no need to spend space in memory storing this data.

The top-N filtering was applied to select, at the maximum, the N closest neighbours to the user. As the system testing was made with only 60 users, the N value was set to 10, as it will be explained in chapter 5.

4.4.5 Hybridization Method

As it was already explained before and illustrated in figure 4.1, the method to combine both approaches is the following:

1. Content-based filtering uses the target User Profile to feed Naive Bayes algorithm, generating a recommendation list sorted by descending order of relevance;
2. Collaborative filtering technique generates a collaborative profile (Predicted Profile) based on neighbour users' profiles with similar tastes;
3. Collaborative profile is used to fed the Naive Bayes algorithm, that analyses candidate items content and generates another recommendation list sorted by descending order of relevance;
4. Both recommendations can then be presented separately.

These hybridization methods brings together two different types of hybridization designs: in one hand, it works like a mixed parallelized hybridization design, since the top-scoring items from each recommender are then displayed to the user next to each other, as in [BHY97]. On the other hand, it works like a feature combination monolithic hybridization design, since in this kind of systems, a single recommendation algorithm is provided with features from different recommendation data sources.

4.5 System Improvements

Besides the improvements made within the goals of this dissertation, some other changes were also made to the system:

- Behavioural data parsing corrected, to correct some problems that caused the watching ratio for TV programmes being calculated incorrectly; some β values did not correspond to the actual watching ratio for the programmes, which was translated in wrong weight values for some features during the user profiling.

- Initially the system was collecting data from the Sapo EPG corresponding to only 6 channels. When the volume of information increased and the system started to collect daily data from 107 channels, new problems arose. This was solved using more appropriate data structures. Some problems were also corrected in the characterization process, since the system was not prepared to deal with some error situations when parsing data from Sapo EPG.
- The original channel characterization was only prepared to associate channels with Types (e.g. Music, Sports). When new channels were added to the system, it became necessary to characterize channels with Genres also (e.g. The Poker Channel should be associated with the Genre "Poker" and not with the Type "Sports"). So, this was also improved to allow better content characterization.
- Robustness of the system improved: the system runs a thread once a day to get data from external providers. In some situations when the internet connection was slow or the services were unavailable, the system simply could not get the information for that day or just stopped working. These kind of problems was also overtaken, therefore increasing the system robustness.

Chapter 5

Integration and Validation

This chapter starts by describing the work performed to integrate the developments made into the (modified) existing system, enhancing it with extended functionality and making it more robust. It then presents the tests that were conducted to validate the implemented solution.

Section 5.1 explains how integration was accomplished.

Section 5.2 presents the experimental setup as well as an analysis to the results obtained, enabling to better understand the benefits and drawbacks of the developed solution.

Finally, section 5.3 shows a summary and discussion about the results.

5.1 Integration With Existing System

The research work conducted in this project was led by INESC Porto [1], within the framework of the project QREN Hotel 3.0. This project was developed in partnership with the Nonius Software company [2], with the main goal of developing a recommendation engine to be integrated in their entertainment solution for the hospitality market (the NiVO system [3]). In turn, this dissertation was proposed within the context of that project, with the goal of enhancing and augment the functionality of such engine.

The proposed hybrid solution was developed to improve the existing content-based solution already being developed within this project. As was already explained in the chapter 4, this solution was successfully integrated with the existing recommender system.

An important step to validate the operation of the developed recommendation solution, as well as its performance and fulfilment of the identified requirements, was to integrate it into the NiVO system. This would allow to collect real data from the users' actions to build real user profiles,

[1] Institute for Systems and Computer Engineering of Porto - is a private non-profit association, recognised as a Public Interest Institution and an Associate Laboratory since 2002

[2] Nonius is a technology company enabling guest experience excellence for organizations in the hospitality industry

[3] For more information about this system, see http://www.noniussoftware.com/pdfs/nivo_br_en.pdf

as well as to obtain real feedback from users, providing a measure of the users' acceptance of the recommendations made.

This integration was successfully performed, proving that the solution, in operational terms, was performing as expected, smoothly interoperating with the NiVO system and delivering the expected functionality. However, the tests were only conducted in laboratory environment, as it was not possible to install the equipment in a real-world environment, with real hotel guests using it. Accordingly, the integration was only able to validate the solution in operational terms. To overcome this limitation and be able to obtain some measure of its performance, an approach based on manually generated user history records and consequent creation of corresponding user profiles, was devised and implemented.

This validation setup is described in the next section.

5.2 Solution Validation

5.2.1 Experimental Setup

As it was not possible to test the system in a real context of use, an alternative experimental setup was formulated. The essential data necessary to be able to validate the performance of the system consisted in 3 different data sets: 1) users' actions registry, i.e., a record of the users' consumption history; 2) candidate content; and 3) users' feedback to provide an indication of the quality of the recommendations made to them.

The procedure adopted to obtain these data sets, was to conduct an online survey to collect users' preferences concerning TV consumption. This survey was made to a universe of people of different ages and different professional backgrounds. In total, it was possible to collect valid responses from 60 different users. In parallel it was collected TV programming of one week by accessing the Sapo EPG API during 7 consecutive days. By comparing the responses of each user with the TV programming referred in those 7 daily EPGs, the programmes that would most likely be selected each day by each user were identified. This initial procedure has thus provided for the first data set.

For each user, the programmes appearing in their history were analysed and characterized, obtaining a set of tags that allowed to build their profiles. Then, EPGs from 7 consecutive days of the following week were collected and the programmes referred in them extracted and characterized. This step provided for the second required data set.

With all these data, the system was then able to execute its normal operation and generate lists of recommendations for 7 days for each user. The third data set was finally obtained by directly contacting 12 of the 60 users and asking them to provide their explicit opinion about the recommendations that had been generated for them.

But the performance of the system was also evaluated in other more objective dimensions, notably in terms of processing time when adopting a content-based only approach or the hybrid solution, as well as the number of recommendations generated for each user by each alternative.

Table 5.1: Distribution of Responses by Profession and Age

Profession	n	Minimum Age	Maximum Age	Average Age
Students	36	18	34	22.75
Professors	11	28	61	43.8182
Others	13	24	59	35
Total	60	18	61	29.2666

The above referred survey was made mainly to people from FEUP but also to some people from the outside, to get as many people from different realities as possible. As it is possible to see on table 5.1, responses were obtained from people with different professions and from different age groups. “Others” is related mainly to support staff from FEUP. A big concern with this survey was to try to find people from different age groups, because it is known that people with different ages usually have different habits.

As it would be very difficult to get enough answers for all kinds of content (e.g. it is difficult to ask a person which movies/services he likes from a list of thousands or millions of items), the study was only made for TV content. So, these experiments are only related with TV content (both history and recommendations). However, with the resulting profiles it would also be possible to generate recommendations for all kinds of content (TV, VoD and Services).

With these tests it was possible to analyse the TV viewing behaviour of 60 persons during a week (from Monday to Sunday). The data used for these tests is composed by 21250 programmes from 107 different channels. This represents the training data used for the user profiling. Then, with these data, recommendations were generated for the next week, using 25078 programmes (test data or candidate items).

It’s important to note that all the validation was made in a controlled environment. All the tests were performed in the same machine, without executing other processes. The MySQL database was also stored locally, but all the execution times presented in this chapter are only for the algorithms, ignoring the time of access to the database. On table 5.2 it is possible to see the computer main specifications where the tests were made.

Table 5.2: System Specifications

CPU	Intel Core i7-720QM 1,60 GHz
RAM	4GB DDR3
HDD	500GB SATA (7200 rpm)
OS	Ubuntu 12.04 LTS

5.2.2 Performance

Several tests to evaluate the system’s performance and the quality of recommendations were made. In order make the performance evaluation, the execution times for 1, 5, 15, 30 and 60 users was calculated. The users were chosen randomly between the set of users. Besides that, the kNN algorithm used in the CF technique was tested with k values of 2, 5, 8 and 10. Is known from the literature that k values around 10 bring quality to the predictions [HKB99, JZFF10]. An analysis of the MovieLens dataset also indicates that in most real-world situations, a neighbourhood of 20 to 50 seems reasonable [HKR02]. As the maximum number of users in the system is 60, it was decided that 10 neighbours in the maximum was enough for the similarity calculation.

In figure 5.1 it's made a time comparison between the calculations for CB method (Naive Bayes algorithm) and the new CF method, both used in the proposed hybrid approach. The time needed for execution of hybrid approach is the sum of both CF and CB times. The times shown in this chart are an average of the resulting times for the calculations for all *k* values named previously. As it was expected, the CF approach always takes more time than the CB approach, since that the proposed CF technique also uses the Naive Bayes algorithm to generate recommendations. However, this difference is not much significant, since this represents the similarity weight computation between all users and the rating prediction for all the features.

It's important to state that the difference of time between CF and CB approaches is not only because of the time used in the similarity computation and rating prediction on the CF approach; both CF and CB approaches use the Naive Bayes algorithm, but it can take different times in each approach, because it uses different profiles with a different number of features. Table 5.3 shows the average of features for different number of users that reflects the differences of time consumed in different techniques.

In this table is also possible to see that the features in the "User Profile" are only affected by the number of users and not the number of neighbours; they are only used in the rating calculation for the "Predicted Profile". Besides that, when there is only one user in the system, the number of features in the predicted profile is always 0. Another interesting point that can be analysed is that

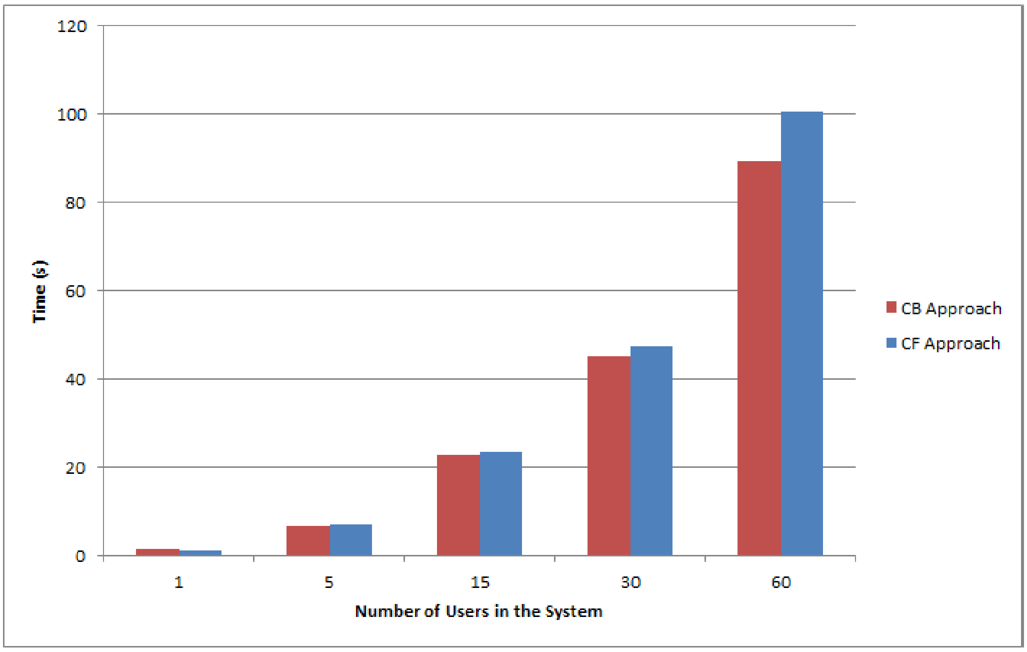

Figure 5.1: Hybrid Approach - Difference in Time Consumption Between CB and CF Changing the Number of Users in the System

Table 5.3: Average of Features in "UserProfile" and "PredictedProfile"

Number of Users	k Neighbours	Features "User Profile"	Features "Predicted Profile"
1	2	27	0
	5	27	0
	8	27	0
	10	27	0
5	2	21	13
	5	21	13
	8	21	13
	10	21	13
15	2	20,0667	12.6
	5	20,0667	20.86667
	8	20,0667	25.06667
	10	20,0667	25.7333
30	2	20,7667	13,76667
	5	20,7667	22.4
	8	20,7667	26.9333
	10	20,7667	26.8333
60	2	21,25	13.2167
	5	21,25	22.2167
	8	21,25	26.5667
	10	21,25	28.4833

for 5 users in the system, the number of features calculated with 2 neighbours and 5 neighbours is the same, since there was not enough users with the minimum threshold (0) to be used in the collaborative profile prediction.

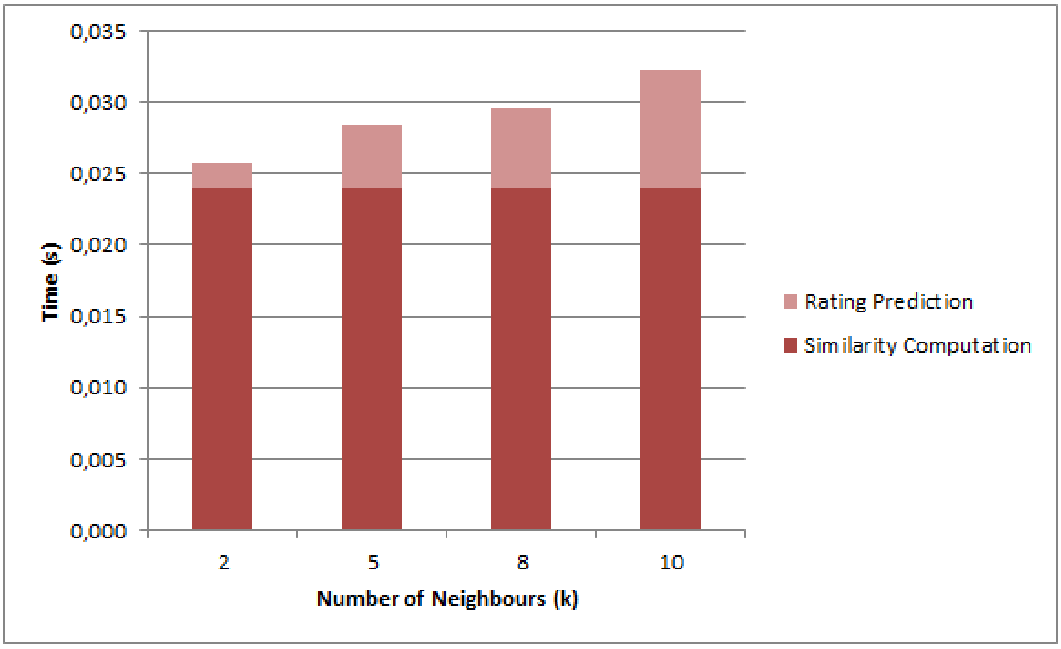

Figure 5.2: Similarity Computation and Rating Prediction Time Consumption for Different k Values for 60 Users

Figure 5.2 shows the time consumption for similarity computation and rating prediction for different k values for 60 users. As it was already expected, the similarity computation time is constant for different k values, since this time only depends on the number of users in the system and not on the number of neighbours that are being considered. On the other hand, the rating prediction time increases when the number of neighbours also increases. This happens because when the predicted profile is being calculated and there are more users being considered, it's necessary to make more calculations.

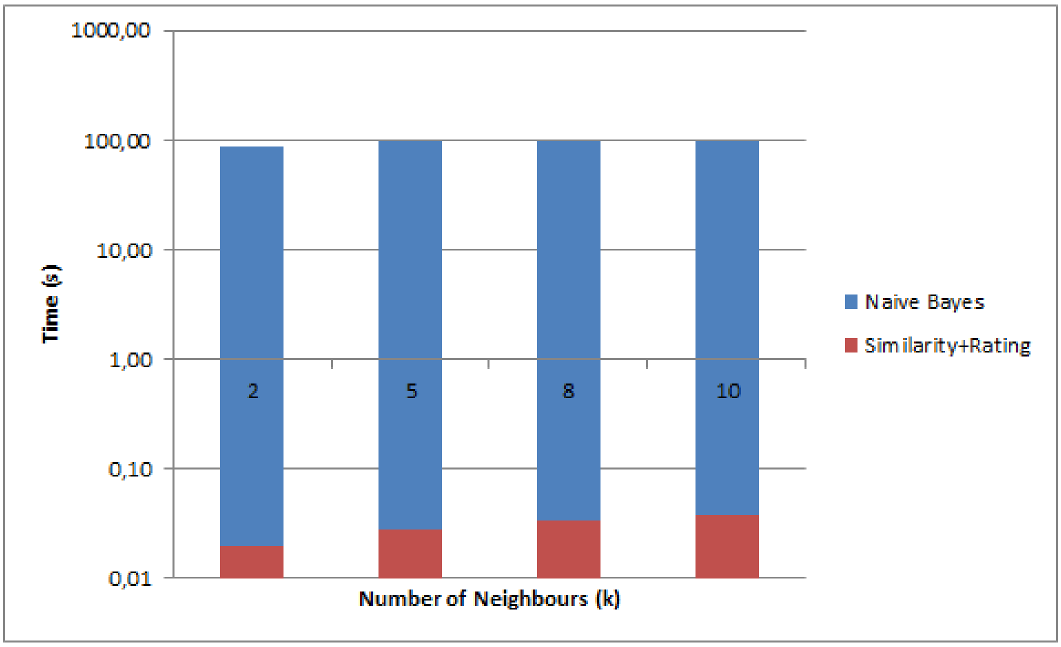

Figure 5.3: CF Recommendation Generation Process Time Consumption for Different k Values for 60 Users

Figure 5.3 shows the time consumption for CF approach. The presented chart represents the time consumption for the similarity computation and rating prediction (shown previously in the figure 5.2), and also the time consumed by the Naive Bayes algorithm to generate the recommendations. From the analyses of this chart, is possible to conclude that the biggest part of time is consumed by the Naive Bayes algorithm to the recommendation generation. For 60 users, the average time for the similarity computation plus the rating prediction for the different values of k is 0,028986 seconds, and for the Naive Bayes is 1 minute and 40,166852 seconds, which means that the Naive Bayes algorithm in the CF approach consumes 99,97107% of the time on average.

5.2.3 Recommendations

Besides the analyses of the system's performance, it's also necessary to analyse the generated recommendations. First of all, since that the user profiling technique was changed, it's important to see the impact its in the system.

The chart in figure 5.4 shows the difference between the number of recommendations using the old profiling technique in the original system and the new profiling technique implemented in

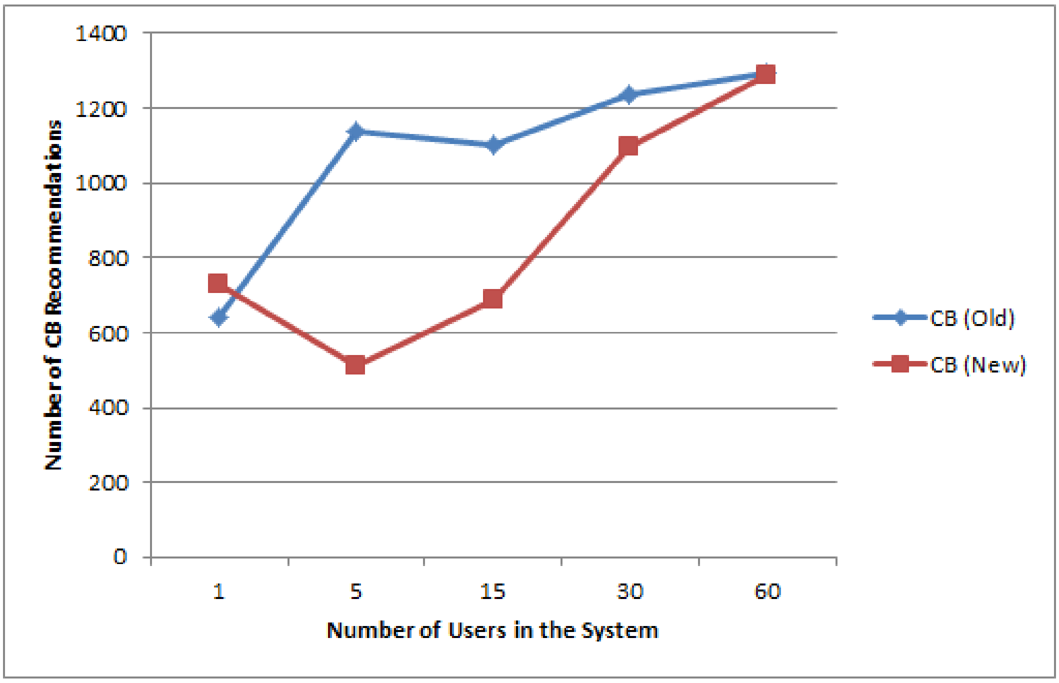

Figure 5.4: Number of Recommendations with Different User Profiling Approaches

the new hybrid approach. It's important to note that his numbers correspond to all the recommendations, without any filtering for the top classified. As it's possible to see, the new user profiling technique, on average, generates less recommendations than the previous one. In practise this represents that the system specializes more in the users interests.

The average of recommendations of the new technique in comparison with all the items in the test set (25078) is 2.92% in contrast to the 4.54% of the previous technique. In practice, a user had 25018 programmes to analyse in each of the 107 channels during a week, and the system would recommend to him only 731 on average (for the whole week). This is a very important aspect, because one of the main problems that the recommender systems try to overcome is the increasing amount of data that makes impossible for the users to analyse exhaustively all the available items.

Figure 5.5 represents the average number of recommendations for different numbers of users. This number is the average of recommendations for all k neighbours for each number of users presented. For only 1 user in the system there are 0 recommendations given, insofar as there are no users to generate collaborative recommendations. As the number of users in the system increases, the number of recommendations in average starts to decrease getting closer to 600. This is because as many users are in the system, it's easier to find groups of similar users, therefore generating more specific recommendations.

When analysing the specific case with 60 users in the system in the chart presented in 5.6, it's also possible to see the same evidence that was analysed in the figure 5.5. As the number of used neighbours in the recommendation generation increases, the number of recommendations decreases.

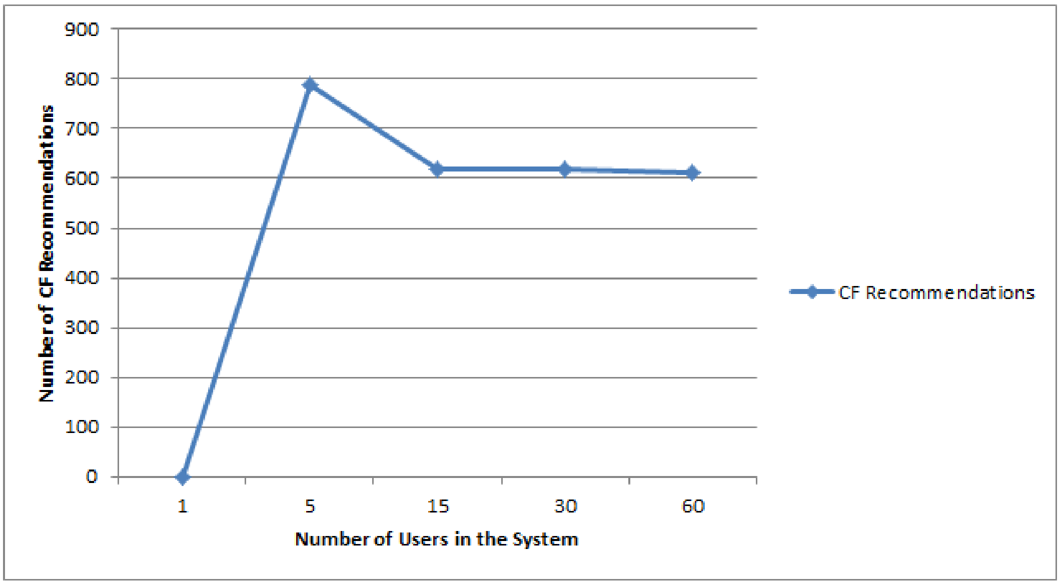

Figure 5.5: Average Number of Recommendations with Different Numbers of Users

In the context of recommendation, the purpose of a classification task is to identify the *n* most relevant items for a given user. In chapter 2.5 (metrics for evaluation) were presented the *Precision* (2.4) and *Mean Absolute Error (MAE)* (2.2), two of the best-known classification metrics [HKR02]. *MAE* compares the generated predictions with the real evaluations made by the users. As these tests were not made with real data usage obtained from the viewing history from the system (implicit profiling), it was not possible to use MAE as a metric to evaluate these results. So, the selected metric for measuring the quality of recommendations was *Precision*. This metric

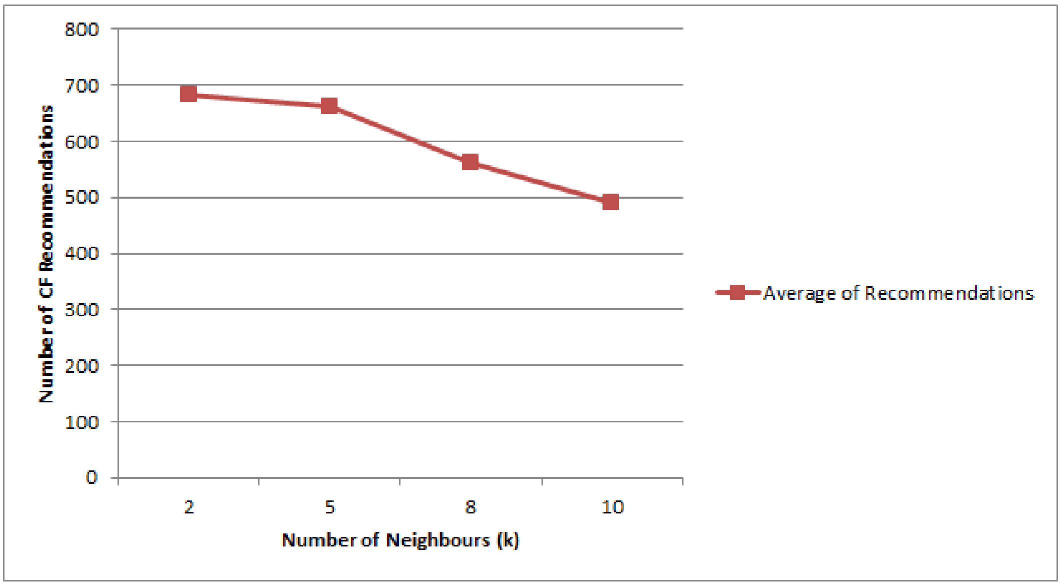

Figure 5.6: Number of CF Recommendations for Different k Values for 60 Users

measures the proportion of recommendations that are good recommendations.

To do this, with the training data given by the viewing behaviour of 60 users that was used for the user profiling, recommendations for one more week were generated. Then, 12 people from the initial set of 60 were asked for rating two sets of items: one, generated with the initial content-based approach, and the other with the proposed hybrid approach. The rest of the profiles were only used as possible neighbours for collaborative profiling creation ("Predicted Profile"). The users were only asked to classify each recommendation as "good recommendation" or "bad recommendation". The obtained precision results were the following presented on table 5.4.

Table 5.4: Precision of Recommendations

Profession	**n**	**CB Recommendations Precision**	**Hybrid Recommendations Precision**	**Precision Improvement**
Students	4	79%	91%	12%
Professors	4	74%	83%	9%
Others	4	78%	89%	11%
Total	12%	77%	87.66667%	10.66667%

According to the results presented on table 5.4, it's possible to conclude that the proposed hybrid approach has an overall Precision of 87.66667%, which is good in comparison with other precision results presented in the literature [SS09]. Besides that, the Precision improvement compared to the initial approach was of 10.66667%, which is a very satisfactory result.

Besides the evaluation in terms of precision, the users were also asked to rate globally the recommendations as "not surprising", "neither surprising nor unsurprising", "surprising". This question was made to measure the recommendations serendipity [BO11, JZFF10, Zan08]. This is an important aspect, since one of the main goals of hybrid approaches is to overcome the over-specialization problem (section 2.3.8), generating surprising recommendations that are appreciated by the user.

Table 5.5: Precision of Recommendations

	CB Recommendations	**Hybrid Recommendations**
Not Surprising	5	1
Neither Surprising Nor Unsurprising	4	3
Surprising	3	8

The results to this question are presented on table 5.5. When talking about the previous CB recommendations, 5 users (41.6667%) think that the recommendations are not surprising, and only 3 users (25%) say that they are surprising.

Comparatively, 8 users (66.6667%) say that the proposed hybrid approach generates surprising recommendations, while only one (8.3333%) says that the recommendations are not surprising.

5.3 Summary

As indicated in the previous section, the recommendation generation using Naive Bayes algorithm is the most expensive task in terms of computing time. In order to evaluate the global performance changes between the original CB method and the proposed hybrid approach, the execution times for 1, 5, 15, 30 and 60 users were compared.

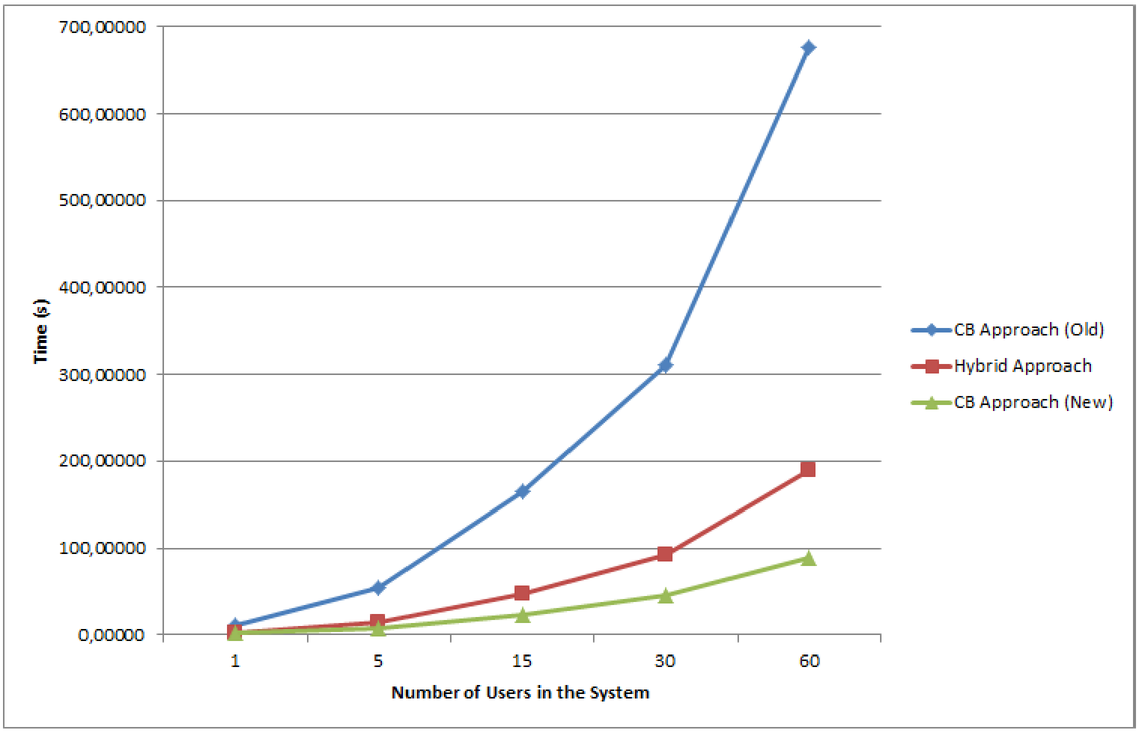

Figure 5.7: Performance Comparison Between Approaches

Figure 5.7 shows the comparison between these different approaches. The blue line represents the time elapsed to run the first content-based approach. The green line represents the time of the new content-based approach implemented in the new hybrid system running solo (this is shown here only for comparison reasons). The red line (in the middle) represents the developed hybrid approach execution time.

As is possible to see, new approach performance is significantly better. For 60 users, the original approach needed 11 minutes and 16.382252 seconds on average to make all the predictions. On the other hand, the new CB approach only took 1 minute and 29.430576 seconds, which represents an increase of performance of 756.3%. The proposed hybrid approach, that combines this CB approach with the new CF technique, only took 3 minutes and 14.289372 seconds, which represents an increase of performance of 356% in this situation. So, even with more calculations to the collaborative profiles generation, it was possible to increase the performance of the original recommender engine.

These differences in performance can be explained by the changes that were made in the process of user profiling and recommendation. The used profiling technique was slightly changed, and now it has less calculations. Besides that, the used containers to store the user profile were also improved. This is what had more impact in the system, because to make predictions for each item in the system, it's necessary to lookup for the features within the user profile. Initially this was done by searching in all elements of a vector for the desired feature. When using a HashMap to do this, we can simply search for a feature and get the correspondent weight, without the need of search in all the elements in the HashMap. When we're dealing with 25018 programmes, generating recommendations for 60 users, each one with an average of 45.64 features in the profiles ("User Profile" + "Predicted Profile"), a small change in the feature organization and access can make a big difference in the end.

Another important conclusion to retain from the solution validation is the precision improvement of 10.6667%, which is translated in an average precision for the approach of 87.6667%.

Besides that, a significant portion of the inquired users (66.6667%) said the system was able to generate surprising recommendations, against only 8.3333% that said the system did not generated surprising recommendations. This is a very important aspect, since one of the main goals of hybrid approaches is to overcome the over-specialization problem, generating surprising recommendations that are appreciated by the users.

Chapter 6

Conclusions

The recommendation engine that was being developed within the Hotel 3.0 project, had adopted a content-based approach and focussed on recommending television content. Some of the limitations that were identified in this initial implementation are: 1) it is known from the literature that a content-based approach can rapidly become very focussed, providing the user always the same type of recommendations ("over-specialization" problem); 2) it only addresses TV content, neglecting other types of resources and services the guest has access to; 3) it neglects the possibility to establish relationships both across the users and other resources, to increase the performance and enrich the type of recommendations.

In this context, the main motivation of this dissertation was to investigate solutions to overcome these limitations. In particular, it aimed at exploring the use of hybrid approaches, by establishing a measure of proximity between guests and thus include in the recommendation list of one guest, items that would have been preferred by other guests with similar profile. In the initial phase, the implemented system only made recommendations based on the characteristics of the items consumed by the users – content-based filtering. The content-based approaches only try to find items with similar characteristics to others that the user consumed before. Trying to find similarities between users, it is possible to make completely unexpected recommendations to them – this is the main advantage of collaborative approaches.

The first identified challenge was related to the creation of a compact, yet complete and flexible, structure and set of tags to characterize and represent content. Given that the area of application covered a broad range of resources (from TV programmes to hotel services, through video-on-demand), an extended feature domain was defined with the capacity to cover any type of item that the system would need to process. This extended feature domain applied to any type of resources, allows the system to deal with a large variety of items and make cross-content recommendations.

To solve the limitation of addressing only TV content, which prevented the possibility to recommend other type of resources and services the guest had access to, the system was enriched with content from video-on-demand and hotel services, establishing relationships across them using the features domain previously defined.

Additionally, to overcome some limitations in the previously implemented user profiling technique, instead of creating two profiles, the “Accepted Profile” and the “Rejected Profile”, it was decided to use only one profile containing all the domain features with an associated weight in the range [0;1]. The previously implemented user profiling technique was also changed: the learning rate α was removed from the weight calculation when a feature is new to a user profile. The feature weight will only be influenced by this factor during the update process. This decision was made to try to avoid the “New User Problem” and start generating recommendations for new users in the system as soon as possible. This is a very important aspect in the domain of hospitality services, since a significant portion of users can use the system very sporadically, so the system needs to respond quickly to their needs.

Although there are different techniques described in the literature to implement the CF recommendation approach, this dissertation proposes a novel CF approach based on k-Nearest-Neighbour (kNN) algorithm. It can be divided into three steps: 1) measurement of similarities between the target and the remaining users in the database, using an improved Pearson Correlation method; 2) selection of the most similar users who will serve as recommender, applying a top-N filtering (to chose the N most similar users) and threshold filtering (ignoring neighbours with a similarity degree less than 0); and 3) generate a “Predicted Profile” based on the weighted average of the neighbours’ profile features, weighted by their similarity to the target user.

The adopted approach creates a new profile to the target user based on his neighbours’ profiles, the “Predicted Profile”, instead of recommending particular items previously consumed by those neighbours. Then, it applies the content-based method to find candidate content similar to the preferences registered in this new “Predicted Profile”.

This novel method proposed in this dissertation adopts a different approach to address the sparsity problem, since there isn’t a user-item rating matrix used in recommendations generation. This can be considered very important, especially when dealing with a large quantity and diversity of resources.

The developed recommendation engine can be considered as an enhanced version of the previously implemented system. Not only it extends the scope of recommendations from the TV domain to include also other types of resources, such as VoD or hotel services, as it incorporates in the process a CF recommendation approach to improve the results of the CB approach. Moreover, it presents an architecture flexible enough to allow the operator to decide: 1) which domains the recommendation process should apply to (by including or excluding domains out of the complete universe of candidate content - the applied design pattern enables to easily fetch new types of candidate content using apposite web services) and; 2) whether the system should use the combination of the two recommendation techniques (CF and CB) or just one them.

Although it was not possible to validate the system in a real context of use, it was possible to test the system performance and the recommendation quality using a TV viewing history for 60 inquired users. With these tests it was possible to analyse the TV viewing behaviour of 60 persons during a week (from Monday to Sunday). The data used for these tests is composed by 21250 programmes from 107 different channels. This represents the training data used for the user profiling. Then, with these data, recommendations were generated for the next week, using 25078 programmes (test data or candidate items).

From the results of the solution validation, we can conclude that the new approach performance is significantly better. For 60 users, the original approach needed 11 minutes and 16.382252 seconds on average to make all the predictions. On the other hand, the new CB approach only took 1 minute and 29.430576 seconds, which represents an increase of performance of 756.3%. The proposed hybrid approach, that combines this CB approach with the new CF technique, only took 3 minutes and 14.289372 seconds, which represents an increase of performance of 356% in this situation. So, even with more calculations to the collaborative profiles generation, it was possible to increase the performance of the original recommender engine.

Another important conclusion to retain from the solution validation, is the precision improvement of 10.6667%, which is translated in an average precision for the approach of 87.6667%.

Besides that, a significant portion of the inquired users (66.6667%) said the system was able to generate surprising recommendations, against only 8.3333% that said the system did not generated surprising recommendations. This is a very important aspect, since one of the main goals of hybrid approaches is to overcome the over-specialization problem, generating surprising recommendations that are appreciated by the user.

6.1 Future Work

As was explained previously, it was not possible have available VoD and hotel services data. Although the system was tested with some sample data, it's necessary to validate the system with real data from other types of content, besides TV programmes. This is a very important aspect to validate the cross-content recommendations that can already be made with the proposed solution. For example, with the already created users profiles, when it is possible to get data from these resources, it will be also possible to generate recommendations to them without any changes.

It would also be interesting to make a precision analyses for the recommendation generation for these types of content. This is important to understand if the changes in features domain were enough to support all types of data, or if it would be necessary to improve the tag list to cover all the items.

With the increase number of users in the system, it would also be good to analyse other values for the number of neighbours (k) that could improve the precision of the system. With real data from users' behaviour, it would also be possible to validate the system with other metrics, like Mean Absolute Error (MAE).

Another important aspect to take in mind in the future work, is to explore other content-based techniques that can improve the performance of the Naive Bayes algorithm already implemented. As was concluded during the solution validation, during the execution of CF approach, the Naive

Bayes algorithm consumes 99.97107% of the total time. Finding another approach with better performance, could improve the global system performance significantly.

Content characterization is another aspect of great importance to be analysed in the future. Although this was not a goal for this dissertation, the proposed hybrid solution also depends on the content characterization quality. Although the content from VoD and hotel services can provide enough metadata to be characterized properly, in some cases TV content depends in the "bag of words method" to be characterized. So, this aspect should also be taken into consideration in the future.

References

[AT05] 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, June 2005.

[Bel92] NJ Belkin. Information filtering and information retrieval: two sides of the same coin? *Communications of the ACM*, 29(10):1–10, 1992.

[BFLNPA+11] Yolanda Blanco-Fernández, Martín López-Nores, José Pazos-Arias, Jorge García-Duque, and Manuela Martín-Vicente. TripFromTV+: Targeting Personalized Tourism to Interactive Digital TV Viewers by Social Networking and Semantic Reasoning. *IEEE Transactions*, pages 953–961, 2011.

[BHCO98] Chumki Basu, H Hirsh, W Cohen, and Others. Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the National Conference on Artificial Intelligence*, pages 714–720. JOHN WILEY & SONS LTD, 1998.

[BHK98] John S Breese, David Heckerman, and Carl Kadie. Empirical Analysis of Predictive Algorithms for Collaborative Filtering. *Proceedings of the 14th conference on Uncertainty in Artificial Intelligence*, 461(8):43–52, 1998.

[BHY97] R.D. Burke, K.J. Hammond, and BC Yound. The FindMe approach to assisted browsing. *IEEE Expert*, 12(4):32–40, 1997.

[BKV08] RM Bell, Y Koren, and C Volinskt. The bellkor 2008 solution to the netflix prize. *Statistics Research Department at*, (12):1–21, 2008.

[BMCMB+10] Ana Belén Barragáns-Martínez, Enrique Costa-Montenegro, Juan C. Burguillo, Marta Rey-López, Fernando a. Mikic-Fonte, and Ana Peleteiro. A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition. *Information Sciences*, 180(22):4290–4311, November 2010.

[BO11] Cleomar B. Jr and Marcius Oliveira. Recommender Systems in Social Networks. *Revista de Gestão da Tecnologia e Sistemas de Informação*, 8(3):681–716, 2011.

[BP00] Daniel Billsus and M.J. Pazzani. User modeling for adaptive news access. *User modeling and user-adapted interaction*, 10(2):147–180, 2000.

[BRBG08] Amancio Bouza, Gerald Reif, Abraham Bernstein, and Harald Gall. SemTree: Ontology-Based Decision Tree Algorithm for Recommender Systems. *In Proceedings of the 7th International Semantic Web Conference*, 2008.

[BS97] Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3), 1997.

[Bur02] Robin Burke. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction*, 12(4):331–370, 2002.

[Bur07] Robin Burke. Hybrid web recommender systems. In *The adaptive web*, pages 377–408. Springer-Verlag, 2007.

[CGM+99] M. Claypool, A. Gokhale, T. Miranda, P. Murnikov, D. Netes, and M. Sartin. Combining content-based and collaborative filters in an online newspaper. In *Proceedings of ACM SIGIR Workshop on Recommender Systems*. Citeseer, 1999.

[CML+11] Rubén González Crespo, Oscar Sanjuán Martínez, Juan Manuel Cueva Lovelle, B. Cristina Pelayo García-Bustelo, José Emilio Labra Gayo, and Patricia Ordoñez De Pablos. Recommendation System based on user interaction data applied to intelligent electronic books. *Computers in Human Behavior*, 27(4):1445–1449, July 2011.

[CS00] Paul Cotter and Barry Smyth. Ptv: Intelligent personalised tv guides. In *Proceedings Of The National Conference On Artificial Intelligence*, pages 957–964. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2000.

[Erl05] Thomas Erl. *Service-Oriented Architecture: Concepts, Technology, and Design*. The Prentice Hall Service-Oriented Computing Series from Thomas Erl Series. Prentice Hall PTR, 2005.

[FGG97] Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian Network Classifiers. *Machine Learning*, 29(1):131–163, 1997.

[GCP03] S Gauch, J Chaffee, and A. Pretschner. Ontology-based personalized search and browsing. *Web Intelligence and Agent Systems*, 1(3/4):219–234, March 2003.

[Gha10] M Ghazanfar. An Improved Switching Hybrid Recommender System Using Naive Bayes Classifier and Collaborative Filtering. I, 2010.

[GNOT92] David Goldberg, David Nichols, B.M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[GR05] Jacques Guyot and Saïd Radhouani. Ontology-based multilingual information retrieval. *Working Notes in Multilingual Textual Document Retrieval Track*, (1):21–23, 2005.

[HKB99] JL Herlocker, JA Konstan, and A Borchers. An algorithmic framework for performing collaborative filtering. *Proceedings of the 22nd*, 1999.

[HKR02] Jon Herlocker, Joseph A. Konstan, and John Riedl. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Information retrieval*, 5:287–310, 2002.

[HKTR04] Jonathan L. Herlocker, Joseph a. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, January 2004.

[HSRF95] 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. ACM Press/Addison-Wesley Publishing Co., 1995.

[JZFF10] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich. *Recommender Systems: An Introduction*, volume 40. Cambridge Univ Pr, 2010.

[KRRS11] PB Kantor, F Ricci, L Rokach, and B Shapira. *Recommender Systems Handbook*. Springer US, Boston, MA, 2011.

[LB03] Daniel Lemire and Harold Boley. RACOFI: A Rule-Applying Collaborative Filtering System. *Database*, (October):13–24, 2003.

[LG07] George Lekakos and George M. Giaglis. A hybrid approach for improving predictive accuracy of collaborative filtering algorithms. *User Modeling and User-Adapted Interaction*, 17(1-2):5–40, January 2007.

[LSY03] Greg Linden, Brent Smith, and Jeremy York. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[Mir01] B.J. Mirza. *Jumping Connections: A Graph-Theoretic Model for Recommender Systems*. PhD thesis, 2001.

[MK08] Eleni Mangina and John Kilbride. Evaluation of keyphrase extraction algorithm and tiling process for a document/resource recommender within e-learning environments. *Computers & Education*, 50(3):807–820, April 2008.

[MMN01] Prem Melville, Raymond J Mooney, and Ramadass Nagarajan. Content-boosted collaborative filtering. In *Proceedings of the 2001 SIGIR Workshop on Recommender Systems*, 2001.

[MR99] R.J. Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the SIGIR-99 Workshop on Recommender Systems*, number August. ACM, 1999.

[Paz99] M.J. Pazzani. A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13(5):393–408, 1999.

[Paz07] D Pazzani, M; Billsus. Content-based recommendation systems. *The adaptive web*, pages 325–341, 2007.

[PB97] Michael Pazzani and Daniel Billsus. Learning and Revising User Profiles: The Identification of Interesting Web Sites. *Machine Learning*, 27(3):313–331, 1997.

[PGF04] S Perugini, M.A. Gonçalves, and E.A. Fox. Recommender systems research: A connection-centric survey. *Journal of Intelligent Information Systems*, 23(2):107–143, 2004.

[Ram10] José Ramos. *Algoritmos Colaborativos para Sistemas de Recomendação*. PhD thesis, FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO, 2010.

[RIS+94] Paul Resnick, N. Iacovou, M. 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. ACM, 1994.

[RM08] L Rokach and O Z Maimon. *Data mining with decision trees: theroy and applications*. World Scientific, 2008.

[RT08] K. Nageswara Rao and V.G. Talwar. Application Domain and Functional Classification of Recommender Systems - A Survey. *DESIDOC Journal of Library & Information Technology*, 28(3):17–35, 2008.

[Sch98] H Schildt. *C++: The Complete Reference Third Edition*. Number 926553365. Osbourne McGraw-Hill, 1998.

[SKJ01] Ingo Schwab, Alfred Kobsa, and Ivan Joychev. Learning user interests through positive examples using content analysis and collaborative filtering. *Internal Memo, GMD, St. Augustin,*, 2001.

[SKKR00] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Application of dimensionality reduction in recommender system-a case study. In *Proceedings of the ACM WebKDD Workshop*, 2000.

[SM93] B Sheth and P Maes. Evolving Agents for Personalized Information Filtering. In *Proceedings The Ninth Conference on Artificial Intelligence for Applications*, pages 345–352. IEEE Comput. Soc. Press, 1993.

[SM95] U. 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. ACM Press/Addison-Wesley Publishing Co., 1995.

[Som07] Ian Sommerville. *Software Engineering*. International Computer Science Series. Addison-Wesley, 2007.

[SS09] S. Staab and R. Studer. *Handbook on Ontologies*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2 edition, 2009.

[Wal06] HM Wallach. Topic modeling: beyond bag-of-words. *Proceedings of the 23rd international conference on*, (1), 2006.

[Wan98] Pei Wang. Why recommendation is special. In *Workshop on Recommender Systems, part of the 15th National Conference on Artificial Intelligence (AAAI 1998)*, pages 111–113, 1998.

[WTLH06] Sung-Shun Weng, Hsine-Jen Tsai, Shang-Chia Liu, and Cheng-Hsin Hsu. Ontology construction for information classification. *Expert Systems with Applications*, 31(1):1–12, July 2006.

[YZ04] Zhiwen Yu and Xingshe Zhou. TV3P: an adaptive assistant for personalized TV. *Consumer Electronics, IEEE Transactions on*, 50(1):393–399, 2004.

[Zan08] Markus Zanker. A collaborative constraint-based meta-level recommender. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 139–146. ACM, 2008.

[ZJ09a] Markus Zanker and Markus Jessenitschnig. Case-studies on exploiting explicit customer requirements in recommender systems. *User Modeling and User-Adapted Interaction*, 19(1):133–166, July 2009.

[ZJ09b] Markus Zanker and Markus Jessenitschnig. Collaborative Feature-Combination Recommender Exploiting Explicit and Implicit User Feedback. *2009 IEEE Conference on Commerce and Enterprise Computing*, pages 49–56, July 2009.

[Zur92] J M Zurada. *Introduction to Artificial Neural Systems*. West Publishing Company, 1992.