Systèmes de recommendation : adaptation Dynamique et Argumentation

Julien Gaillard

HAL Id: tel-01168476
https://tel.archives-ouvertes.fr/tel-01168476

Submitted on 25 Jun 2015

THÈSE

présentée à l'Université d'Avignon et des Pays de Vaucluse
pour obtenir le diplôme de DOCTORAT

SPÉCIALITÉ : Informatique

École Doctorale 536 « *Sciences et Agrosciences* »

Laboratoire Informatique d'Avignon (EA 931)

Recommender Systems : Dynamic Adaptation and Argumentation

par

Julien Gaillard

Soutenue publiquement devant le jury composé de :

Pr.	**Esther Pacitti**	LIRMM, Montpellier	Rapporteur
Pr.	**Éric Gaussier**	LIG, Grenoble	Rapporteur
Pr.	**Marc El-Bèze**	Université d'Avignon	Directeur de thèse
Pr.	**Eitan Altman**	Université d'Avignon - INRIA	Directeur de thèse
Pr.	**Emmanuel Ethis**	Université d'Avignon	Directeur de thèse

Laboratoire Informatique d'Avignon

Acknowledgments

First of all, I would like to thank all members of the PhD committee for having kindly agreed to evaluate my work. I am honored that Esther Pacitti and Éric Gaussier have agreed to report on my thesis.

I would like to warmly thank my supervisor Marc El-Bèze, for the constant support, help and confidence he has shown me the last three years. Sharing the same office was a very enriching experience: it allowed us to interact continuously and have many constructive discussions.

I warmly thank my co-supervisor Eitan Altman, for his support, inspiration and relevant advices. The brainstorming sessions were always enthralling.

I warmly thank my co-supervisor Emmanuel Ethis, for giving me the opportunity to approach the problems from a sociological point of view, which is rather unusual and very enriching for a computer scientist.

I would like to thank Agorantic and its members, for trusting in my thesis project since the beginning. I also thank the European project Congas. I thank Cyril Barthet, Benoit de Malartic and Chris Navas (Vodkaster) for providing the data and their feedback on my work.

I'd like to thank my fellow PhD students and collaborators who contributed to this research, *La Maison de la Recherche*, Aude Mosca and Caroline Girerd-Potin. To my musician friends and bandmates, thank you for all the music we have shared together and the support you gave me during this thesis. I am very grateful to all of you.

Last but not least, I'd like to thank Jean-Michel Renders for inviting me to present my work at the Xerox Research Center Europe and for our recent collaboration.

Finally, the writing of this thesis would not have been possible without the unlimited support, understanding and love from my family. You are amazing.

To my grandparents, who were born in a time where studying was not always easy. I'd like to have a special thought to my Grandpa' Michel Cazorla, who had no choice but to leave school at the age of 14 and start working as a baker to feed his family, when his father died.

Thank you so much everyone. This is just the beginning of a great adventure!

Dedicated to my family.

CONTENTS

INTRODUCTION

This thesis presents the results of a multidisciplinary research project (Agorantic) on Recommender Systems.

The goal of this work was to propose new features that may render recommender systems (RS) more attractive than the existing ones. We also propose a new approach to and a reflection about evaluation.

In designing the system, we wanted to address the following concerns:

1. People are getting used to receive recommendations. Nevertheless, after a few bad recommendations, users will not be convinced anymore by the RS.

2. Moreover, if these suggestions come without explanations, why people should trust it?

3. The fact that item perception and user tastes and moods vary over time is well known. Still, most recommender systems fail to offer the right level of "reactivity" that users are expecting, i.e. the ability to detect and to integrate changes in needs, preferences, popularity, etc. Suggesting a movie a week after its release might be too late. In the same vein, it could take only a few ratings to make an item go from *not advisable* to *advisable*, or the other way around.

4. Users might be interested in less popular items (in the " long tail") and want less systematic recommendations.

To answer these key issues, we have designed a new semantic and adaptive recommender system (SARS) including three innovative features, namely Argumentation, Dynamic Adaptation and a Matching Algorithm.

- Dynamic Adaptation: the system is updated in a continuous way, as each new review/rating is posted. (Chapter 4)

- Argumentation: each recommendation relies on and comes along with some keywords, providing the reasons that led to that recommendation. This can be seen as a first step towards a more sophisticated argumentation. We believe that, by making users more responsible for their choices, it will prevent them from losing confidence in the system. (Chapter 5)

- Matching Algorithm: allows less popular items to be recommended by applying a matching game to users and items preferences. (Chapter 6)

The system should be sensed as less intrusive thanks to relevant arguments (well-chosen words) and less responsible to unsatisfaction of the customers.

We have designed a new recommender system intending to provide textually well-argued recommendations in which the end user will have more elements to make a well-informed choice. Moreover, the system parameters are dynamically and continuously updated, in order to provide recommendations and arguments in phase with the very recent past. We have included a semantic level, i.e words, terms and phrases as they are naturally expressed in reviews about

items. We do not use tags or pre-determined lexicon.

The performances of our system are comparable to the state of the art. In addition, the fact that it provides argumentations makes it even more attractive and could enhance customers loyalty.

Publications & Talks

by Julien Gaillard

International Journals

- Majed Haddad, Julien Gaillard, Eitan Altman, Dieter Fems. *Paradoxes in Semi-Dynamic Evolutionary Power Control Game: When Intuition Fools You!*
  IEEE Transactions on Wireless Communications (2013)

French Journals

- Myriam Dougados, Jean-Louis Fabiani et Julien Gaillard. *Qui m'aime me suive. Les usages de Twitter dans la production et la diffusion des opinions esthétiques*
  Culture et Musées (2014)

International Conferences

- Jean-Valère Cossu, Julien Gaillard, Killian Janod, Emmanuel Ferreira and Marc El-Bèze. *LIA@Replab 2014 : 10 methods for 3 tasks.* Proceedings of the 5th International Conference of the CLEF initiative. **(2014, Sheffield, UK)**

- Julien Gaillard, Marc El-Bèze, Eitan Altman, Emmanuel Ethis. *Flash reactivity: Adaptive models in recommender systems.*
  Proceedings of the 2013 International Conference on Data Mining. **(2013, Las Vegas, NV)**

- Julien Gaillard, Marc El-Bèze, Eitan Altman, Emmanuel Ethis. *Well-argued recommendation: adaptive models based on words in recommender systems.*
  Proceedings of the 2013 Joint Conference on Empirical Methods on Natural Language Processing - **EMNLP (2013, Seattle, WA)**

- Eitan Altman, Julien Gaillard, Dieter Fiems, Majed Haddad. *Semi-Dynamic Hawk and Dove Game Applied to Power Control.*
  The 31st Annual IEEE International Conference on Computer Communications - INFO-COM **(2012, Orlando, FL)**

- Majed Haddad, Julien Gaillard, Eitan Altman, Dieter Fiems. *A Semi-dynamic Evolutionary Power Control Game.*
  The 11th International Conference on Networking **(2012, Prague, Czech Republic)**

- Julien Gaillard, Eitan Altman, Majed Haddad, Piotr Wiecek. *Dynamic Hawk and Dove Games within Flocks of Birds.*

Proceedings of the 2011 International ICST Conference on Bio-Inspired Models of Network, Information, and Computing Systems **(2011, York, England)**

French Conferences

- Jean-Valère Cossu, Julien Gaillard, Marc El-Bèze, Juan-Manuel Torres-Moreno. *Contextualisation de messages courts : l'importance des métadonnées.*
13e Conférence Francophone sur l'Extraction et la Gestion des Connaissances. **(2013, Toulouse, France)**

International Talks

- Julien Gaillard, *Well-argued recommendation: adaptive models based on words in recommender systems.*
International Conference on Empirical Methods on Natural Language Processing - EMNLP. **Seattle, WA, October 2013**

- Julien Gaillard, *Flash reactivity: Adaptive models in recommender systems.*
International Conference on Data Mining, WORLDCOMP'13 **Las Vegas, NV, July 2013**

- Julien Gaillard, *Semi-Dynamic Hawk and Dove Game Applied to Power Control.*
The 31st Annual IEEE International Conference on Computer Communications **Orlando, FL, March 2012**

- Julien Gaillard, *A Semi-dynamic Evolutionary Power Control Game.*
The 11th International Conference on Networking **Prague, Czech Republic, May 2012**

- Julien Gaillard, *Dynamic Hawk and Dove Games within Flocks of Birds. .*
International ICST Conference on Bio-Inspired Models of Network, Information, and Computing Systems, BIONETICS **York, England, December 2011**

Notations

In this section, we list the objects that are manipulated by a recommender system. In the remainder, we will use the following notations:

M: number of users

N: number of items

$u, v \in (1, .., M)$: indexes for users

$i, j \in (1, .., N)$: indexes for items

U: set of users

I: set of items

$S_u :\subseteq I$ set of items rated by u

$T_i :\subseteq U$ set of users who rated i

$r_{u,i}$: rating given by user u on item i

$\hat{r}_{u,i}$: predicted rating given by user u on item i

$Sim(x, y)$: any similarity function

CHAPTER

1

STATE OF THE ART

Contents

Abstract

The aim of this chapter is to give an overview of the most important concepts used in recommender systems and present existing research work. We will focus on classical methods and explanation interfaces. Thus we will have a solid basis to start from.

1.1 Introduction

Although automatic recommendation has emerged as an independent field of research and became popular in the 1990's, one could say that it is somehow taking root in the late 1970's with the work of Rich (Rich, 1979) in cognitive sciences, in which he uses stereotypes to perform a user modeling task. Some work in information retrieval and forecasting theory might be considered as the beginning of recommendation as a research area (Salton, 1989) (Armstrong, 2001).

Experience goods (Nelson, 1970) identify assets that need to be consumed before knowing their satisfaction level. Cultural contents can be seen as the quintessential experience good, one that creates a multibillion-dollar global industry, one that is based on a myriad of genres and countless artists.

Consumers are faced with the difficult task of using their limited budgets to acquire some of these contents, without fully knowing how fulfilling they are. In such situations, recommendations can offer a substantial improvement in decision making of what to purchase. Online stores that incorporate a recommender system are becoming more and more competitive with respect to traditional stores. The last decade has shown a historical growth of websites offering online services.

It is not only electronic commerce services that gain from recommendation, but any service based on a large amount of contents. Indeed, there are several multimedia platforms containing a huge amount of cultural products such as Spotify (music), Netflix (video on demand) and so on. Recommender systems aim at suggesting appropriate items to consumers from a large catalog of products.

These systems are designed depending on the domain and the data source available. For instance, Netflix users give ratings to the movie they watched, on a scale of 1 (disliked) to 5 (liked). Additionally, the system may have access to user-specific and item-specific profile attributes such as demographics and product descriptions respectively.

Nowadays, recommender systems and their evaluation on real-world issues is a very active area of research.

1.2 Recommender Systems

1.2.1 Definition

Basically, to keep things simple, a recommender system is able to provide suggestions (recommendations) to users, in multiple contexts such as when they are making a choice among a large

catalog of items or whenever they want to receive suggestions. (Meyer., 2012) identifies 4 key features:

- Help to Decide: predicting a rating for a user for an item

- Help to Compare : rank a list of items in a personalized way for a user

- Help to Discover: provide a user with unknown items that will be appreciated

- Help to Explore: give items similar to a given target item

Most of the applications of recommender systems are on e-commerce websites. The site displays a list of recommended items to the end user.

Resnick and Varian (Resnick et Varian, 1997) give a definition of a recommendation engine as a system able to learn users' preferences about different items and use these preferences to propose new items that users might be interested in.

Herlocker and colleagues (Herlocker et al., 2000) describes a recommender system as one that predicts which items might correspond to the tastes and needs of given users.

As we can see, while Herlocker's description focuses more on the prediction aspect of recommender systems, Resnick and Varian's approach is closer to a real-world recommendation concept.

According to Burke (Burke, 2002), a recommender system must be able to provide individualized recommendations and guide users in a personalized way. Burke's definition adds new notions such as individualization and personalization.

1.2.2 Recommender Systems Function

We previously defined recommender systems as tools and methods providing users with item suggestions they might like to purchase or utilize. In this section, we aim at give a more refined definition by illustrating a wider ranger of the possible uses a RS can be. The first distinction that has to be made is between the user of the RS and the service provider (Ricci, 2002). For instance, a restaurant or hotel recommender system is typically used by an intermediary (e.g TripAdvisor) in order to increase its conversion rate, that is increase the number of people going to a given restaurant, or to sell more hotel rooms.
On the other side, the user's motivations for using a system like TripAdvisor are finding a restaurant or hotel that corresponds to his tastes and needs, increasing thus his satisfaction.

As a matter of fact, there are several reasons why service providers employs recommendation engines:

- **Increase the revenue.** In other words, it means increase the numbers of items that are sold. This function is most likely to be the most important in an industrial context (commercial RS). The goal here is to actually sell more items than there would have been without any recommendations. To meet this objective, the system recommends items that are expected to satisfy the user's tastes and needs. However, we must distinguish between predicting users interests in an item and the probability that users will actually choose/select the recommended item.

- **Increase diversity of items sold.** The aim of this function is to incite users to select items that would remain unknown without recommendation. For example, in the case of a book RS (e.g Amazon bookstore), the service provider wants to be able to sell books from all its catalogue and not only the top 5 most popular ones.

- **Improve the user experience.** If the system works correctly and is designed in a proper manner, it can increase the user satisfaction. Indeed, by receiving interesting, diverse and relevant suggestions, the user will appreciate the experience on the website or application.

- **Understanding users.** Another major function of a RS is to be able to describe users preferences. These preferences may have been collected explicitly or by predicting them. This data might be used by the service provider to better manage its production or stock.

We described the main reasons service providers are using RS. However, users must not be forgotten in the equation. A well designed RS should have a good equilibrium between users and service providers (Burke, 2007).

We expose now the functions users might be interested in when using a RS.

- **Rank a list of items.** This is probably one of the most important function for a RS, that is to provide some good items to the current user, according to the rating predictions. In other words, recommend items that the user should like.

- **Recommend a sequence.** This function is aiming more at fitting the long-term preferences of users. The principle is to generate a coherent sequence of recommendations instead on providing a succession of independent ones. For instance, it would make sense to recommend *Matrix 2 Reloaded* after having recommended *Matrix 1*. (Shani et al., 2005)

- **Provide annotations.** In this case, instead of generating a list of items, we consider an already existing context such as the weekly movie release in movie theaters. The function of the RS would be to emphasize movies that are more likely to correspond to the user's tastes and preferences.

- **Navigation improved.** Given a large catalog, the task of a RS can be to improve the user's browsing experience by helping her finding items that fit her tastes and needs. (Brusilovsky, 1996).

7

1.2.3 Data source

Recommender engines feed themselves with data. They collect different types of data such as very simple and basic data (user ratings/evaluations), more knowledge dependent (ontological description) or social relations and activities of users. No matter what the data source is, we generally identify three entities: items, users and relations between users and items.

In our work, we will use basic data such as ratings (MovieLens, Netflix, Chapter 4) and more knowledge dependent with Vodkaster (Chapter 5), as we will make use of the reviews posted by users (textual content).

The most convenient data is high-quality explicit feedback, which includes explicit input by users regarding their interest in products. For example, Netflix collects star ratings for movies, and TiVo users indicate their preferences for TV shows by pressing thumbs-up and thumbs-down buttons.

Usually, explicit feedback comprises a sparse matrix, since any single user is likely to have rated only a small percentage of possible items.

When explicit feedback is not available, recommender systems can infer user preferences using implicit feedback, which indirectly reflects opinion by observing user behavior including purchase history, browsing history, search patterns, or even mouse movements.

1.3 Recommender system classification

Previous classifications ((Resnick et Varian, 1997),(Schafer et al., 1999),(Schafer et al., 2001)) of recommendation techniques have already been published. We only give here some general information about the main techniques.

1.3.1 Simple and classical topology

The most common way to describe and identify the different types of recommender systems is the following:

1. Collaborative Filtering systems (CF)
2. Content-Based Filtering systems (CBF)
3. Hybrid systems (combination of CF and CBF techniques)

The typical Collaborative Filtering Item-based recommendation (like on Amazon) is entirely based on user-item rating (e.g., a user rated a movie with 4 stars, or a user "likes" a movie).

When we compute the similarity between items, we are not supposed to know anything other than all users' history of ratings. Therefore, the similarity between items is computed based on the ratings instead of the meta data of item content.

On the contrary, the point of Content-Based Filtering is that we have to know the so-called "content" of both user and item. Usually a user-profile and item-profile are created using the content of shared attribute space. For instance, for a movie, we represent it with the movie stars in it and the genres (using a binary coding for example). For user profile, we can do the same thing based on the users likes some movie stars etc. Then the similarity of user and item can be computed using e.g., cosine similarity.

1.3.2 Burke's classification

Burke (Burke, 2002) proposes a very complete classification of existing recommendation techniques by identifying each method's input data and its algorithm used. He defines five types of recommendation techniques:

- Collaborative filtering
- Content-based
- Demographic
- Utility-based
- Knowledge-based

We describe these techniques in the following section (1.4).
Mark van Setten (Setten, 2005) gives another classification for recommendation techniques. He introduced a social-based technique and an information-based technique.

1.4 Recommendation Techniques

Identifying useful items for users is the core function of a recommender system (Adomavicius et Tuzhilin., 2005). In order to predict these, a RS has to be able to predict the utility of these items. Then, based on the results, the system decides which items are recommendable.

1.4.1 Collaborative Filtering

User-user Collaborative Filtering

User-user collaborative filtering is based on the central idea that users who have an interest in same items and similar ratings will thus have similar preferences (Resnick et al., 1994), (Shardanand et Maes, 1995). Given a similar rating behavior, the recommender system is able to predict if a user might be interested in an unseen item. The process of a typical user-user collaborative filtering is generally divided into three steps (Herlocker et al., 2000):

1. **Similarity measurement** The recommender system computes the similarities between the active user and other users who have rated the same items. This step is a well-researched area in the field of recommender systems.

2. **Neighborhood** Similar users are regrouped in a subset namely the « neighborhood » of the active user. Typically, this neighborhood is made of the most similar users computed during step 1.

3. **Prediction and recommendation generation** In the third and final step, the system gathers information from the active user's neighborhood. Then an algorithm generates a list of items to be recommended. In the real world context of e-commerce websites, it is called the « Top-N » recommendation list. Additionally, a rating prediction can be computed for a specific item.

In their survey, Herlocker et al. (Herlocker et al., 2002) present several algorithms for the recommendation generation step. The strongest advantage of user-user collaborative filtering is its domain independency.

Important: Indeed, the fact that similarity between users is calculated by using only rating data makes the system adaptable to any type of products. However, the main flaw of this technique is that the similarity, being computed with rating data only, cannot take into account the reasons that led to a good or bad rating. And thus, two users might have liked the same item, but for some totally different reasons. We present in Chapter 5 a new method that allows a recommender system to include this new dimension.

Item-item Collaborative Filtering

Item-item collaborative filtering is somehow similar to user-user collaborative filtering. and can be considered as the same approach but from an item point of view.

We have the following principle: items that have been rated in the same way are likely to share some similar characteristics, thus users who like one of them should like the others that are similarly rated (Herlocker et J., 2001).

Amazon is probably the most well-known e-commerce website to using item-item collaborative filtering (Linden et al., 2003). Amazon, thanks to its highly effective and well-functioning item-item recommendation technology made a large gain increase. As it also requires only the rating data, item-item collaborative filtering has the same strengths and flaws of user-user collaborative filtering.

1.4.2 Content-based Filtering

Basically, Content-Based approaches build a model (or profile) of users interests based on the characteristics of items they rated. To do so, the system analyze the description of items previously rated. The process of recommendation is essentially finding good matches between the user profile and items features.

This technique presents advantages such as *user independence*, since CBF systems only use ratings of the active user to build her model, as opposed to collaborative filtering techniques, which rely on "neighbors". Additionally, when a new item appears and has not yet been rated, CBF systems are able to recommend it. This is a collaborative filtering well-known issue, namely cold-start or "first-rater" problem.

However, these techniques suffers from over-specialization, as they are not capable of finding unexpected items: the user will receive recommendations of items similar to the ones she rated before. This problem of novelty is also know as the serendipity problem.

Important remark   It is critical to note that the so-called Content-Based techniques are actually not based on the real content of items. For example, in the context of book recommendation, a CBF system does not really rely on the book content. It is only based on the mere description, keywords or abstract in the best case. Most of the time, the "content" is just the genre, author, editor or other metadata. Additionally, those methods do not really take into account the textual content that has been written about items, by users, blogs or whatever. They generally apply semantic analysis by using ontologies or Word Domain Disambiguation or Word Sense Disambiguation. Examples of such attempts to introduce some semantics in the recommendation process can be found in (Magnini et Strapparava, 2001), (Stefani et Strapparava, 1998), (Degemmis et al., 2007), (Semeraro et al., 2009).

These approaches integrate linguistic knowledge in the process of learning user profiles. The main drawback of these techniques is that the linguistic knowledge comes solely and exclusively from the WordNet lexical ontology. Therefore, the so-called "sense-based" profile representation relies and depends on some deterministic concepts: it is not really learnt from users writings about items. Moreover, these methods is language ans domain dependent.

We will see in Chapter 5 how to integrate the *real textual content* dimension with a domain and language independent approach, by using a similarity based on words.

1.4.3 Demographic

Demographic-based system are able to recommend items according to the demographical information of users. This type of system rely on the hypothesis that each demographic class should be recommended differently. It is a simple and effective personalization method. Websites can easily adapt the displaying language according to the user's country or language. Recommendations can take into account the user's age. These techniques have been more studied in the marketing literature than in the RS area.

1.4.4 Knowledge-Based

This type of systems recommends items according to a specific information (knowledge) on the relevance of a given item's features for a given user's needs. In knowledge-based systems, the similarity gives a measure of the matching quality between user's preferences (or needs) and the items to be recommended. In this case, similarity can be assimilated to the utility for user (Bridge et al., 2006).

1.4.5 Statistical Summarization

People generally trust statistics. The most often used statistical summarization techniques are popularity and average. These methods are popular because they are efficient in explaining to users the current system's authority.

Popularity

Popularity-based recommendation techniques relies on a simple idea. Generally, e-commerce platforms show a list top selling items on the homepage. Herlocker (Herlocker et al., 2000) proposed a system that takes into account every users ratings and from this overall average, the top rated items are selected. In fact, people have a tendency to be attracted to popular items during an online shopping session.

Average

A simple and pragmatic technique is using the average combined with popularity. The inner quality and popularity of an item is somehow encapsulated in its average rating.

For instance, the Internet Movie Database (IDMb.com) presents the average rating for each movie. It allows users to have an insight into the general opinion (positive or negative) about a given movie.

On Vodkaster, a french VOD platform and social network for movies, another type of average is presented, which is the average rating of friends. This is both a statistical and social information. It is generally assumed this type of average is very persuasive (Herlocker et al., 2000).

1.4.6 Social Navigation Technique

The social navigation technique relies on the simple principle of showing the activity of others users (e.g « User 1 likes/purchased Item 3 ») (Konstan et Riedl., 2002; Riedl et Konstan, 2002). According to Herlocker et al. (Herlocker et al., 2000), users attach importance to what their similar peers like.

Although Bonhard (Bonhard, 2004) proposed a more complex approach using social networking, a simple social navigation interface improves the user's recommendation experience, as his neighbors are in fact potential recommenders.

More recently, (Pacitti et al., 2011a,b) proposed a social-based P2P recommendation system for large-scale content sharing that leverages content-based and social-based recommendation. The main idea is to recommend high quality documents related to query topics and contents held by useful friends (of friends) of the users, by exploiting friendship networks.

1.4.7 Hybrid Recommender System

The techniques we have presented so far can be combined into a hybrid recommender system. Indeed, by doing so, the weaknesses and shortcomings of each single technique may cancel each other out and thus performance is improved too (Burke, 2002). For instance, Collaborative Filtering techniques have to face the new-item problem, i.e they are not able to recommend items that have not been rated yet. However, new items features (description) are generally available and can be used with content-based methods.

As a matter of fact, a recommender system like the one operating on Amazon combines several recommendation techniques, leading to more efficient and accurate predictions (Linden et al., 2003).

1.5 Data Mining techniques for Recommender Systems

In this section, our goal is to provide an overview of the Data Mining methods used in this thesis.

1.5.1 Data Mining process

Typically, there are 3 steps in a Data Mining process: Data Preprocessing, Data Analysis and Result Interpretation. Figure 1.1 shows the most important methods for each of these steps.
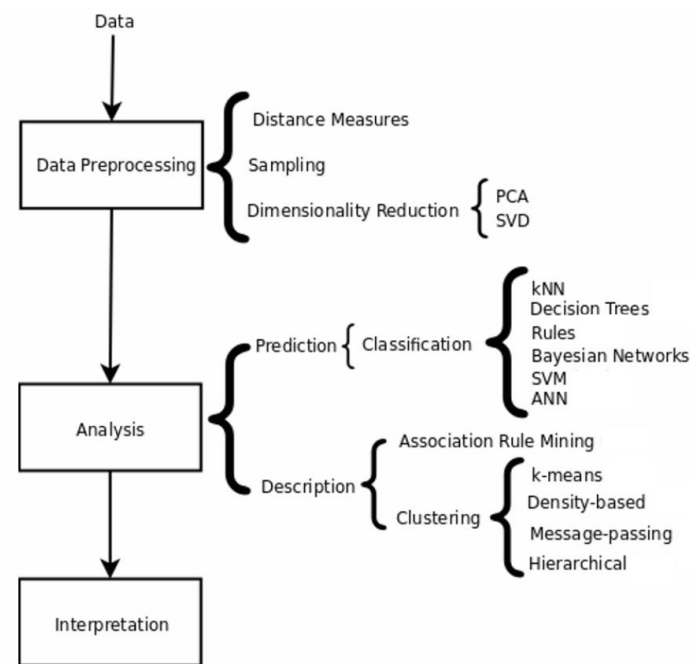
Figure 1.1: *Main methods of Data Mining.*

1.5.2 K-Nearest Neighbor techniques (kNN)

The K-Nearest Neighbor techniques are one of the most commonly used ones in the recommendation area (Adomavicius et Tuzhilin., 2005). Generally based on binary or real-valued data, these approaches use some association rule principles and generalize them to compare objects (e.g items or users). Even if these methods are very well adapted for item-to-item recommendations, they suffer from the lack of scalability. Indeed, the time to search neighbors increases quadratically with the number of elements.

Usually, there are three main components for a collaborative filtering approach using kNN:

- A similarity measure
- A function that retrieves the neighborhood using the similarity measure

- A rating prediction function based on the neighbors' ratings

Depending on what association is used, we can distinguish two *k*NN techniques: item-based and user-based.

_k_NN Item-based

In this approach, the rating prediction of a given user u on a given item i is computed using previous ratings of u on items that are similar to i. (Sarwar et al., 2001). *k*NN item-based approches present a strong advantage in terms of computational complexity. Indeed, as mentioned above, the neighborhood calculation time is proportional to the square of the number of objects to compare. As a matter of fact, the number of users is almost always much larger than the number of items. Therefore, item-based approches are often more efficient. However, one could argue that this is less true on user-generated catalogs such as Youtube, since the number of videos is such that item-item matrices are huge.

The most famous example of *k*NN Item-based approach is Amazon and its famous "people who have bought this item also purchased these items".

Let n be the number of users, m number of items and K the number of neighbors considered. Item-based approaches have a time complexity of $O(m^2 \times n \times K)$ to build the neighborhood models and $O(K)$ for a rating prediction. The space complexity is $O(m \times K)$

_k_NN User-based

In this approach, the rating prediction of a given user u on a given item i is computed using previous ratings of u's neighbors on item i (Shardanand and Maes, 1995). Therefore, the matrice of user-user similarities has to be computed. As before, the neighborhood has to be determined and the set of K nearest neighbors is selected. In the end, the ratings of the neighbors are combined in the rating prediction function.

1.5.3 Matrix Factorization techniques

The matrix factorization approach has become popular with the Netflix Prize, mainly because they allow to deal with large amounts of data, while being quite accurate and fast (Bell et Koren, 2007). These methods are considered to be state-of-the-art for a static rating prediction task. Indeed, the algorithms used in such techniques are easily implemented and quite efficient. This has been verified during the Netflix Challenge. On the other hand it seems to be not adapted to item-to-item recommendation as it is a pure scoring method.

General principles

In its basic form, matrix factorization characterizes both items and users by vectors of factors inferred from item rating patterns. High correspondence between item and user factors leads to

a recommendation. Recommender systems rely on different types of input data, which are often placed in a matrix with one dimension representing users and the other dimension representing items of interest.

Matrix Factorization model

Matrix factorization models are methods that project users and items to a common latent factors space of dimensionality k. In this new space, the interactions between user and item are represented as inner products. Each item i is associated with a vector $q_i \in$, and each user u is associated with a vector $p_u \in$.

To estimate user u's rating of item i, we compute the dot product $q_i^T \dot{p}_u$. which is denoted by $r_{u,i}$:

$$\widehat{r_{u,i}} = q_i^T p_u \tag{1.1}$$

The mapping of each item and user to factor vectors q_i, $p_u \in$ is the difficult part. Once this task is done, the recommender system can use it to predict any $\widehat{r_{u,i}}$, simply by using equation 1.1.

This technique is somehow similar to singular value decomposition (SVD) (Golub et Reinsch, 1970). In Information Retrieval, the SVD approach is often used to identify latent semantic factor. In the context of Collaborative Filtering, it is necessary to factorize the users-items rating matrix. Such matrices are usually incomplete due to missing values and are therefore relatively sparse. If handled without care, sparsity can induce overfitting.

Fill in the matrix and make it dense has been proposed in (Sarwar, 2000). The main issue with this approach is the significant increase in the amount of data and distortion if imputation is not accurate.

(Funk, 2006; Koren, 2008; Patarek, 2007), in more recent works proposed to model the observed ratings only and avoid overfitting by using regularizers in their model. Hence, the factor vectors p_u and q_i are learnt so that the regularized squared error on the set of known ratings is minimized:

$$\min_{q,p} \sum_{(u,i) \in \kappa} (r_{u,i} - q_i^T \dot{p}_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2) \tag{1.2}$$

κ is the set of the pairs (u,i) for which $r_{u,i}$ is known, that is the training set. The model is fitted on the previously observed ratings. As the system has to be able to predict unknown ratings, these previous observations have to be generalized. Hence, the learned parameters have to be regularized. This is the role of λ (constant), that sets the intensity of regularization. Cross-validation is usually employed to determine this constant. In the paper "Probabilistic Matrix Factorization" (Salakhutdinov et Mnih, 2008), Ruslan Salakhutdinov and Andriy Mnih's propose a probabilistic approach for regularization.

Drawbacks

Even if these methods offer a good scalability and predictive accuracy, their classical state-of-the-art versions do no allow Dynamic Adaptation. Note that Dynamic Adaptation (presented in Chapter 4) has to be distinguished from the ability to accept new users. Indeed, Sarwar (Sarwar et al., 2002) presented an incremental singular value decomposition algorithm, that actually computes an approximated decomposition, as space is not orthogonal. This allows to accept new users without having to recompute the model built from existing data. However, the parameters of users and items models are not updated neither adapted.

We propose in Chapter 4 a new approach that allows Dynamic Adaptation, by modifying the models of a matrix factorization technique presented in (Recht et Re, 2011).

1.6   Marketing theory

Recommender systems are an essential feature for modern e-commerce website. And several theories on marketing may have influenced the way recommender systems are designed, especially in terms of explanation interface.

Indeed, the role of the recommender system's explanation interface has gained more and more importance, due to the fact that, without sufficient argumentation / information / explanation, people are less convinced to buy an item.

Mass Customization and Internet Marketing

"If I have 3 million customers on the Web, I should have 3 million stores on the Web." Jeff Bezos, CEO of Amazon.com

In the book "Mass Customization" (Pine, 1993), Joe Pine identified the need for modern vendors to move their attention from "mass production" to "mass customization". And recommender systems are precisely successful at achieving that task.

Even if a recommender system cannot literally customize its items, it can customize the list of products for a given user. Nowadays, customers are not only paying attention to the product's quality but they are also looking for good « user experience ».

The "black box" approach of most recommender systems does not present any satisfactory explanation to their users (Bilgic et Mooney, 2005).

Internet marketing is a nearly related concept that has always been influenced the development and evolution of recommender systems.

We find the same traditional business-to-business (B2B) model, business-to-consumer (B2C) model and the newer peer-to-peer (P2P) model.

Recommender systems is aimed to enhance the Internet marketing return on investment of all three types of business and studies have been looking in this direction, but the role played by the recommender system's explanation interface has not been well-studied yet.

1.7 The Reasons Why to Explain

Recommender system literature has only been focusing on improving the accuracy of recommendations. This is mainly due to the fact that it is generally assumed that improving accuracy improves the user's satisfaction and gives a better promotion effect. As a result, even if recommender systems and collaborative filtering have been in the research focus for the last decade, the « transparency » approach (or « white box » approach) integrating explanation interfaces has only been studied recently, in the 2010's. However, the existing approaches are not consistent enough to be implemented as a recommender system with well-argued explanations and good visualization interfaces.

Herlocker et al. (Herlocker et al., 1999) discuss two types of accuracy metrics: statistical accuracy and decision-support accuracy. Popular measures such as Mean Absolute Error (MAE) (Shardanand et Maes, 1995) and Coverage belong to the former group and so do Recall and Precision (Sarwar, 2000).

The traditional "black box" approach of recommender systems tends to be counter-intuitive. Indeed, asking family or friends for recommendations is a natural and transparent social process.

It has been shown in previous studies on expert systems that justification for experts advice play a major role in the way a system is designed (Buchanan et al., 2005).

Explanation is the link between the human and the system (Johnson et Johnson, 1993) (Koenemann et Belkin, 1996).

More recently, it has been argued (McNee et al., 2006) that most recommender algorithms have almost the same results in terms of accuracy measurements, thus the next step is to improve the human-recommender interaction.

2

A FORMAL FRAMEWORK FOR AUTOMATIC RECOMMENDATION

Contents

Abstract

In this chapter, we provide a general definition of what a recommender system is and what it does. We present the objects involved in these systems: items, users, metadata on users and items. We also list the core functions, main features and describe a formal framework for recommender systems.

2.1 Introduction

In this chapter, we present a formal framework for automatic recommender systems.
We both present classical methods and our contribution to the state of the art.

2.2 Similarity measure

We have deliberately limited our work to simple methods with a view to allow the integration of a fast online real-time adaptation. Similarity measures are the keystone of recommender systems (Ziegler et al., 2005). Resnick (Resnick et Hal., 1997) was one of the first to introduce the Pearson correlation coefficient to derive a similarity measure between two entities. Other similarity measures such as Jaccard and Cosine have been proposed and are standards (Meyer., 2012) (Sarwar et al., 2001).

As mentioned before, S_u denotes the set of items rated by u, T_i the set of users who have rated item i, $r_{u,i}$ the rating of user u on item i and $\overline{r_x}$ the mean of x (user or item).

2.2.1 Pearson

The Pearson similarity between items i and j is computed as follows:

$$Pearson(i,j) = \frac{\sum_{u \in T_i \cap T_j}(r_{u,i} - \overline{r_i})(r_{u,j} - \overline{r_j})}{\sqrt{\left(\sum_{u \in T_i \cap T_j}(r_{u,i} - \overline{r_i})^2\right)\left(\sum_{u \in T_i \cap T_j}(r_{u,j} - \overline{r_j})^2\right)}} \tag{2.1}$$

This similarity measure is often used in the literature (Adomavicius et Tuzhilin., 2005) (Rao et Talwar, 2008). However, there is a problem in the case where two items have only one user in common. Suppose this user gave the same rating to both items, then the similarity will be equal to 1 (the maximum value). Furthermore, it will be the same if a thousand users give the equal ratings to i ad j.

2.2.2 Cosine

The Cosine measure is defined as follows:

$$Cosine(i,j) = \frac{\sum_{u \in T_i \cap T_j} r_{u,i} \times r_{u,j}}{\sqrt{\left(\sum_{u \in T_i \cap T_j} r_{u,i}^2\right)\left(\sum_{u \in T_i \cap T_j} r_{u,j}^2\right)}} \tag{2.2}$$

It suffers from the same bias as Pearson. Moreover, in the case of collinear vectors, the cosine similarity is maximum. In a pure geometrical context, this result can make sense but in our case it does not. This problem as been well identified and described in (Breese et Kadie).

For instance, assume we have two items (*a,b*) and $T_a \cap T_b = (u_1, u_2, u_3, u_4, u_5)$. Suppose we have the following ratings:

	u_1	u_2	u_3	u_4	u_5
Item *a*	1	1	1	1	1
Item *b*	5	5	5	5	5

Table 2.1: Example of failure

A quick calculation leads to the result $Cosine(a, b) = 1$, which is the maximum similarity. Yet, *a* and *b* are totally different. An alternative to this problem is to take the union instead of the intersection, at the denominator. This is true for both Cosine and Pearson formulae.

2.2.3 Jaccard

The Jaccard similarity can be useful to deal with dataset in which only binary events are reported. In those cases, neither Pearson nor Cosine are interesting because they both are equal to zero. Jaccard is also useful to compare two items based on their metadata, if any. It can also be adapted in order to fit other situations, different from binary data.

$$Jaccard(i, j) = \frac{|T_i \cap T_j|}{|T_i \cup T_j|} \tag{2.3}$$

2.2.4 Examples of similarities

The aim of this paragraph is to show the differences between different similarities.
To do so, we take two users *u* and *v* from the Netflix dataset, that have given respectively 204 and 205 ratings. They have 31 items in common. We also have have $\bar{r}_u = 3.81$ and $\bar{r}_v = 3.29$.

items	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}	i_{12}	i_{13}	i_{14}	i_{15}	i_{16}	i_{17}
u	5	3	4	3	5	4	4	5	4	1	5	3	5	4	2	5	5
v	5	5	5	2	4	5	4	4	3	2	4	4	5	5	1	4	4

items	i_{18}	i_{19}	i_{20}	i_{21}	i_{22}	i_{23}	i_{24}	i_{25}	i_{26}	i_{27}	i_{28}	i_{29}	i_{30}	i_{31}
u	5	4	2	5	5	3	1	5	4	5	3	2	3	2
v	5	2	4	4	5	1	3	3	2	3	4	2	2	3

Table 2.2: Example of similarity

Be giving a quick look, we can intuitively guess that these two users are more or less similar, since we do not have many strong disagreement such $r_{u,i} = 5$ and $r_{v,i} = 1$. Wa have calculated their similarity by using different methods:

- Pearson(u,v) = 0.504

- Cosine(u,v) = 0.948

- Jaccard(u,v) = 0.082

- Manhattan(u,v) = 0.745 (MWC)

We can observe that the differences are quite significant. The MWC method seems to find its place in between the Pearson and Cosine method, while Jaccard index is really far below everything. While Pearson and MWC both take into account the average of users, then do not operate similarly, as shown in these results. Indeed, Pearson subtracts the $\bar{r}_u$ to each of its rating and take the square if that difference whereas MWC takes into account the global difference $\bar{r}_u - \bar{r}_v$.

MWC seems to be a good compromise between a too severe similarity such Pearson and a too lax one like Cosine. Moreover, we avoid the mentioned drawbacks of these measures.

2.3 Rating prediction

Let us consider an item i and a user u. Since users are most of the time not allowed to put multiple ratings on the same item, we assume the pair (u, i) is unique. This rule applies to most of social networks or video on demand platforms. Recommend or not item i to a user u amounts to predict the rating $r(u, i)$. Then depending on the rating prediction, the system will decide whether item i is recommendable or not to user u.

The information available to the system are previous ratings of user u and ratings of item i given by other users $v \in T_i$. From that perspective, when trying to predict $r(u, i)$ we first need to look if among users v who rated i before, some have items in common with u. If so, we are able to compute a similarity between u and each of these other users. If not, the similarity will be equal to zero.

Finally, we do a weighted average of previous ratings given by others users, in which the weights are the similarities between u and $v \in T_i$.
Sim stands for some similarity function (as defined in section 2.2) in the following formula.

$$rating(u, i) = \frac{\sum_{v \in T_i} Sim(u, v) \times r_{v,i}}{\sum_{v \in T_i} |Sim(u, v)|} \tag{2.4}$$

This approach is clearly user oriented in the sense that we take into account the similarities between users.

A symmetrical formula $rating(i, u)$ item-oriented can be derived from (6.1):

$$rating(i, u) = \frac{\sum_{j \in S_u} Sim(i, j) \times r_{u,j}}{\sum_{j \in S_u} |Sim(i, j)|} \tag{2.5}$$

To consider both points of view, we do a linear combination.

$$\hat{r}_{u,i} = \beta \times rating(u,i) + (1-\beta) \times rating(i,u) \tag{2.6}$$

In order to balance and somehow correct the prediction, we take into account $\overline{r_u}$ and $\overline{r_i}$. These two averages are combined with two coefficients: m_i for $\overline{r_i}$ and m_u for $\overline{r_u}$., with $m_i + m_u = 1$. We call this new function the weighted rating function:

$$\hat{r}w_{u,i} = \gamma\hat{r}_{u,i} + (1-\gamma)(\alpha\overline{r_u} + (1-\alpha)\overline{r_i}) \tag{2.7}$$

If $\hat{r}_{u,i}$ is not computable, we use a fallback solution relying on $\alpha\overline{r_u} + (1-\alpha)\overline{r_i}$. In the case where i (or u) has no ratings, $\overline{r_i}$ (or $\overline{r_u}$) does not exist. If one of these two averages is missing, we only rely on the other one. We show in section 2.4 how we made further improvements of the rating prediction function, by including "Gaps".

2.4 Gaps

There exist user and item biases in the way that some users have a systematic tendency to give higher ratings than others, and thus some items receive higher ratings than others. In (Bell et al., 2007), the authors propose a solution to encapsulate those effects, with what they call baseline predictors. The method is the following:
Denote by μ the overall average rating. Their baseline prediction for an unknown rating $r(u,i)$ is denoted by b_{ui} and accounts for the user and item effects:

$$b_{ui} = \mu + b_u + b_i \tag{2.8}$$

The parameters b_u and b_i indicate the observed deviations of user u and item i, respectively, from the average. This approach allows to isolate the part of the signal that is really representative if the user-item interaction.

Keeping the same general idea, we have defined a more accurate function, namely the *Gap function*, allowing us to adjust the rating prediction for each user. It is more accurate in the sense that the Gap is not calculated with respect to the overall average. Indeed, it is derived from each respective deviation to the rated items average. It is defined as follows:

$$Gap(u) = \frac{\sum_{i \in S_u}(r_{u,i} - \overline{r_i})}{|Su|} \tag{2.9}$$

As said before, it allows to identify if the user has a certain tendency to give ratings above (more tolerant) or below (more severe) the average. In the end, we use the *Gap function* during the rating prediction process, in order to correct individually (each user has its own gap) the item rating average.

Note that the Gap(u) is updated continuously, as it will be shown in the next chapter. Hence if there is any change in the way a person behaves, the system will take it into account right away. The same holds for items:

$$Gap(i) = \frac{\sum_{u \in S_i}(r_{u,i} - \overline{r_u})}{|Si|} \tag{2.10}$$

There are four possible cases. Note that the Gap function can be negative.

	Highly rated item	Lowly rated item
Tolerant user	$Gap(u) > 0, Gap(i) > 0$	$Gap(u) > 0, Gap(i) < 0$
Severe user	$Gap(u) < 0, Gap(i) > 0$	$Gap(u) < 0, Gap(i) < 0$

Table 2.3: Possible cases in Gaps

For instance, let's take a severe user *Anthony* and a highly rated item *Star Wars: Episode V - The Empire Strikes Back*. By including the gaps, the system will take into account these two pieces of information and thus compensate the rating prediction.

In concrete terms, the Gap function is integrated in the weighted rating function $\hat{r}w_{u,i}$ (formula 2.7). It corrects both averages as follows:

$$m_i\left(\overline{r_i} + Gap(u)\right) + m_u\left(\overline{r_u} + Gap(i)\right) \tag{2.11}$$

The item average is corrected by the $Gap(u)$ and the user average by $Gap(i)$. The reader may ask why $Gap(u)$ should correct the item average and not the user average. By definition, $Gap(i)$ is the "result" or consequence of users who have rated item i. It reveals how item i is generally rated with respect to each user's average rating. Hence, it has to correct precisely the user average m_u in the rating prediction function, and not the item average. The same reasoning holds for $Gap(u)$, it has to correct the item average m_i.

2.5 Automatic tuning through randomness

In this part, we present the method we have applied to optimize jointly several parameters of the system. This is a key point in our system: the tuning process is automatic and random.
A large class of optimization problems can be handled by random search techniques. These methods become competitive in some specific circumstances, for instance when the function characteristics (except possibly function evaluations) are difficult to compute, when there is only limited computer memory available, when the function to be minimized is very "bumpy," when it is highly desirable to find the global minimum of a function having many local minima, etc. Random search techniques were first proposed by Anderson (Anderson, 1953) and later by Rastrigin (Rastrigin, 1963) and Karnopp (Karnopp, 1963).

2.5.1 Definition

Basically, random optimization algorithms work by iteratively moving to better positions in the search-field which are sampled using for instance a normal distribution near the current position (Sarma, 1990). We have modified this classic principle and adapted it to our recommender system. Note that we do not adjust each parameter by intuition nor based on observations. The idea is to combine K best random sets of parameters when predicting a rating. This way, we will take benefit from these multiple points of view.

The process of this optimization is the following:

1. Run n experiences with a set of random parameters (judge).

2. Evaluate each judge on the development set, using the metric to be optimized (RMSE,...).

3. Select the K best judges.

4. Run on the Test set

5. Combine, for each pair (u, i), the ratings predicted by the K judges.

6. Evaluate.

In the method described above, we combine the K rating predictions $r_k(u, i)$ by doing a simple average. In order to be more accurate, we employ a weighted average, i.e we give more weight to the best judges. These weights W_k are directly proportional to the score of the associated judge. Therefore, the final rating prediction is calculated as follows:

$$\hat{r}_{u,i} = \frac{\sum_k r_k(u, i) \times W_k}{\sum_k W_k} \tag{2.12}$$

2.5.2 Specific judges

We describe here a more sophisticated approach, in which we select specific judges well-suited for each part of the user population, and/or each set of items. To do so, we have first divided the users and items population according to some indicator. This indicator could be extracted from metadata, such as age, gender, country and so on. Unfortunately, it is generally difficult to have access to such data in a academical context. Hence, we decided to split the population of users and items with respect to variance, because it reflects rather well the users behavior.

2.5.3 Example

We present here some results of optimization through randomness on the Netflix dataset. In this example, we have performed 10M experiences, each one of them corresponding to a set of randomly chosen parameters. When then sorted them and keep the top 10.

	RMSE	MAE	α	β	γ
Judge 1	0.919	0.713	0.609	0.315	0.085
Judge 2	0.919	0.715	0.123	0.505	0.068
Judge 3	0.919	0.716	0.094	0.434	0.162
Judge 4	0.919	0.715	0.489	0.075	0.093
Judge 5	0.919	0.715	0.234	0.176	0.080
Judge 6	0.919	0.714	0.670	0.536	0.071
Judge 7	0.919	0.715	0.175	0.555	0.164
Judge 8	0.920	0.714	0.374	0.619	0.185
Judge 9	0.920	0.716	0.560	0.848	0.105
Judge 10	0.920	0.713	0.604	0.388	0.074

Table 2.4: Example of judges on Netflix

We observe that depending on the judge, the relationships between the parameters can change significantly. Recall that:

- α is the weight of $\bar{r}_u$, $(1 - \alpha)$ for $\bar{r}_i$

- β is the weight of $rating_{u,i}$, $(1 - \beta)$ for $rating_{i,u}$

- γ is the weight of the similarity based model, $(1 - \gamma)$ for the average model

The first observation is that for all of the judges, γ is rather small (approx. 0.1). That is to say, most of the models selected here relies on the average (or baseline) predictors.

If we look at judges 3 and 6, we observe that there is a big difference for α. Indeed, we have $\alpha_3 = 0.094$ and $\alpha_6 = 0.67$. It means that judge 3 uses almost 100% of $\bar{r}_i$ in its baseline predictor. He relies only on the average of items and doesn't take into account the users' average. On the contrary, judge 6 gives more weight to the average of users (67%).

Judges 4 and 9 are also very different, if we look at parameter β. We have the following: $\beta_4 = 0.075$ and $\beta_9 = 0.848$. It means that in the similarity based model, judge 4 relies mostly on $rating_{i,u}$ whereas judge 9 relies mostly on $rating_{u,i}$.

Our point here is that by combining these very different points of view, we are able to reduces the localness of our optimum. Therefore, when applying the same judges on the test dataset, we will have more stable results than a best unique selected judge.

2.6 Evaluation metrics

Performance evaluation of recommender systems is a major issue.

It should be noted that we are not able to make online experiments. Therefore, we can not measure the feedback on our recommendations. Therefore, we had to use the classical machine learning test protocol, in which the predicted ratings are compared to the real ones.

The literature mainly focuses on improving recommendation accuracy by proposing new algorithms or new techniques.

Prediction accuracy measures the nearness of the system predictions to the users actual real ratings, by simply comparing the estimated value $\hat{r}_{u,i}$ for a given pair (u, i), with the true rating $r_{u,i}$. Many measures of prediction accuracy have been proposed in the past, and several authors have made proposal about what should be used when comparing the accuracy of methods applied to univariate time series data (J.E. Hanke, 1995) (B.L. Bowerman, 2004). Alternative metrics have been proposed in addition to accuracy such as diversity, coverage, novelty and many others.

We have used different types of metrics: the classical metric *Root Mean Square Error* (RMSE), *Mean Absolute Error* (MAE), coverage, categorization accuracy and *Mean Average Precision* (MAP) (Said et al., 2013).

The MAP is an interesting metric as it takes into account the users preferences. We define these different metrics in the following.

2.6.1 Prediction-based metrics

Predictive metrics aim at comparing the predicted values against the actual values. The result is the average over the deviations.

Root Mean Square Error (RMSE)

This measure namely *Root Mean Square Error* is often used to evaluate different methods applied in RS. It It also has become popular in the recommender system field with the Netflix Challenge (Bell et al., 2007). Let R be the set of the predicted ratings, the RMSE is defined as follows :

$$RMSE = \sqrt{\frac{1}{|R|} \sum_{(u,i,r) \in R} (\hat{r}_{u,i} - r_{u,i})^2} \qquad (2.13)$$

It is widely assumed that reducing the RMSE amounts to increase the relevance and precision of the recommendations (Su et Khoshgoftaar., 2009). However, one could argue that this measure suffers from a discontinuous behavior. Indeed, by definition the RMSE gives more weight to errors greater than one and less to the ones less than one.

Mean Absolute Error (MAE)

The Mean Absolute Error (MAE) formula is the following:

$$MAE = \frac{1}{|R|} \sum_{(u,i,r) \in R} |\hat{r}_{u,i} - r_{u,i}| \tag{2.14}$$

Unlike the RMSE, this metric is even. And thus could also be criticized for being too lax on large errors. One could take benefit from mixing MAE for errors less than 1 and the RMSE for others.

Mean Absolute Percentage Error (MAPE)

The mean absolute percentage error (MAPE), also known as mean absolute percentage deviation (MAPD), is a measure of accuracy of a method for constructing fitted time series values in statistics, specifically in trend estimation. It usually expresses accuracy as a percentage, and is defined by the formula:

$$\text{MAPE} = \frac{1}{n} \sum_{t=1}^{n} \left| \frac{\hat{r}_{u,i} - r_{u,i}}{r_{u,i}} \right| \tag{2.15}$$

where $r_{u,i}$ is the actual value and $\hat{r}_{u,i}$ is the predicted value.

The difference between $r_{u,i}$ and $\hat{r}_{u,i}$ is divided by the Actual value $r_{u,i}$ again. The absolute value in this calculation is summed for every fitted or forecasted point in time and divided again by the number of fitted points n. Multiplying by 100 makes it a percentage error.

2.6.2 Decision-based metrics

Decision-based metrics evaluates the top-N recommendations for a user. Usually recommendations are a ranked list of items, ordered by decreasing relevance. Yet, the decision-based metrics do not take into account the position -or rank- of the item in the result list.

There are four different cases to take into account:

- True positive (TP). The system recommends an item the user is interested in.

- False positive (FP). The system recommends an item the user is not interested in.

- True negative (TN). The system does not recommend an item the user is not interested in.

- False negative (FN). The system does not recommend an item the user is interested in.

	Relevant	Not relevant
Recommended	True positive	False positive
Not recommended	False negative	True negative

Table 2.5: Confusion matrix

Precision (P) and recall (R) are obtained from the 2x2 contingency table (or confusion matrix) shown in Table 2.5.

These measures need a threshold t defining what is recommendable and what is not. In the remainder, we call it the recommendability threshold.
We could have considered a neutral class. But in fact, we are only interested by the positive class since only this one will be used in the MRR measure which is described in the next section.

Precision

Precision measures the fraction of relevant items over the recommended ones.

$$Precision = \frac{TP}{TP + FP} \tag{2.16}$$

Precision can also be evaluated at a given cut-off rank, considering only the top–n recommendations. This measure is called precision–at–n or P@n. When evaluating the top–n results of a recommender system, it is quite common to use this measure:

$$Precision = \frac{|hitset|}{N} \tag{2.17}$$

where $|hitset| = |test \cap topN|$.

Recall

Recall measures the coverage of the recommended items, and is defined as:

$$Recall = \frac{TP}{TP + FN} \tag{2.18}$$

Again, when evaluating the top–N results of a recommender system, one can use this measure:

$$Recall = \frac{|hitset|}{|test|} \tag{2.19}$$

F-measure

F–measure combines P and R results, using the weighted harmonic mean. The general formula (for a non-negative real beta value) is:

$$F_\beta = \frac{(1 + \beta^2) \cdot (\text{precision} \cdot \text{recall})}{(\beta^2 \cdot \text{precision} + \text{recall})} \tag{2.20}$$

Two common F–measures are F_1 and F_2. In F_1 recall and precision are evenly weighted, whilst F_2 weights recall twice as much as precision.

The main drawback of the decision–based metrics is that do not take into account the ranking of the recommended items. Thus, an item at top–1 has the same relevance as an item at top–20. To avoid this limitation, we can use rank–based metrics.

2.6.3 Rank-based metrics

We need to be careful with these metrics as it makes no sense to compute them if a user appears only once in the test set. Additionally, since the ground truth data can have ties, rank-based metrics are sometimes hard to interpret.
As a matter of fact, if we get rid of the cases mentioned above, only 1000 users are eligible to these metrics among 20,000 on the Vodkaster dataset. Which is very few. Hence, rank-based metrics have to be interpreted with precautions.

Spearman's ρ

Spearman's ρ computes the rank–based Pearson correlation of two ranked lists. It compares the predicted list with the user preferences (e.g. the ground truth data), and it is defined as:

$$\rho = \frac{1}{n_u} \frac{\sum_i (r_{ui} - \bar{r})(\hat{r}_{ui} - \bar{\hat{r}})}{\sigma(r)\sigma(\hat{r})} \tag{2.21}$$

Kendall–τ

Kendall–τ also compares the recommended (topN) list with the user's preferred list of items. Kendall–τ rank correlation coefficient is defined as:

$$\tau = \frac{C^+ - C^-}{\frac{1}{2}N(N-1)} \tag{2.22}$$

where C^+ is the number of concordant pairs, and C^- is the number of discordant pairs in the data set.

Mean reciprocal Rank

Mean Reciprocal Rank (MRR) is defined as:

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{Q} \frac{1}{\text{rank}_i} \tag{2.23}$$

Recommendations that occur earlier in the top–n list are weighted higher than those that occur later in the list.

Mean Average Precision

Mean Average Precision (MAP) is defined as:

$$\text{MAP} = \frac{\sum_{q=1}^{Q} AP(q)}{Q} \tag{2.24}$$

where Q is the number of queries, and Average Precision (AP) equals:

$$\text{AP} = \frac{\sum_{k=1}^{n} (P(k) \times rel(k))}{\text{number of relevant documents}} \tag{2.25}$$

where $P(k)$ is Precision at top-k, and $rel(k)$ is an indicator function equaling 1 if the item at rank is a relevant document, and zero otherwise.

Recommendations that occur earlier in the top–n list are weighted higher than those that occur later in the list.

2.6.4   Choice of metrics

Spearman and Kendall–τ are not very easily readable because they do not handle ties. In the context of recommendation, it is very frequent to have identical ratings due to the discrete nature of the scale (usually integers from 1 to 5, included). For that reason, we decided to put these two metrics aside.

The Mean Reciprocal Rank is a good rank-based measure, but it does not consider the order of the predicted list of elements and their rank. Hence, as we want to be able to evaluate the ability of our system to order and rank a list of items, MRR is not useful and will not be used.
The Mean Average Precision, however, does take into account the order and rank of elements in the predicted list. The MAP will be the only rank-based metric in our evaluation protocol.

2.7   Experiments

This work has been carried out in partnership with the website Vodkaster [1], a Cinema social network. To be able to compare our results with others, we also have tested our system on the famous dataset extracted from the video on demand platform Netflix and on the public dataset MovieLens.

---

[1] www.vodkaster.com

2.7.1 Datasets

We evaluated our system with three different datasets, Netflix, Vodkaster and MovieLens. Netflix and MovieLens have the advantage to be public and comparable to other techniques. Vodkaster allows to work on the semantic level since the dataset contains not only ratings but also textual comments. Like Netflix and MovieLens, users rate and comment movies.

For the Netflix and MovieLens data, the log is structured as follows $userID, itemID, rating, date$. For Vodkaster, it includes the text or review about the item. Moreover, users and items are not identified by a numerical ID but by username and title. The log for Vodkaster is then: $userID, itemID, rating, review, date$.

Unlike on Netflix and MovieLens, there was no recommender system on Vodkaster. Therefore, the distribution of pairs (u, i) could be more neutral on Vodkaster. Indeed, the fact that a recommender system is operating on Netflix and MovieLens could have had an influence on the users behavior. Still, we can fairly assume that the impact on the users tastes is negligible.

Vodkaster

The corpus has been extracted from Vodkaster's database in May 2014. Users post *micro-reviews* (MR) to express their opinion on a movie and rate it, within a 140 characters Twitter-like length limit. At this date, the corpus was containing about 200k MR and 2M ratings. We divided the corpus into three parts, chronologically sorted: training (Tr), development (D) and test (T). Note that in our experiments, the date is taken into account since we also work on Dynamic Adaptation.

	Tr	D	Tr+D	T	Total
Size	2M	20k	2,02M	20k	2,04M
Nb of Films	26 097	5520	26 248	5683	26 344
Nb of Users	19922	1426	2041	1412	20213

Table 2.6: Statistics on Vodkaster dataset

Netflix

The dataset used for the Netflix Prize is very large (100M ratings, 400k users, 17k movies). To be able to compare results obtained on Vodkaster and MovieLens datasets, we took a subset of the whole Netflix dataset. We chose to keep the data of year 2005, because it has a higher rating density. The dataset was still too large, so we chose to keep 6000 random users with at least 50 ratings. We have then split the data as follows: Training set, Development set and Test set. We

have respected the chronological order of the ratings.

	Tr	D	Tr+D	T	Total
Size	1,3M	20k	1,32M	20k	1,5M
Nb of Films	15345	4416	15419	4562	15497
Nb of Users	6000	6000	6000	6000	6000

Table 2.7: *Statistics on Netflix dataset*

MovieLens

This dataset contains 1M anonymous ratings of approximately 3,900 movies made by 6040 MovieLens users who joined MovieLens in 2000. Each user has at least 20 ratings. Again here, the chronological order is respected.

	Tr	D	Tr+D	T	Total
Size	980k	10k	990k	10k	1M
Nb of Films	3701	2503	3703	2432	3706
Nb of Users	6039	356	6039	348	6040

Table 2.8: *Statistics on MovieLens dataset*

2.7.2 Cold-start simulation

To observe explicitly the effect of Dynamic Adaptation, we need two conditions. The first one is that both training and test sets are chronologically sorted, from older ratings to more recent. The second one is to encounter cold-starts during the test phase, for users and items, or both.

The Dynamic Adaptation makes really sense on real data. That is, data that have not been modified at all. The only real world and full dataset we have is the one from Vodkaster. The MovieLens and Netflix dataset are only partial and have been modified (and/or filtered). We'll see in the next chapter that the impact of Dynamic Adaptation is much greater on Vodkaster (full real data).

This being said, we do not need to intensify artificially the cold-start on Vodkaster since the dataset has not been modified whatsoever and thus reflects a real world situation. Hence, we have decided to take the number of cold starts on Vodkaster as a reference, that is 178 cold-starts for users and 114 for items. The extreme case where a new user rates a new item occurs only 9 times on the whole dataset.

After treatment, we introduced the same proportion (1%) of cold-starts on the MovieLens and Netflix datasets.

2.7.3 Rating distribution

Figure 2.1 shows that globally the rating distributions on MovieLens, Netflix and Vodkaster are rather similar (4 stars being the most common rating). However, we can notice that Vodkaster users are less likely to give a 5 stars rating. They are also more likely to give 1 and 2 stars rating. This makes sense as Vodkaster users are movie-buffs and therefore more severe when they evaluate a movie.
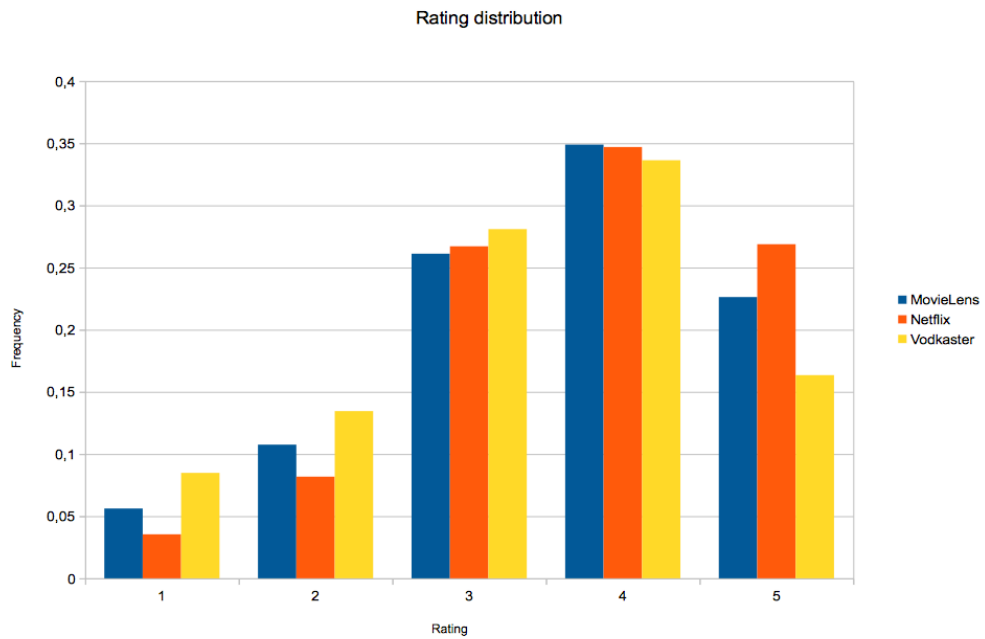

Figure 2.1: *Rating distribution*

CHAPTER

3

A SOCIOLOGICAL STUDY OF USERS

Contents

Abstract

The goal of this chapter is to provide a sociological study of Vodkaster and MovieLens users. We try to define the main typologies of users by studying different variables such as age, occupation or the way users express themselves about movies. We have performed several experiments and interviewed Vodkaster users online. Ultimately, the aim of this work is to incorporate a sociological dimension in the way we approach the recommendation problem.

3.1 Introduction

In this chapter, we present a sociological study of MovieLens and Vodkaster users.
The aim is to have a better understanding of how users behave and identify typologies. Eventually, the results of this study will be taken into account in our approach to automatic recommendation.

These two datasets complete each others as the MovieLens dataset provides demographical information about users while the Vodkaster dataset contains textual reviews associated with ratings. Therefore, on one side we have quantitative data (MovieLens) and on the other side, we have qualitative data (Vodkaster).
Note that is the sociological research area, it is very rare to have and work with large amounts of qualitative data such as we do on Vodkaster.

3.2 General observations

3.2.1 Evolution of users average rating as a function of time

In this graph (Figure 3.1), the time is represented implicitly by and relatively to the order of ratings. We compute the average n-*th* rating for each user and plot it as a function of *n*. Note that these ratings are ordered in the chronological order so we can clearly see the evolution in time.

We observe that the first ratings are higher than the rest. This is observable on average for every user.
We can thus assume that users have a certain tendency to rate movies they appreciate first.
This allow us to introduce the notion of Pantheon movies. That is, great movies, not necessarily the greatest movies, for which users have a personal connection to and love so much they define the users tastes. We give some insight about what movies are generally rated first. In their book *Les films de campus, l'Université au cinéma*, Emmanuel Ethis and Damien Malinas tell us that, as we reach adulthood and go to the university, we carry with us the movies that we loved as teenagers and we look back at them with a sense of nostalgia. They are the movies we grew up with, that helped us build ourselves and that we learn to love. We can then assume that the movies that we rate first are those particular movies, as they are the ones we constantly carry with ourselves, wherever we go.
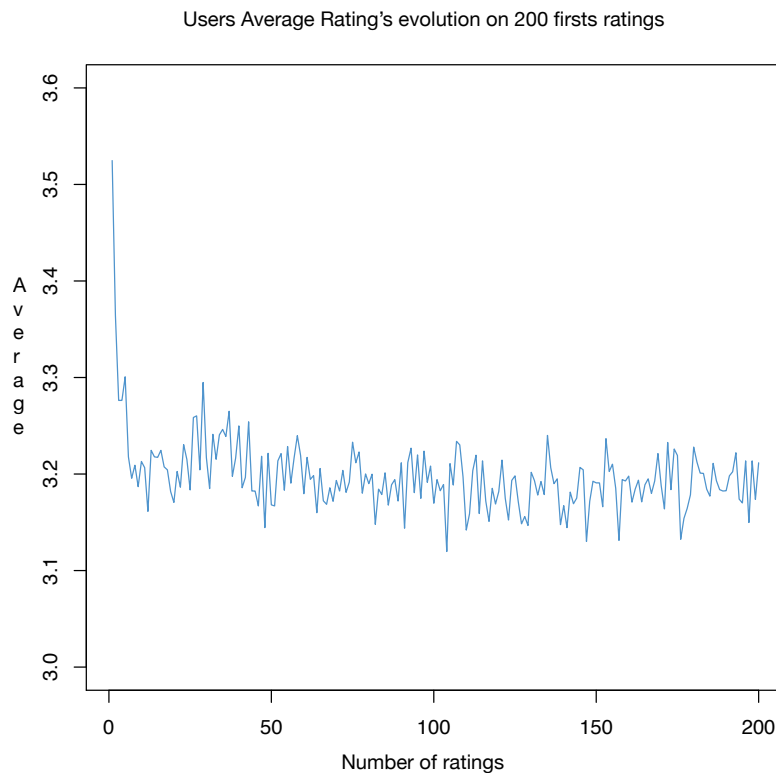
Figure 3.1: *Evolution of users average rating as a function of time - Vodkaster dataset*

3.2.2 Evolution of users behavior as a function of the number of movies rated

We observe in Figure 3.2 the evolution of the users severity (average rating), as a function of the number of movies they have seen. The global trend is a decrease of the average rating as users rate more movies. This observation shows that, the more users watch movies, the more the become severe and critical.

In the next paragraph, Figure 3.3 supports our hypothesis. Before going further, we need to introduce the notion of "gap".
The gap is an individual parameter defined as the average deviation of a user with respect to the average rating of the movies she rates. Hence, the gap forms a good indicator to situate a user with respect to the average appreciation of movies.

Note that a positive gap (greater than 0) is the reflection of a person who globally gives ratings higher than the average rating of items she rates. On the contrary, a person with a negative gap (less than 0) globally gives ratings lower than the average rating of items she rates.

Users average rating as a function of the number of ratings
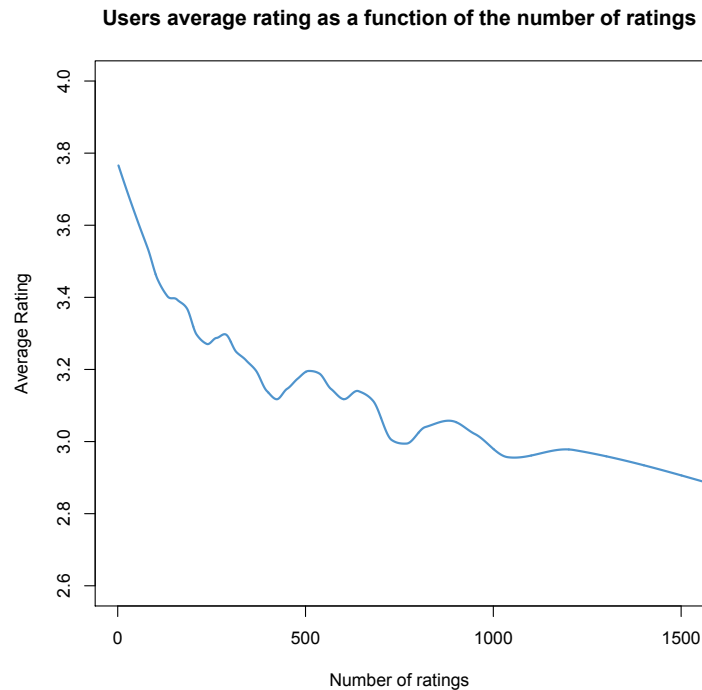
Figure 3.2: Evolution of user average rating as a function of the number of movies rated - Vodkaster dataset

The following graph (Figure 3.3) displays the gap evolution as a function of the number of movies rated.
Note that this is the average gap, calculated for all users. Therefore, we observe a global behavior.

We can notice a main tendency in Figure 3.3. Indeed, as people rate more movies, their gap becomes negative and decreases (dark blue area).
However, this tendency does not seem linear. For instance, when users have rated approximately 350 400 movies, the gap drops rather dramatically and passes through the zero line. It means people are getting more severe.

This phenomena can be interpreted as some kind of maturation or evolution process during which the critique sense and value judgment change. Additionally, this standpoint might reveal the fact that people want to distinguish themselves from the others.

It is also interesting to observe an increase right after the drop. One could assume that people have found their tastes and thus select movies accordingly.
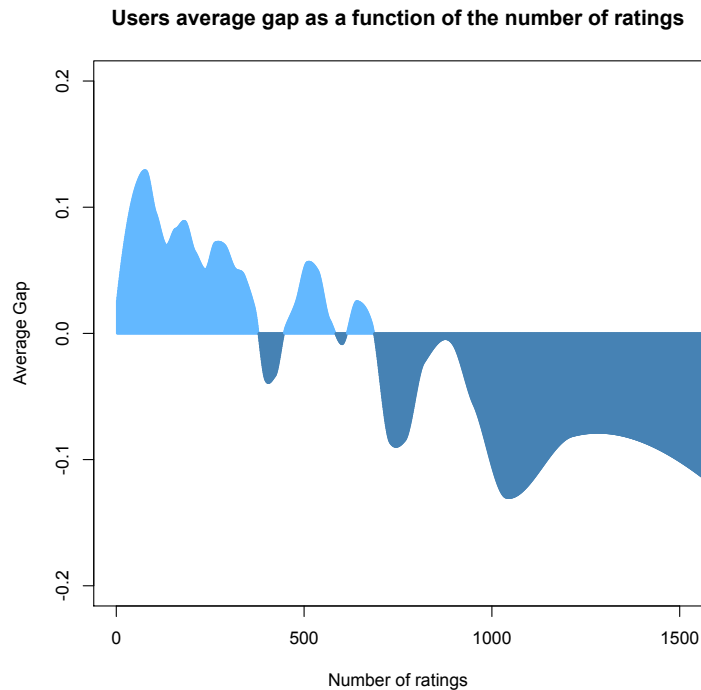
Users average gap as a function of the number of ratings

Figure 3.3: *Evolution of user gap as a function of the number of movies rated - Vodkaster dataset*

3.3 The MovieLens case

In this section, we aim to study the MovieLens users with respects to their demographics.

3.3.1 Rating distribution as a function of age

We present here the rating distribution for each age range. The 18-24 years old population seems to be more "critical" and severe than the average.

As mentioned before, the 18-24 category is the most severe. We can observe a tendency of giving higher ratings as users grow older. We could make two assumptions. Maybe users become more and more tolerant as they become older. They have lowered their expectations. Or maybe they know more and more their tastes and do not take too many risks. This phenomena could also be due to a combination of these two hypothesis.

The 18-24 category seems to be an exploration phase. The individuals might be seeking their tastes as they are building their cultural personality. As a result, they tend to assert their cultural preferences without being moderate. Their young cultural age makes them being in a paradigm where they either love or hate a movie because they're still trying to define who they are. And

Age / Rating distribution (%)	1	2	3	4	5	Avg	# users
UNDER 18	8.17	10.79	23.11	32.38	25.55	3.56	222
18-24	**7.14**	**12.01**	**25.91**	**32.80**	**22.14**	**3.51**	**1103**
25-34	6.02	11.29	26.31	34.60	21.79	3.55	2095
35-44	4.55	10.16	26.59	36.16	22.54	3.62	1193
45-49	4.07	10.04	26.66	36.25	22.98	3.64	550
50-55	4.11	8.25	25.45	36.39	25.80	3.72	496
56+	**4.02**	**7.76**	**23.67**	**36.76**	**27.79**	**3.77**	**380**

Figure 3.4: *Rating distribution as a function of age - MovieLens dataset*

at the beginning of the cultural construction that they go through, they need those radical notes as it helps them shape who they are.
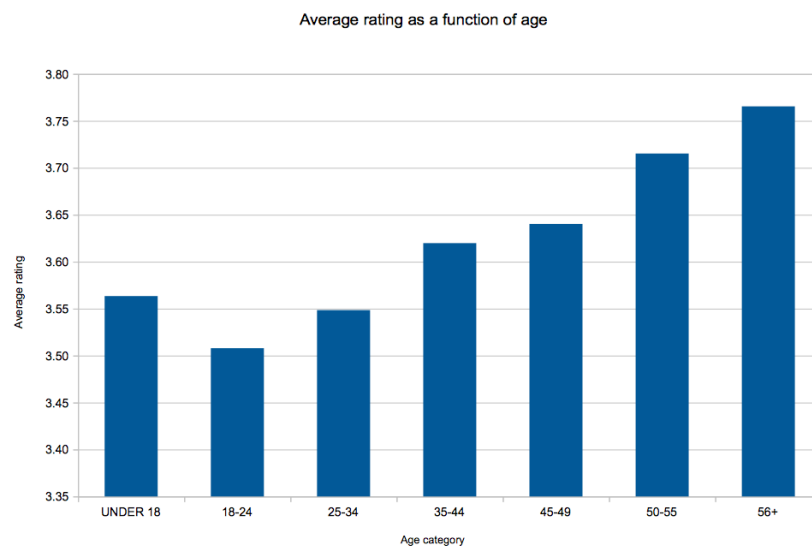

Figure 3.5: *(Histogram) Rating distribution as a function of age - MovieLens dataset*

3.4 The Vodkaster case

After a rather quantitative study of the data provided by MovieLens and the general observations we concluded from it, we have decided to focus on the users of Vodkaster.

3.4.1 Questionnaire results

We administered an online questionnaire and gathered 194 responses. The complete results are available in Appendix A. We can see that 53% of people that responded are between the age of 18 and 25, 29% are between the age of 25 and 35, 7% are under 18 and 10% are above 35 years old. Therefore, 90% of the people that answered are under the age of 36, and 67% of them are under the age of 25. Hence, we can assume from that sample that the vast majority of Vodkaster users are still in a phase of constructing their cultural identity.

3.4.2 Identity construction

The fact that people construct their cultural identity is even more important when we talk about Vodkaster users. Indeed, Vodkaster is a social media on which we can create a profile and other people ("friends" or "connections") are able to see the rating we give to and what we write about movies.
Thus, we also build our cultural identity relatively to others. This is why it is interesting to incorporate the Gap whenever we consider a user. We have presented in Chapter 2 how we integrate the gap in our models.

The social aspect of our cultural and general identity construction is extremely important: we do not necessarily like the same movies depending on the context. That is, we do not show the same face of ourselves when we are alone and when people are watching.

This brings us to the idea of « guilty pleasure ». The whole point of having a « guilty pleasure » is that it is guilty. But why is it guilty? It is because of the image this movie has in our social group. Guilty movies are not the same in every group and we are all subject, especially in the 18-24 age category, of what one could call a cultural peer pressure. Therefore, if we go back to the context of recommendation, we should take into account that our main data source, that is users rating history, might be somehow biased by their cultural peer pressure.

3.4.3 Principle of distinction

Another way of analyzing these data would be through the principle of distinction defined by Pierre Bourdieu in his work called *La distinction. Critique sociale du jugement* (1979). Bourdieu writes that our cultural choices are a social production and reveal our social status but also where we want to stand socially.

Hence, according to the principle of distinction, to achieve a higher social rank, or to aspire to a higher social rank, we need to distinguish ourselves from what we perceive as the lower

class. To do so, we need to inform our friends, through social media (Facebook via Vodkaster for instance), that we do not like movies that we consider to be not worthy enough of our social rank. Or on the other hand, that we like some movies that we consider worthy enough for us.

The principle of distinction might be a way to understand the evolution of the gap in users ratings, associated with the fact that Vodkaster is a social media, hence a social space, in which a latent hierarchy in social groups is created. Looking at the comments under a micro-reviews shows that some are agreeing and some are disagreeing and an actual discussion between members of two different parties takes place. In these discussions, people often refer to the other movies that the person they are arguing with liked and disliked, creating a social hierarchy, in this social space that is Vodkaster. Even if this is based solely of movies, the whole process of rating and posting reviews reveals a deeper understanding our social class.

3.4.4 Observations on words

Most common words (4-grams) used for the first movie rated and reviewed

We have looked the most common expressions and phrases used when users post their first review. Results are shown in Table 3.1

High ratings	Low ratings
" du début à la fin "	" Casse pas trois pattes à un canard "
" une mise en scène "	" Dommage que la mise en scène "
" le meilleur film de "	" Ne suffit pas à faire un "
" un très bon film "	" Moins bon de la saga "
" à voir et à revoir "	" Du début à la fin "
" le film le plus "	" Est au cinéma ce que "
" à couper le souffle "	" Mais il lui manque un "
" au sommet de son art "	" Plus c'est long moins c'est bon "
" un des meilleurs films "	
" ne laisse pas indifférent "	
" une ode à la "	
" dans le rôle de "	
" de tous les temps "	
" dans toute sa splendeur "	

Table 3.1: Most common phrases employed on first reviews (sample)

We observe that users are mainly focused on describing aspects of the movie such as scenario, special effects, direction, characters rather than the feeling they had while watching the movie. However, the way users speak about movies does not seem to be formatted, in the sense that each user has its own style and cinephilic vocabulary. Therefore, we will have to take this fact into account when using words and phrases to build users profiles. Using pre-determined targets or concepts would be too restrictive as the diversity and richness in vocabulary is so

important.

Usage of the pronoun "je" (first person)

In Table 3.2, we observe that most of the time, "je" is followed by "ne" which is the negation in French. Users have thus tendency to describe themselves as what they are NOT. This is somehow coherent with the principle of distinction (see 3.4.3), in which a user defines himself with respect to his differences with others.

Phrase	Frequency (%)
je ne sais pas	9.46
je ne suis pas	7.03
je ne comprends pas	3.57
je ne peux pas	2.48
je ne me suis	2.33
je crois que je	2.12
je m'attendais à un	2.12
je n'arrive pas à	1.96
je ne vois pas	1.96

Table 3.2: Most common phrases starting with first person. Frequency is computed with respect to all phrases starting with "je".

These observations show the importance of negation in the way users express their opinions.

3.5 Conclusions

We have made several observations on the users of MovieLens and Vodkaster. We have discovered that users had to be considered as dynamic object, that evolve with time. We will see how to integrate the user's dynamics in Chapter 4. We have also seen that there was a bias induced by the image users want to promote about themselves, and therefore we will try to take that into account too. We also observed that the way users express themselves in reviews is very rich and diverse, therefore the analysis, processing and use of these textual data can not be reduced to some fixed, pre-determined targets.

CHAPTER

4

DYNAMIC ADAPTATION

Contents

Abstract

In this chapter, we describe the dynamic and adaptive framework. We present a method based on adaptation in real time ("on the fly") providing recommendations in phase with the very present instant. The system includes a Dynamic Adaptation to enhance the accuracy of rating predictions by applying a new similarity measure. We did several experiments on real world data, showing that systems incorporating Dynamic Adaptation improve significantly the quality of recommendations compared to static ones (static matrix factorization for instance).

4.1   Introduction

Recommender systems are designed to suggest appropriate items to users from a large catalog of products. Those systems are individually adapted by using a profile for each user, itself made upon an analysis of past ratings. The most common techniques used in automatic recommendation are Content-Based Filtering (CBF) and Collaborative Filtering (CF). Hybrid systems combine collaborative and content-based techniques, thus taking advantages from both methods.

The fact that item perception and user tastes and moods vary over time is well known. Still, most recommender systems fail to offer the right level of "reactivity" that users are expecting, i.e. the ability to detect and to integrate changes in needs, preferences, popularity, etc. Suggesting a movie a week after its release might be too late (**?**). In the same vein, it could take only a few ratings to make an item go from *not advisable* to *advisable*, or the other way around.

One of the motivations of this work was based on the observation of the dramatic drop in performance when going from random train/test splits as in a standard cross-validation setting towards a strict temporal split. For instance, the difference in rating prediction accuracy as measured by the RMSE (Root Mean Squared Error) exceeds 5% (absolute) when using the famous MovieLens (1M ratings) data set. Another motivation was to ensure the efficiency and the scalability of the algorithms, to respect the real-time constraints on very large recommendation platforms, so that we excluded from our scope some approaches based on Bayesian, probabilistic inference methods (e.g. those based on probabilistic matrix factorizations and non-linear Kalman filters).

In this chapter, we propose a "'reactivity" mechanism in the similarity-based approach to Collaborative Filtering, which updates the similarity measures between users and between items with some form of forgetting factor, allowing to decrease the importance of old ratings. We also propose an Adaptive Matrix Completion method that makes the system very flexible with respect to dynamic behaviors. The factor matrices are dynamically and continuously updated, in order to provide recommendations in phase with the very recent past. It should be noted that the method is truly adaptive and not only incremental, in the sense that it could give more weight to recent data – and not uniform weights to all observations – if this is needed. We are considering the case where no other information than the a set of <user, item, ratings> tuples is given and, consequently, we are not addressing the "(strictly) cold start" problem, where a completely new user or a new item is appearing, with no associated information. The method's principle is that, when receiving a new observation (<user, item, rating> tuple), we update the corresponding entries (rows and columns) of the factor matrices, controlling the trade-off between fitting as close as possible to the new observation and being smooth and consistent with respect to the previous entries. This gives raise to some least-squares problem with temporal regularization, coupling the update of both users- and items-related factors. We will show that the problem could be solved by a simple iterative algorithm (requiring the inversion of a $K \times K$

matrix, where K is the reduced rank in the matrix factorization), converging in a few iterations (typically 2 or 3), so that it could easily update the models even with a rating rate of several thousands ratings per second.

In section 4.2, we present previous works that take into account the temporal and dynamic aspects of recommender systems environment into account. Then, we motivate our approach and describe its constituent algorithms in section 4.5. Finally we report experimental results on three different real-world data sets in section 5.8.

4.2 Related Work

One of the first works to stress the importance of temporal effects in Recommender Systems and to cope with it was the *timeSVD++* algorithm (Koren, 2010). The approach is to explicitly model the temporal patterns on historical rating data, in order to remove the "temporal drift" biases. It means that the time dependencies are modelled parametrically as time-series, typically under the form of linear trends, with a lot of parameters to be identified. Other approaches (Lu et al., 2009; Agarwal et al., 2010; Stern et al., 2009) rely on a Bayesian framework and on probabilistic matrix factorization, where a state-space model is introduced to model the temporal dynamics. One of their main advantages is that they could easily be extended to include additional user- or item-related features (addressing in this way the cold-start problem). But, in order to remain computationally tractable, they update only either the user factors, or the items factors, but never both factors simultaneously; otherwise, they should rely on rather complex non-linear Kalman filter methods. An earlier work (Rendle et Schmidt-thieme, 2008) also proposed to incrementally update the item- or user-related factor corresponding to a new observation by performing a (stochastic) gradient step of a quadratic loss function, but allowing only one factor to be updated; the updating decision is taken based on the current number of observations associated to a user or to an item (for instance, a user with a high number of ratings will no longer be updated).

Interestingly, tensor factorization approaches have also been adopted to model the temporal effects of the dynamic rating behavior (Xiong et al., 2010): user, item and time constitute the 3 dimensions of the tensors. Tensor factorization is useful for analyzing the temporal evolution of user and item-related factors, but it could hardly extrapolate rating behavior in the future.

4.3 Methods

4.3.1 Motivation

There many situations in various area in which it is very hard to consider adapting models. For instance in Natural Language Processing, the adaptation is almost always an important issue

because it would mean that users have to be involved in the process. Unfortunately, users are often times quite opposed to requests from systems trying to engage them into a collaborative procedure.

In the context of recommendation, and in particular with public datasets such as Netflix and MovieLens, we have the great opportunity to make use of users explicit feedback. The key idea of Dynamic Adaptation is to take advantage of this opportunity.

4.3.2 Principle

The adaptive framework we have presented in (Gaillard et al., 2013a) makes it possible for the system to have a continuous and Dynamic Adaptation along time. By doing so, we can overcome most of the drawbacks due to the cold-start. This section describes the process used to obtain such a framework. The key idea follows the simple principle that each update or new pair (u, i) needs to be taken into account instantaneously by the system.

On one hand, we can assume that delaying it for some days would not make sense since everything changes so fast nowadays. It could already be too late and thus lead to bad recommendations, especially for the ones based on a few number of ratings. One log of rating can make a big difference.

But on the other hand, one could think that it is wiser to wait for some time before adapting the models. This could also be a parameter for each user, adjusting the "stabilization time".

Updating continuously a recommender system is challenging and very demanding in terms of resources. Matrices of similarities have to be updated at each iteration. Such matrices are usually huge. Therefore, depending on the complexity level of the similarity update, continuous adaptation can be very difficult, not to say impossible.

For instance, the *Pearson* or *Cosine* similarity measure are not easily handled to pre-calculate similarities because of the structure of the formulae (square root, sum of squared differences) . It is therefore very hard to consider updating continuously.

Hence, a new function had to be designed allowing the system to update items-to-items and users-to-users similarities in a very efficient way.

4.4 Dynamic Adaptation in classical a CF approach

We present here the methods that have been developed in order to implement Dynamic Adaptation in a classical collaborative filtering system, without dimension reduction or pre-processing on data.

4.4.1 A new application of a classical similarity measure

We derived a similarity measure from the distance of Manhattan, also known as the taxicab distance (Krause, 1987). Since we want to have a similarity measure, we take its complement to one. Hence, the more ratings are close, the more the distance is short, the more the similarity tends to 1. And therefore the more users or items are considered to be alike.

In the remainder, this similarity function is used for both users and items. In the following k is either an item or user, x, y is a pair of items or users depending on the case. D is a contant standing for the maximum difference between two ratings.

$$Manhattan(x,y) = 1 - \frac{\sum_{k \in T_x \cap T_y} |r_{k,x} - r_{k,y}|}{|T_x \cap T_y| \times D} \tag{4.1}$$

As mentioned before, users can be on average more or less tolerant. To make allowance for these different behaviors, the difference $\overline{r_x} - \overline{r_y}$ is added to the formula, with a coefficient F.

For instance, let us look at the similarity between user a and user b. User a has a certain tendency to be very severe on items he rates. On the contrary, b is more indulgent. This difference of behaviors between a and b is somehow related to the difference of average ratings. In the end, this heterogeneity is taken into account in the similarity by a coefficient F.

Note that it is not the same as the gap function, that is used to adjust individually the rating prediction function. Here, we are dealing with similarities between two users or two items.

$$Manhattan(x,y) = 1 - \frac{\sum_{k \in T_x \cap T_y} |r_{k,x} - r_{k,y}| + F|\overline{r_x} - \overline{r_y}|}{(|T_x \cap T_y| + F)D} \tag{4.2}$$

We use a coefficient proportional to the cardinality of the intersection $T_x \cap T_y$ as a confidence measure. Therefore we are giving more weight to items sharing a greater number of users. We call this similarity measure the Manhattan Weighted Corrected similarity (*MWC*).

$$MWC(x,y) = Manhattan(x,y) \times \left(1 + \epsilon - \frac{1}{|T_x \cap T_y|^\alpha}\right) \tag{4.3}$$

The Manhattan Weighted Corrected function (eq. 4.3) is designed to address the need of reactivity. We thus reduced the complexity by one degree, keeping our system very well-fitted to Dynamic Adaptation.

For instance, we look at the similarity between item *a* and item *b*. User *robert* has previously rated item *a* and now rates item *b*, that he has never rated before. Why this new similarity measure is easy to update? Let us take a look at the numerator sum in the MWC function:

$$\sum_{u \in T_a \cap T_b} |r_{u,a} - r_{u,b}|$$

We need to add $|r_{robert,a} - r_{robert,b}|$ to the pre-calculated sum, as *robert* is now belonging to $T_a \cap T_b$. Then, the cardinality of the intersection $T_a \cap T_b$ has to be incremented by one. $\overline{r}_{robert}$ and $\overline{r_b}$ are easily updated. And we are done. We have updated the database by doing four simple additions. The same process is applicable for items.

Taking advantage of this property, we ran the updating algorithm on the training corpus. The complexity has been reduced from $o(n^2)$ to $o(n)$ (square to linear). If we consider the whole set of updates, we reduced from $o(n^3)$ to $o(n^2)$.

Additionally, a logical extension of the adaptation principle would be to give more weight to recent ratings (or less to old ones), wether it is in a similarity measure or a rating prediction. The algorithm has been designed to do so, by applying a time-based filter, described in the next paragraph.

4.4.2 Time-based weighting

Following the idea of adaption, we applied a time-based weighting function on ratings. Hence, more recent ratings will have more weight in the prediction of a rating, for a given user and a given item. By doing so, we aim to follow the evolution of tastes and stay close to them as time goes on.

Here is the formula we have used for recency. Our approach is more to penalize the past than enhance the present, but we could have combined both. Basically, we define a time penalty function P.

Let:

- r_x be a rating,

- t_x date it has been given,

- t the current time.

Then we have:

$$P(r_x) = \frac{1}{(t - t_x)^\alpha} \qquad (4.4)$$

The tricky part is the unit to use for time difference and α. In our case, we used months and $\alpha = 0.33$

We used months because it seemed an easily time scale to interpret to us. The parameter α has been determined so that a 6-months-old rating has twice less weight as a "present" rating.

Note that the time at which the rating has been given is not the only one that could be considered. Indeed, users can watch and rate a movie on two different days. That data could somehow be integrated in the model. Unfortunately, we do not have access to this data.

4.4.3 Error adaptation: learning from mistakes.

We are in an ideal case since once we have predicted a rating, we have access to the real one. It gives us a explicit feedback on how good our system is. Of course, we have access to these only once we have performed the rating prediction.
Therefore, for each user, we are able to compute an average error e_u. It is defined as follows:

$$e_u = \frac{1}{|S_u|} \sum_{i \in S_u} r_{u,i} - \hat{r}_{u,i} \qquad (4.5)$$

In this case, S_u includes the training set, so that we have more data to compute the estimation of e_u. By applying this correcting factor in equation 2.7 presented in Chapter 2, as follows:

$$\hat{r}w_{u,i} = \gamma \hat{r}_{u,i} + (1 - \gamma)(\alpha \overline{r_u} + (1 - \alpha)\overline{r_i}) + K \times e_u \qquad (4.6)$$

We performed several experiments with by varying the constant K on [0, 1]. Surprisingly, the effect of this correcting factor did not change significantly the performance of the system, according to the RMSE metric.
The work of sociologist Emmanuel Ethis (Ethis, 2004) on the interaction between the public and a movie supports this results. That is, no matter how hard we try to be accurate, there will always be an unpredictable part and something unique when a person and a movie come across.

4.5 Adaptive Matrix Completion

Starting from one of the standard static settings of matrix completion for Collaborative Filtering (CF), we will extend it to the time-varying case, by adopting an incremental, on-line approach based on temporal regularization.

Let **R** be a $n \times m$ rating matrix (n users, m items), with of course a lot a missing data. One of the standard state-of-the-art CF approaches amounts to approximate **R** by a low-rank matrix **X** that optimizes a criterion mixing:

- the approximation quality over observed ratings, typically the sum of squared errors;

- a complexity penalty, typically the nuclear norm (a.k.a. trace-norm) of **X**, as a way to recover a low-rank matrix.

Assuming the decomposition **X** = **L**. **R′** (with **L** and **R** having K columns if **X** is rank K at most), and introducing user- and item-specific biases (often called user subjective bias and item popularity) noted as **a** and **b**, the nuclear norm problem can be approximated by the following minimization problem (Recht et al., 2010):

$$\min \sum_{(i,j) \in \omega} \left(r_{i,j} - m - a_i - b_j - \sum_{k=1}^{K} L_{i,k} R_{j,k} \right)^2 + \mu_a \|\mathbf{a}\|^2 + \mu_b \|\mathbf{b}\|^2 + \mu_L \|\mathbf{L}\|_F^2 + \mu_R \|\mathbf{R}\|_F^2 \quad (4.7)$$

where ω designates the set of available rating tuples, m is the average rating over ω, a_i, b_j, $r_{i,j}$, $L_{i,k}$ and $R_{j,k}$ are respectively the elements of **a**, **b**, **R**, **L** and **R**, corresponding to user i, item j and latent factor k. $\|\mathbf{M}\|_F^2$ is the Frobenius norm of matrix **M**.

It should be noted that the regularization terms, including the ones related to **a** and **b**, are particularly critical in our case. Indeed, in real world cases, the test sets are chronologically posterior to the training and development sets so that, in practice, the standard iid (independent and identically distributed) assumption between the training and the test sets is far to be verified and a strong regularization is needed. One usual way of solving this optimisation problem is to use Alternating (Regularized) Least Squares or Stochastic Gradient Descent, see (Recht et Ré, 2013) for instance. Typically, the choice of the μ_a, μ_b, μ_L, μ_R, K parameters are done by grid search on a development set. In practice, we first look for the best values of the μ_a and μ_b parameters without any factor matrix in the model (i.e. a simple $r_{i,j} \approx m + a_i + b_j$ model); then we fix them and optimize for the remaining parameters.

4.5.1 Adaptation of a_i and bj

Let us first consider the simple model including only the item and user biases, before describing the extension to the complete model based on matrix factorization (MF). When observing a new tuple $< i, j, r_{i,j} >$, we update a_i and b_j by minimizing the following criterion

$$\min(r_{i,j} - m - a_i - b_j)^2 + \alpha_1(a_i - \tilde{a}_i)^2 + \beta_1(b_j - \tilde{b}_j)^2 \tag{4.8}$$

where $\tilde{a}_i$ and $\tilde{b}_j$ are the values before the adaptation. This criterion is a trade-off between approximation quality with respect to the new observation and smoothness in the evolution of the biases. For new users and items, $\tilde{a}_i$ and $\tilde{b}_j$ are set to 0. The values of α_1 and β_1 are obtained by a grid search on a development set, which is chronologically posterior to the training set.

Solving this optimization problem leads to the following simple update equations:

$$a_i = \frac{(\alpha_1 + \alpha_1/\beta_1)\tilde{a}_i + r_{i,j} - m - \tilde{b}_j}{1 + \alpha_1 + \alpha_1/\beta_1} \tag{4.9}$$

$$b_j = \frac{(\beta_1 + \beta_1/\alpha_1)\tilde{b}_j + r_{i,j} - m - \tilde{a}_i}{1 + \beta_1 + \beta_1/\alpha_1} \tag{4.10}$$

4.5.2 Adaptation of L_i and R_j

Latent factor terms $\mathbf{L}$ and $\mathbf{R}$ are adapted too, according to the same idea: observe $< i, j, r_{i,j} >$ then update L_i and R_j (respectively the i-th row of $\mathbf{L}$ and the j-th row of $\mathbf{R}$), such that:

$$\min(\widehat{r_{i,j}} - \sum_k L_{i,k} R_{j,k})^2 + \alpha_2 \|L_i - \widetilde{L}_i\|_F^2 + \beta_2 \|R_j - \widetilde{R}_j\|_F \tag{4.11}$$

where $\widehat{r_{i,j}}$ is equal to $r_{i,j} - m - a_i - b_j$ (i.e. the residual rating), while $\widetilde{L}_i$ and $\widetilde{R}_j$ are the values of the corresponding rows before adaptation. For new users and items, the entries of $\tilde{L}_i$ and $\tilde{R}_j$ are set to 0. The values of α_2 and β_2 are obtained by a grid search on the development set.

Unfortunately, there is no closed-form solution to this problem, due to the coupling between L_i and R_j. However, this could be solved iteratively by applying recursively the following equations:

$$L_i^{(t)} = (\alpha_2 I + R_j'^{(t-1)}.R_j^{(t-1)})^{-1}.(\alpha_2 L_i^{(t-1)} + \widehat{r_{i,j}}.R_j^{(t-1)}) \tag{4.12}$$

$$R_j^{(t)} = (\beta_2 I + L_i'^{(t-1)}.L_i^{(t-1)})^{-1}.(\beta_2 R_j^{(t-1)} + \widehat{r_{i,j}}.L_i^{(t)}) \tag{4.13}$$

with $L_i^{(0)} = \widetilde{L}_i$ and $R_j^{(0)} = \widetilde{R}_j$. Experimentally, for all datasets we used and the corresponding values of α_2 and β_2, two or three iterations were sufficient to converge.

We observe important differences across datasets. Vodkaster and Netflix are totally opposed to each other. The weight given to recent ratings for Vodkaster users is 10 times higher (10%) than the one used on Netflix users (1%). This is coherent with what we observed in 4.5.1. Indeed, we have seen that Vodkaster users are dynamic and prone to changes in time. Therefore, it makes sense to put the emphasis on recent ratings. The contrary is true for items, that are more static on Vodkaster than on Netflix or MovieLens. We have studied and investigated where this difference of nature between datasets might come from in Chapter 2 and in Chapter 3.

4.6 Results

In this section, we present the results obtained with and without adaptation, on Vodkaster, Netflix and MovieLens datasets. We also provide baseline results, that is without adaptation, for both collaborative filtering and matrix factorization techniques.

Experiments have been performed on 3 datasets: MovieLens (1M ratings), Vodkaster (2M), Netflix (2M), divided into 3 temporal (chronologically ordered) splits: Train , Dev (20k), Test (20k). Recall that Vodkaster is a rather new Movie Recommendation website, dedicated to rather movie-educated people. These datasets show very different characteristics: Netflix has a high number of users and is spread over a short time period (less than 10 months, Dev and Test sets represent each 1 week). MovieLens has a high number of users and is spread over a long time period. Vodkaster has a low number of users and is spread over a short time period (one year), but users are very "loyal" and active.

4.6.1 Baseline results

We present here the results obtained with our baseline.

Dataset	RMSE	MAE
Netflix	1.0484	0.8669
MovieLens	1.0976	0.9097
Vodkaster	1.1158	0.8924

Table 4.1: *Results with Baseline 1, $x_{i,j} = \mu$*

4.6.2 Results with Classical Collaborative Filtering

In this section, we want to explicit the effect of adaptation on different metrics and on different datasets at full coverage, with a classical collaborative filtering *k*NN method.

Tables

We can see in Table 4.4, that every metrics has been improved. The gain is approximately 10% on RMSE and MAE. *Remark*: optimization and tuning of the model has been performed according to RMSE on the development set. However, the results with Dynamic Adaption on the Development and Test sets are close, and this is true across datasets (Netflix, MovieLens, Vodkaster). Therefore, Dynamic Adaptation is a robust approach.

	Netflix Dev set		Netflix Test set	
	No Adaptation	**Adaptation**	No Adaptation	**Adaptation**
MAE	0.753	0.713	0.772	0.728
RMSE	0.986	**0.920**	0.999	**0.927**
Precision users	0.807	0.833	0.795	0.822
Recall users	0.904	0.931	0.896	0.925
F score users	0.841	0.868	0.831	0.858
Precision items	0.832	0.849	0.824	0.843
Recall items	0.879	0.898	0.875	0.894
F score items	0.850	0.868	0.843	0.862
MAP	0.629	0.648	0.621	0.633

Table 4.2: Results on Netflix

	MovieLens Dev set		MovieLens Test set	
	No Adaptation	**Adaptation**	No Adaptation	**Adaptation**
MAE	0.759	0.708	0.793	0.692
RMSE	0.984	**0.908**	1.035	**0.885**
Precision users	0.754	0.756	0.778	0.784
Recall users	0.876	0.868	0.882	0.882
F score users	0.801	0.800	0.820	0.824
Precision items	0.695	0.673	0.712	0.701
Recall items	0.798	0.770	0.801	0.779
F score items	0.733	0.708	0.744	0.730
MAP	0.549	0.527	0.512	0.483

Table 4.3: Results on Movielens

	Vodkaster Dev set		Vodkaster Test set	
	No Adaptation	**Adaptation**	No Adaptation	**Adaptation**
MAE	0.992	0.634	0.771	0.652
RMSE	1.293	**0.830**	1.007	**0.834**
MAP Precision users	0.726	0.784	0.769	0.794
Recall users	0.846	0.910	0.891	0.916
F score users	0.770	0.831	0.815	0.841
Precision items	0.653	0.788	0.709	0.757
Recall items	0.713	0.857	0.787	0.833
F score items	0.674	0.813	0.737	0.784
MAP	0.743	0.719	0.748	0.742

Table 4.4: Results on Vodkaster

We can observe that the effect of adaptation is much stronger on the Vodkaster dataset. Indeed, the RMSE without adaptation is 1.293 and with adaptation 0.830, which represents a gain in accuracy of 35%. This could be explained by the fact that the data extracted from Vodkaster are

real-world unmodified data and continuous from a chronological point of view. Moreover, the dataset is "full" and not a subset of a larger dataset.

Additionally, users behavior could be more heterogenous since there is no recommendation engine on Vodkaster. On the contrary, MovieLens and Netflix have always had recommendations, leading to a more stable, controlled and homogenous behavior of users. Consequently, on these datasets, the effect of Dynamic Adaptation is less important as users tastes are more static and thus predictable.

Another difference might come from the fact that Vodkaster is a movie lovers community; users may be a little sharper when they come to evaluate a movie and they may also take risks and explore new genre of movies. Hence their tastes might evolve more than on other datasets like Netflix and MovieLens. This is also coherent with our previous observations in 4.5.1.

On Figure 4.2, it is very interesting to note that on the Netflix Test set (red plain line), as we move temporally away from the Training set, the RMSE increases significantly (from 1.0 up to 1.25). This shows clearly the fundamental importance of Dynamic Adaptation.

Curves

We present here a specific type of graph that allows to explicit the effect of Dynamic Adaptation. Figure 4.5 displays explicitly the effect of dynamic adaptation over time. Each point of coordinates $x = n$ corresponds to the average RMSE after observing n ratings from the user (starting from the beginning of the test set), the average being computed over the users who have rated at least n items in the test set. This corresponds to a relative user-centric timescale and shows that, without adaptation, prediction errors increase, while it is stabilizing to a much lower value with adaptation.

Note that the more x increases, the less users will be remaining. Therefore, we had to put a lower threshold on the number of users below which it does not make sense to compute an average. We chose to stop when only 25 users are remaining. This is why, depending on datasets, x_{max} is different.
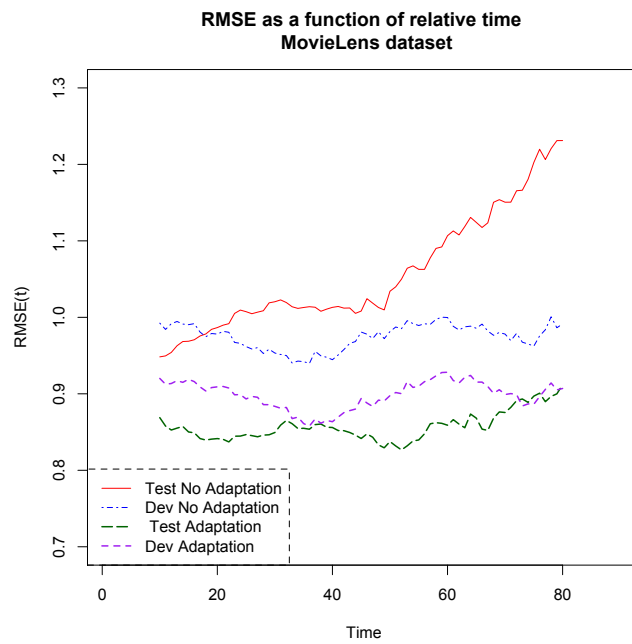
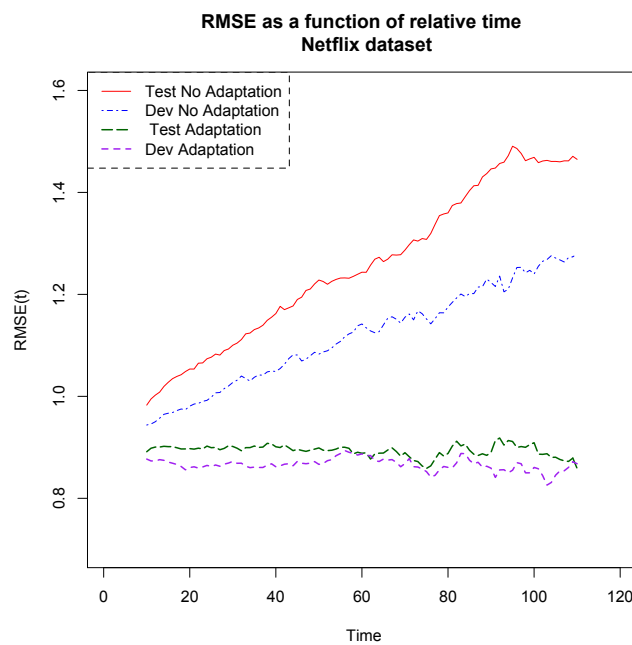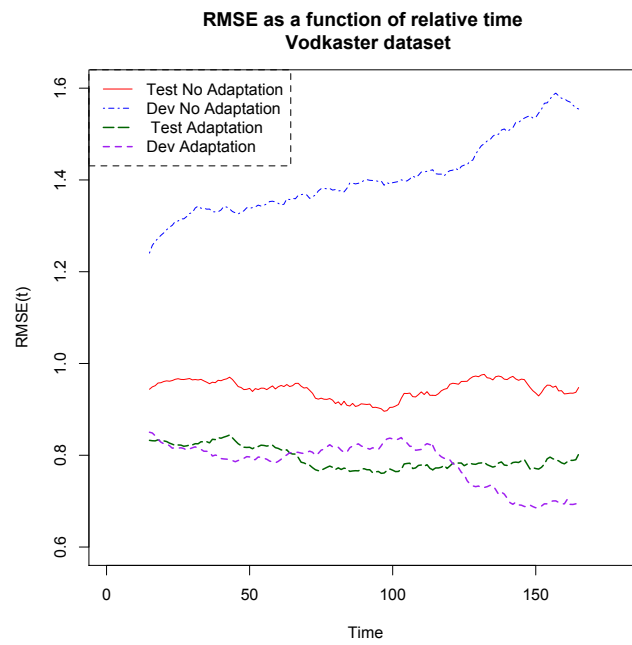**Figure 4.1:** *RMSE as a function of relative time MovieLens Dataset*

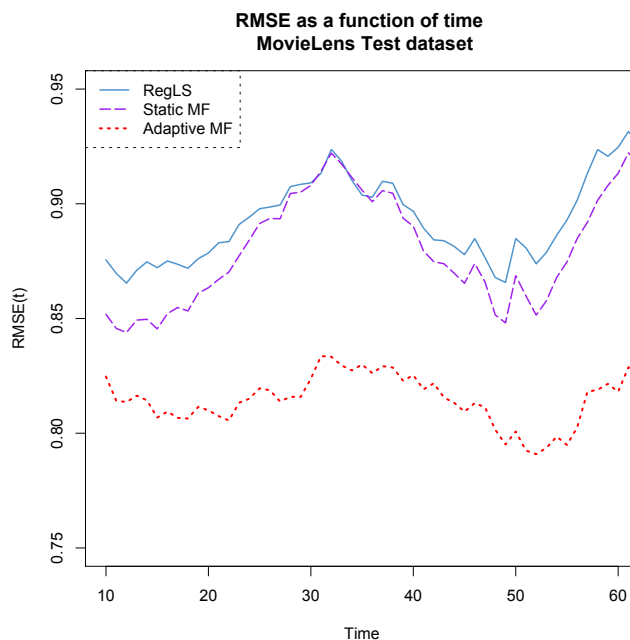**Figure 4.2:** *RMSE as a function of relative time Netflix Dataset*

Figure 4.3: *RMSE as a function of relative time Vodkaster Dataset*

4.6.3 Results with Adaptive Matrix Completion

		RMSE	MAE	MAPE
Vodkaster	Baseline 1 $x_{i,j} = m$	1.1158	0.89239	0.5296
	RegLS	0.8465	0.6603	0.3477
	MF(on residuals)	0.8177	0.631	0.3294
	Adapting a_i and b_j - L_i and R_j	**0.7805**	**0.5993**	**0.3031**
Netflix	Baseline 1 $x_{i,j} = \mu$	1.0484	0.8669	0.3209
	RegLS	0.9344	0.7322	0.2755
	MF(on residuals)	0.9161	0.7118	0.27
	Adapting a_i and b_j - L_i and R_j	**0.8685**	**0.6678**	**0.2506**
MovieLens	Baseline 1 $x_{i,j} = \mu$	1.0976	0.90965	0.385
	RegLS	0.9194	0.713	0.3011
	MF(on residuals)	0.9047	0.7012	0.2943
	Adapting a_i and b_j - L_i and R_j	**0.8435**	**0.6528**	**0.2576**

Table 4.5: Results with Matrix Factorization on Vodkaster, Netflix and MovieLens Test sets. RegLS corresponds to the simple model with biases identified by regularized least squares, while MF designates the prediction model based on Matrix Factorization

The results show that Adaptive methods improve the performances according to RMSE, MAE and MAPE metrics (Table 4.5). The gain in RMSE is much stronger on Netflix (0.916 to 0.868) and MovieLens (0.904 to 0.843) than on Vodkaster (0.817 to 0.78).

Figure 4.4: RMSE as a function of relative time MovieLens Test set - MF

It is very interesting to see on Figure 4.7, how users change their bias and how items change
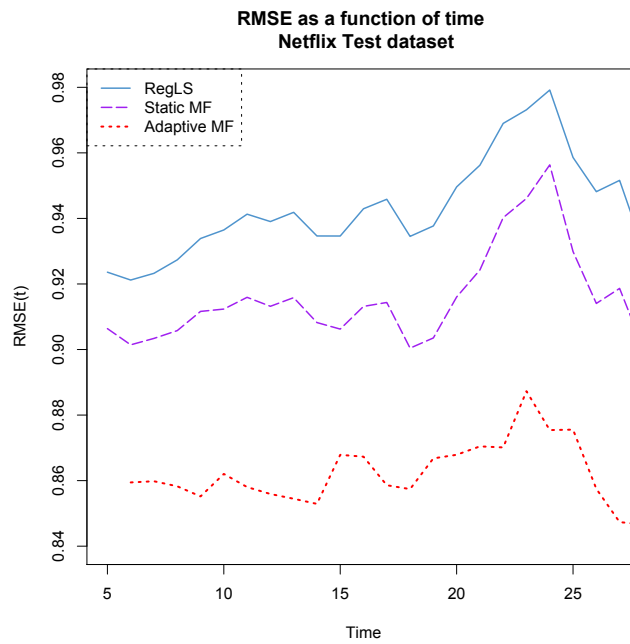
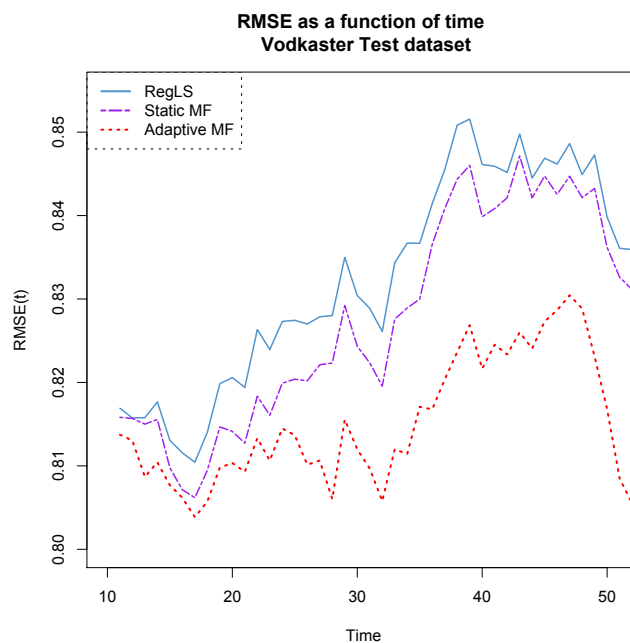**Figure 4.5:** *RMSE as a function of relative time Netflix Test set - MF*

Figure 4.6: *RMSE as a function of relative time Vodkaster Test set - MF*

their popularity along time. We observe that Netflix and Vodkaster datasets are totally different for a dynamical point of view. On Vodkaster, users are very dynamic while items are rather static in term of popularity, or at least it varies much more slowly. On Netflix, we observe the

exact opposite tendency (static users and dynamic items).
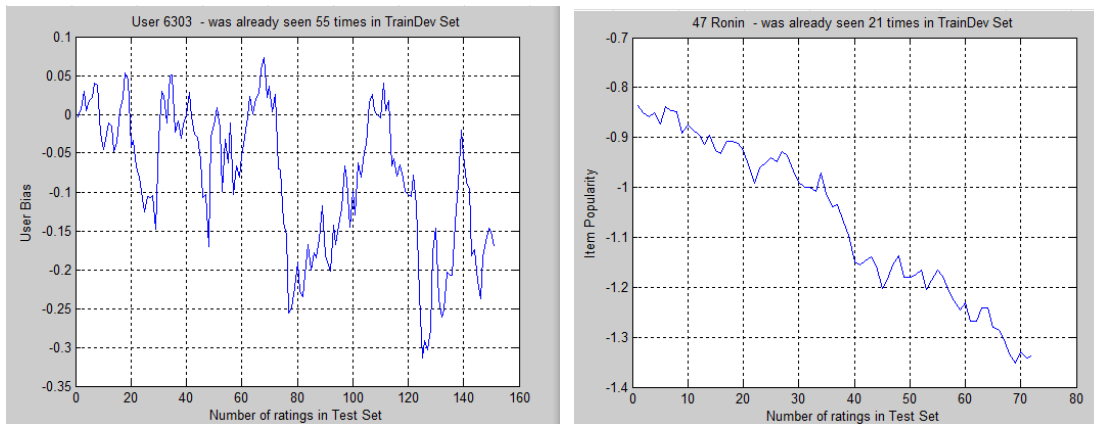
Figure 4.7: *Examples of Adaptation on individual cases, Vodkaster*

We observe important differences across datasets. Vodkaster and Netflix are totally opposed to each others. The weights (α_1, α_2) given to recent ratings for Vodkaster users is 10 times higher (10%) than the one used on Netflix users (1%). This is coherent with what we observed in 4.5.1. Indeed, we have seen that Vodkaster users are dynamic and prone to changes in time. Therefore, it makes sense to put the emphasis on recent ratings. The contrary is true for items, that are more static on Vodkaster than on Netflix or MovieLens.

4.7 Analysis

In this section, we analyze the results and give some examples of good recommendations and some failures.

4.7.1 Dynamic Adaptation impact on performances

The effect of Dynamic Adaptation is observable on the three datasets on which we have tested our system and for both techniques we have used, namely classical Collaborative Filtering and Matrix Factorization.

We observe that Dynamic Adaptation improves significantly the performances on all datasets and that Adapted Matrix Factorization techniques beat the classical Collaborative Filtering approach.

4.7.2 Adapting with predictions

We have performed experiments where the system was adapted according to predictions instead of real values observed. We used the classical CF approach in these experiments. The results on the Test set of Netflix are the following MAE = 0.739, RMSE = 0.946 on the Development set and MAE = 0.750 RMSE = 0.964 on the Test set.

Recall that, with same method on the same dataset, without adaptation RMSE = 0.999 and with adaptation RMSE = 0.927. Hence, even if we did not have access to the real feedback, the Dynamic Adaptation would still be efficient and an improvement.

4.7.3 Reflexion for sociological analysis

We think that the focus should no longer be on proper ratings themselves, but only on the a_i and b_j. We have to uncorrelate users and items and thus use the user bias and the item popularity. Finally it is not the rating in itself that matters, because it combines two things: the item (movie) and the user's sensitivity. Whereas, when we uncouple these two components into two separated variables a_i and b_j, it allows to see a certain regularity. It is like if users would give ratings that are independent from the movie rated. This is why it is more interesting to observe the evolution (if any) of the item popularity along time (b_j), because it does not depend on a particular user. The same thing is true for users: if someone always watches great movies that everybody likes, we will assume this person always gives high ratings. But it is wrong, this user gave high ratings because the movies were very good and the global opinion about them was also very good. So once we have uncoupled the item popularity and the rating, what remains is much more representative of the user.

4.7.4 Examples

We present some examples of good recommendations and mistakes too.

- *The Hobbit : An Unexpected Journey* (2011, USA) recommended to user *Zarai*.

	#ratings in training	#ratings in test	Average
The Hobbit	0	7	3.76
Zarai	0	87	4.4

Predicted rating	5	Real rating	5

We are able to recommend a film that has not been rated yet to a user unseen in the training. Thanks to adaptation, this becomes possible and the prediction is a very good one.

- *Le Père Noël est une ordure* (1982, France) recommended to user *Fernand*.

65

	#ratings in training	#ratings in test	Average
Le Père Noël...	14	2	4.1
Fernand	0	45	4.2

Predicted rating	5	Real rating	5

We recommend this film seen 14 times in the training corpus to a user named Fernand unseen in the training corpus (does not include any movie rated by him). However, at the moment we recommend this movie, we can take into account all of his ratings found in the test so far. This example is a good proof of the interest of a short-term adaptation.

- *The Nightmare Before Christmas 3D* (2006, USA) recommended to user *Bart*.

	#ratings in training	#ratings in test	Average
The Nightmare...	2 ($r = 1, 3$)	2 ($r = 4.5, 5$)	3.4
Bart	0	31	4.68

Predicted rating	4.7	Real rating	2.5

This is the first error observed when the predictions values are sorted in decreasing order. The user has rated this movie 2.5. However, it has been well rated in the test and it's probably the main reason we have recommended it. Before the recommendation has been done, Bart's average rating was 4.68, which is very high. In this case, adaptation misleads the system.

4.8 Conclusions

In order to obtain a flash reactivity, we have proposed a new application of the similarity measure based on the distance of Manhattan. This new measure named *Manhattan Weighted Corrected* similarity leads to a significant decrease in complexity and allows an instantaneous adaptation. Hence we are able to update the parameters of the recommender system step by step, whenever a new rating occurs.

We have also proposed an Adaptive Matrix Completion method that allows Recommender Systems to be highly dynamic. Experimental results showed that this method improves significantly the accuracy of predicted ratings, even if there is still a residual noise which seems unavoidable when using only rating data and no other features. Future works should now focus on extending this scheme to time-varying user- and item- features, but also investigate other matrix regularizers to automatically determine the optimal reduced rank K. Ideally, we should also introduce some meta-adaptation that allows the adaptation rates ($\alpha_1, \beta_1, \alpha_2, \beta_2$) to vary over time.

Thanks to this new approach, we obtained results outperforming the one's obtained with a classical static system. Moreover, by applying the same algorithm during the training phase, we have dramatically reduced its complexity. We have also shown that this method allows us to perform a detailed analysis of the prediction errors (bad recommendations).

This is coherent with our conception of recommendation. We believe that nowadays recommender systems have to be instantaneous, giving the right recommendation at the right time, learning from their mistakes, and adapting the model not to repeat again and again the same errors.

However, we have seen that perfection is obviously unreachable. This why efforts should be put on finding new approaches to the recommendation problem, that are not only based on a prediction problem. In fact, it might even be necessary to reformulate the problem and model the task differently.

CHAPTER

5

ARGUMENTATION

Contents

Abstract

In this chapter, we present an innovative proposal for designing a domain-independent Semantic Recommender System relying on a word based similarity function (WBS), providing textually well-argued recommendations to users. Moreover, this system has been developed in a dynamic and adaptive framework presented in chapter 4. This might be the first step really made towards an anthromorphic and evolutive recommender.

69

5.1   Introduction

Some systems incorporate semantic knowledge to improve quality. Generally, they apply a concept-based approach to enhance the user modeling stage and employ standard vocabularies and ontology resources. For instance, ePaper (scientific-paper recommender), computes the matching between the concepts constituting user interests and the concepts describing an item by using hierarchical relationships of domain concepts (Maidel et al., 2008). Codina and Ceccaroni (Codina et Ceccaroni, 2010) propose to take advantage of semantics by using an interest-prediction method based on user ratings and browsing events. However, none of them are actually based on the user opinion as it is expressed in natural language.

We have designed a highly dynamic system relying on textual content, extracted from movie reviews.

5.2   Argumentation

In this section, we describe in details the methods already introduced in (Gaillard et al., 2013b) to provide textual argumentation. Also, we propose new enhancements we have implemented since then.

Sim can be replaced by several similarity such as Pearson, Cosine or MWC similarity, in formula (6.1) (Tan et al., 2005).
These measures are used to estimate the likeliness between two users, or items. Based on that estimation, we consider that two users or items are alike, or not. Before going further, it would be a good idea to define what "alike" should mean in the particular case of recommendation. Assume two users rate the same movies with equals ratings. Then, according to all of these similarities, which will be maximal, these two users will be identical.

Still, they might have totally different reasons to rate the same movies identically. Therefore, is not obvious that they are "alike".
By the same token, none of these similarity functions is able to provide the reasons why two users or items are similar. The fact that they rely on ratings only make them very hard to interpret.

5.3 Review segmentation: Hidden Markov Model and Viterbi algorithm

Reviews might have good parts and bad parts. To be able to extract which features users like and dislikes about a given item, we need to identify positive and negative segment within reviews.

To do so, we use the well-known Viterbi algorithm combined with Hidden Markov Models (HMM). Let's consider the following simple HMM. This model is composed of two states, Pos (positve) and Neg (Negative).

There are several paths through the hidden states (Pos and Neg) that lead to the given sequence, but they do not have the same probability. The Viterbi algorithm is a dynamical programming algorithm that allows us to compute the most probable path (as well as its probability).

It requires knowledge of the parameters of the HMM model and a particular output sequence and it finds the state sequence that is most likely to have generated that output sequence. It works by finding a maximum over all possible state sequences.

In fact there are often many state sequences that can produce the same particular output sequence, but with different probabilities. It is possible to calculate the probability for the HMM model to generate that output sequence by doing the summation over all possible state sequences. This also can be done efficiently using the Forward/Backward algorithm, which is also a dynamical programming algorithm.

Finally, we use this approach to segment micro-reviews into positive and negative parts. This will be useful when we display arguments for recommendation (see section 5.7)

5.4 New similarity based on words

We have proposed in (Gaillard et al., 2013b) a new similarity method, taking into account words used by users in their past reviews about items. In the remainder, we call it the *Word Based Similarity* (WBS).

For each user x (or item), we define a vocabulary set V_x. Each word $w \in V_x$ is associated with a set of ratings $\mathbf{R}_{w,x}$. Since in most cases, words are used several times, we compute for each w its average $\overline{r_w}$.

Some words are more or less important and thus their weight in the similarity should be balanced. To do so, we define a weight function F_w, mixing the well-known Inverse Document Frequency $IDF(w)$ with the variance σ_w^2.

Consequently, words commonly employed and words w with very heterogenous ratings $\mathbf{R}_{w,x}$ (i.e a high variance) will have a smaller contribution in the similarity.

N_w is the number of items in which the word w appears. N_{tot} is the total number of items. D is the maximum difference between two ratings.
Note that F_w has to be updated at each iteration. Consequently, we have to update the σ_w^2 and IDF(w) at each iteration, for every word. Paying attention to avoid a whole re-estimation of these two variables, we derived an iterative relation for the two of them.

$$F_w = -log\left(\frac{N_w}{N_{tot}}\right) \times \frac{1}{\sigma_w^2} \tag{5.1}$$

Finally, the Word Based Similarity is defined as follows:

$$WBS(x,y) = \frac{\sum_{w \in V_x \cap V_y}(D - |\overline{r_{w,x}} - \overline{r_{w,y}}|)F_w}{D \times |V_x \cap V_y|\sum_{w \in V_x \cap V_y} F_w} \tag{5.2}$$

The Manhattan Weighted and Corrected similarity (MWC), that we introduced in Chapter 2 and (Gaillard et al., 2013a), will be used as a point of comparison. Again, note that the textual content is not taken into account.

5.5 Basic textual recommendation

Using this method, the similarity measure employed does not need to be based on words. The system actually provides selected reviews from user u and item i (written by other people).

To incite u to choose i, the system shows the following reviews:

- Among reviews written about i and associated with the highest ratings, select the one posted by the most alike user v, with respect to u.

- Among reviews written by u and associated with the highest ratings, select the one posted about the most alike item j, with respect to i.

On the contrary, to recommend *against* choosing i, the system displays the following reviews:

- Among reviews written about i and associated with the lowest ratings, select the one posted by the most alike user v, with respect to u.

- Among reviews written by u and associated with the lowest ratings, select the one posted

about the most alike item j, with respect to i.

This method is quite simple but turns out to be reasonably efficient. Examples of outputs are given in section 5.7.1, page 74.

5.6 Extraction of specific arguments

This second innovative feature somehow amounts to predict what particular characteristics a user is going to like or dislike about an item. This has been made possible thanks to the new similarity measure (WBS), introduced above.

5.6.1 Principle

Let us consider a user u and an item i.

The system has access to what u has written on other items in the past and what other users have written on item i. When doing rating prediction, it computes the similarities between u and $v \in T_i$ and between i and $j \in S_u$.
For each of these similarities, we select the term that have contributed the most in the calculation of $WBS(i, j)$ and $WBS(u, v)$, that is word associated to the greatest element in the sum (see equation 5.2). We define a set B containing of tuple (words, rating) that we sort by relevance using F_w. Then, we define two subsets: P_w contains previously selected words associated with a high rating in i and N_w for low ratings.

Finally, we display the top 5 most relevant arguments contained in both P_w and N_w, and each of them is given in the context they have been used for item i. As an example, some outputs are shown in section 5.7.2.

5.6.2 Display

In order to be more readable, we put the top 5 list of selected arguments back into their original context. To do so, we have to keep track of where these words come from with an inverse index. If a given word has been used by many neighbors, we choose the context of the nearest one, i.e the most similar to the current user.
The context is delimited by positive and negative segments, i.e if a selected word, once put back into its original context, lies inside a positive part, we will show the positive context only.

5.7 Argumentation outputs

In this paragraph, we show some outputs of our system. We first show two examples of outputs given by the basic method.

5.7.1 Basic method outputs

Here are some outputs:

Arguments

For the recommendation of "New York 1997" to Olga85.

👍 What Hal5000 wrote about "New York 1997" with a rating = ★★★★★:

"Genre film, but committed, cult and transgressive. The city as a character. A hero-synthesis. A darkness assumed, delicious. Memorable."

You and Hal5000 are 82 % alike.

👎 What Cityhunter wrote about "New York 1997" with a rating = ★:

"Saw the french version. It is a quiet film, losing the pace in some parts and boring characters."

You and Cityhunter are 78 % alike.

👍 What you wrote about "Gataca" with a rating = ★★★★:

"For once, the trailer is a true reflection of the movie. Not unpleasant at all."

"New York 1997" and "Gataca" are 83 % alike.

👎 What you wrote about "The Dandelions" with a rating = ★★:

"Pretentious film with a spectacular start but does not keep its promises."

"New York 1997" and "The Dandelions" are 92 % alike.

Arguments

For the recommendation of "`The Wicker Man`" to `Truman`.

👍 What `Nicolas23` wrote about "`The Wicker Man`" with a rating = ★★★★ :
"Shamefully too little known. We are taken in by this atmosphere that gets our goat right from the start and it is confirmed by the ending."
You and `Nicolas23` are 88 % alike.

👎 What `Tomy1` wrote about "`The Wicker Man`" with a rating = ★ :
"Yet another rubbish film for Nicolas Cage."
You and `Tomy1` are 71 % alike.

👍 What you wrote about "`Mr. Nobody`" with a rating = ★★★:
"A good futuristic movie but it has died out. Some lengthy parts."
"`The Wicker Man`" and "`Mr. Nobody`" are 79 % alike.

👎 What you wrote about "`Seed`" with a rating = ★★:
"Bad, really bad. Full of incoherences and a pathetic ending. Total failure."
"`The Wicker Man`" and "`Seed`" are 81 % alike.

5.7.2 Specific arguments outputs (WBS)

We took the same examples so that comparison is easy. The most relevant elements extracted (in bold) are put back into their context. The number next to each argument is its weight, on a scale from 0 to 100.

Some words are present in both methods. This shows that in both case, the arguments are relevant or at least they are coherent.

Arguments

For the recommendation of "New York 1997" to Olga85.

👍 **What you might like**

71 *One of the best **sci-fi** movie, real thing.*

69 *Fifteen **amazing** minutes*

45 ***Poetical** and delicate*

27 *Gets our **goat** right from the start*

👎 **What you might dislike**

63 ***Boring** characters*

49 *Losing the **pace** in some parts.*

Arguments

For the recommendation of "The Wicker Man" to Truman.

👍 **What you might like**

85 *Every now and then, a real **comedy***

74 *Some really **touching** scenes*

37 *Gore, funny and **cult.***

32 *Gets our **goat** right from the start*

👎 **What you might dislike**

72 *Purely and simply **rubbish***

64 *Cage-ish **crap**, good start though.*

29 ***pathetic** ending*

5.8   Results

Table 5.1 shows the results for 4 different methods: the classical Pearson (PEA) method without adaptation, the MWC method with and without adaptation (MNA) and WBS. Within the confidence interval, in terms of accuracy, the same performances are obtained by MWC and WBS. Both outperform PEA and MNA. Note that the key point here is the comparison between results obtained the baseline, i.e Pearson without adaptation and the method we propose, WBS. Our word based approach is thus able to offer the arguments feature without any loss of performances with respect to any others RS methods that we know of.

Set	Method	RMSE	MAE	%Precision	CI
D	PEA	1.302	0.997	71.2	1.49
E	MNA	1.293	0.992	72.6	1.26
V	MWC	0.83	0.634	78.4	1.12
	WBS	0.85	0.651	77.2	1.16
T	PEA	1.01	0.78	75.7	1.51
E	MNA	1.007	0.771	76.9	1.30
S	MWC	0.834	0.652	79.4	1.22
T	WBS	0.857	0.672	79.1	1.24

Table 5.1: *Results with Pearson (PEA), MWC, MWC without Adaptation (MNA), WBS. CI is the radius confidence interval estimated in % on accuracy (Acc.).*

MNA (MWC without adaptation) being better and more easily updated than Pearson (PEA), we have decided to use the adaptive framework only for MWC. We want to point out that the results are the same for both MWC and WBS methods, within a confidence interval (CI) radius of 1.16%.

Unfortunately, it is impossible to evaluate directly the quality of the arguments proposed. However, considering the fact that they have been selected according to their weights in the rating prediction function and that results in terms of accuracy are not deteriorated; we can assume and estimate that the selection of these particular arguments is relevant. In short, from a qualitative point of view, these results can be seen as an assessment of our approach based on words.

5.9   Conclusion

Our conception of relevance is not binary. Items should not be either recommendable or not. We want to capture the whole spectrum of relevancy by letting users make their own choice, by providing them interpretable guidelines, i.e arguments. Eventually, improving the RMSE by 5% does not look so important. What really matters is finding a way to let relevance speak from itself, through the critical judgement of people using our recommender system.

The goal of this chapter was to make an innovative proposal for designing a domain-independent semantic recommender system relying on a word based similarity function (WBS), providing textually well-argued recommendations to users. Moreover, this system has been developed in a dynamic and adaptive framework. This might be the first step really made towards an anthropomorphic and evolutive recommender.

As a perspective of improvement, we ask the following open question: is generating a real argumentation fully articulated, sentenced and personalized within or out of our reach?

CHAPTER

6

MATCHING GAMES IN RECOMMENDER SYSTEMS

Contents

Abstract

In this chapter, we transpose a famous problem in game theory known as matching games to the recommendation problem. The idea is to satisfy both side of the recommendation: users and items. Finally, the system will find a stable equilibrium that maximize both utilities on the users and items side.

6.1 Introduction

As we have seen in Chapter 2, when the system performs a rating prediction for a pair (u, i), it actually does a linear combination of two rating predictions. Here is a quick reminder:

$$rating(u, i) = \frac{\sum_{v \in T_i} Sim(u, v) \times r_{v,i}}{\sum_{v \in T_i} |Sim(u, v)|} \quad (6.1)$$

This approach is clearly user oriented in the sense that we take into account the similarities between users.

A symmetrical formula $rating(i, u)$ item-oriented can be derived from (6.1):

$$rating(i, u) = \frac{\sum_{j \in S_u} Sim(i, j) \times r_{u,j}}{\sum_{j \in S_u} |Sim(i, j)|} \quad (6.2)$$

Thanks to these two rating functions, we can derive the predicted preferences for users and items, on a given test dataset.

One of the main goals of a recommendation engine is to provide a personalized ranked list of items to users. The main difference here is that we no longer perform some linear combination of these two rating predictions and then sort items according to this combination. Instead, we approach the a recommendation list with a general assignment problem, using a matching game algorithm.
The idea behind using this approach is to make the algorithm less systematic and less oriented towards the rating prediction alone. We aim to give more importance to items not popular but that might be potentially interesting for a given user. Users satisfaction is not only measured by how precise the system is.

6.2 The college admission problem

This problem is in fact very common. Suppose we have the following typical situation: considering a set of n applicants to a college able to admit a quota of only q. The admissions office has to decide which students to admit, with respect to their qualifications. The method consisting in admitting only the q best applicants is not satisfactory in most cases, since we can not be sure that all who are offered admission will actually accept. For that reason, if a college wants to receive q acceptances, it will often have to accept more than q candidates. The choice of which ones and how many to admit is quite tricky and is often done by experimented recruiters.

In this type of situation, for a given applicant, we might not know:

- If he has applied in other colleges

- How he has ranked his preferences

- If other colleges will accept him

Hence, there is a important part of unknown, leading to problems for both students and colleges. In (Gale et Shapley, 1962), the authors propose a procedure for assigning applicants to colleges which should be satisfactory to both groups. Assuming there are enough applicants, this method allows each college to attain precisely its quota.

6.3 Matching game transposed to recommendation

The idea is quite simple, we replace applicants by users and colleges by items. In the remainder, for easy understanding purpose, we personify users and items, as if they were active in the process. They are actually passive, and every choice is made automatically by the system.

6.3.1 Assignment criteria

Assume we have a set of n users. Each one of them has to be assigned among m items. Each item i has a quota q_i. Users and items order of preferences have been predicted.

If a user has no preference between two (or more) items, he still needs to list them in order, so that ties are avoided. In a similar way, each item ranks the users who have been to it in order of preference. Now that we have the quotas of the items and the two sets of orderings, we need to find an assignment of users to items.
To make things clear, let us take an example. Assume we have two items i_1, i_2 and two users u_1, u_2. Suppose u_1 prefers i_1 and u_2 prefers i_2. But i_1 prefers u_2 and i_2 prefers u_1. In that particular case, no assignment can be satisfactory to all preferences.
However, it obvious that items (products) exist for users (consumers) rather than the other way around. Therefore, it would be right to assign u_1 to i_1 and u_2 to i_2. We can thus consider that users should receive more consideration than items.

We define an unstable assignment as follows:
if there are two users u_1 and u_2 who are assigned to items i_1 and i_2, respectively, although u_2 prefers i_1 to i_2 and i_1 prefers u_2 to u_1.
This assignment would be "unstable" in the way that it could be upset by a item and user acting together in a sense which benefits both.

It has been proved that a stable assignment always exist in (Gale et Shapley, 1962). We define an optimal stable assignment:

A stable assignment is called optimal if every users is at least as well off under it as under any other stable assignment.

6.3.2 Example

We are considering here the particular case where there are the same number of users (n) and items (m), and the quota for each item is equal to one. We assume that each user and each item has its ordered list of preferences. Suppose we have a ranking matrix of three users u, v and w, and three items, i,j, and k.

	i	j	k
User u	1,3	2,2	3,1
User v	3,1	1,3	2,2
User w	2,2	3,1	1,3

Table 6.1: Example

The first number of each pair in the matrix gives the ranking of items by users, the second number is the ranking of users by items. Therefore, u ranks i first, j second, k third. i ranks v first, w second, and u third, and so on. In this example, there are six possible sets of assignments. Three are stable.

One of these is achieved by giving each user his first choice, thus u is assigned to i, v to j, and w to k. Even if each item gets his last choice, the arrangement is stable. Another one is to take each item first choice, that is u assigned to k, v to i and w to j. The last stable arrangement is realized by giving each item and user its second choice: u assigned to j, v to k and w to i. All other arrangements are unstable.

6.4 Algorithm

We present here a classical one-to-one algorithm used to obtain a stable arrangement between users and item and then modify it to have stable arrangements with many items to one user.

6.4.1 Classical one-to-one algorithm

We present here a simple procedure leading to a stable set of assignments.

First step Each item is assigned to his favorite user. Each user receiving more than one item rejects all but its favorite item from among those who have been assigned to it. However, the user does not definitely choose this item yet, but puts him aside in case someone better may come along later.

Second step items who were rejected in the first step are now "proposed" to their second choices. Each user receiving new proposals chooses its favorite item among the new items and the one kept in the first step. Once again, the user rejects every items but its favorite.

Then the process goes on in the same way. items rejected at the second step are proposed to their next choices, and the users again reject all but the best proposal they have had so far.

Finally, in at most $n^2 - 2n + 2$ iterations (n being the number of items), every user will have received a proposal. Indeed, as long as any user has not been proposed to there will be rejections and new proposals. But since no item can be proposed to the same user more than once, every user is sure to get a item in due time. The resulting set of assignments is stable.

It is not necessary to have the same number of users and items. Suppose there are n items and m users with $n < m$. Then the iterative process will stop as soon as n users have received a item. In the case where $n > m$, the process ends when every item is temporary selected by an user or rejected by all of them. In both cases, the set of assignment is stable.

Obviously, there exists a symmetrical way of doing things, in which users are proposed to items. Note that generally, the solutions are not the same. Indeed, when items are proposed to users it is optimal for items and vice-versa.

6.4.2   From one-to-one to one-to-many

The idea is to follow the same principle described in the classical one-to-one algorithm and repeat it. So, now each iteration consists in finding a one-to-one stable arrangement.
The principle is the following:

First step Apply the stable mariage algorithm.

Second step When the stable arrangement is found, each user saves the item that has been attributed to him. Every item has found a user and thus removes this user from his preferences. Then, every item goes back to his new favorite user. Go back to step one and so on.

In the end, we obtain for each user a list of items ordered in a stable way, that satisfies and optimize both users and items utilities. The idea

6.5   Evaluation

The idea in this section is to compare the performance of the same system with and without the matching algorithm. Obviously, the metrics involved here are rank-based, as a matching

game has not effect the predicted ratings. However, it modifies the order of the top-N list of recommendations.

Therefore, we will compare the Mean Average Precision of the predicted list before and after the matching algorithm.

6.6 Results

We have performed our experiments on Netflix, MovieLens and Vodksater Test sets.

Dataset	MAP before	MAP after
Netflix	0.63 %	0.55%
MovieLens	0.48%	0.40%
Vodkaster	0.74%	0.66%

Table 6.2: Results before and after matching algorithm

We observe in Table 6.2 that the matching algorithm has a negative effect on the MAP. However, an interesting result is shown in Table 6.3.

We observe in Table 6.3 that the Matching Algorithm encourages the presence of less popular movies in the top-10 recommendation list. We also discover that the most popular item appears first in the recommendation list.

Firgure 6.1 display the same phenomena. We can see that the blue curve (matching) is more flat and less monotone than the orange one (no matching).

N-th recommendation	After Matching	Before Matching
1	1589	2454
2	1482	2147
3	1312	1993
4	1350	1714
5	1336	1700
6	1438	1644
7	1307	1609
8	1166	1546
9	1117	1658
10	1237	1537

Table 6.3: Average number of times the n-th recommended movie has been seen. Results before and after matching algorithm on the Netflix dataset.
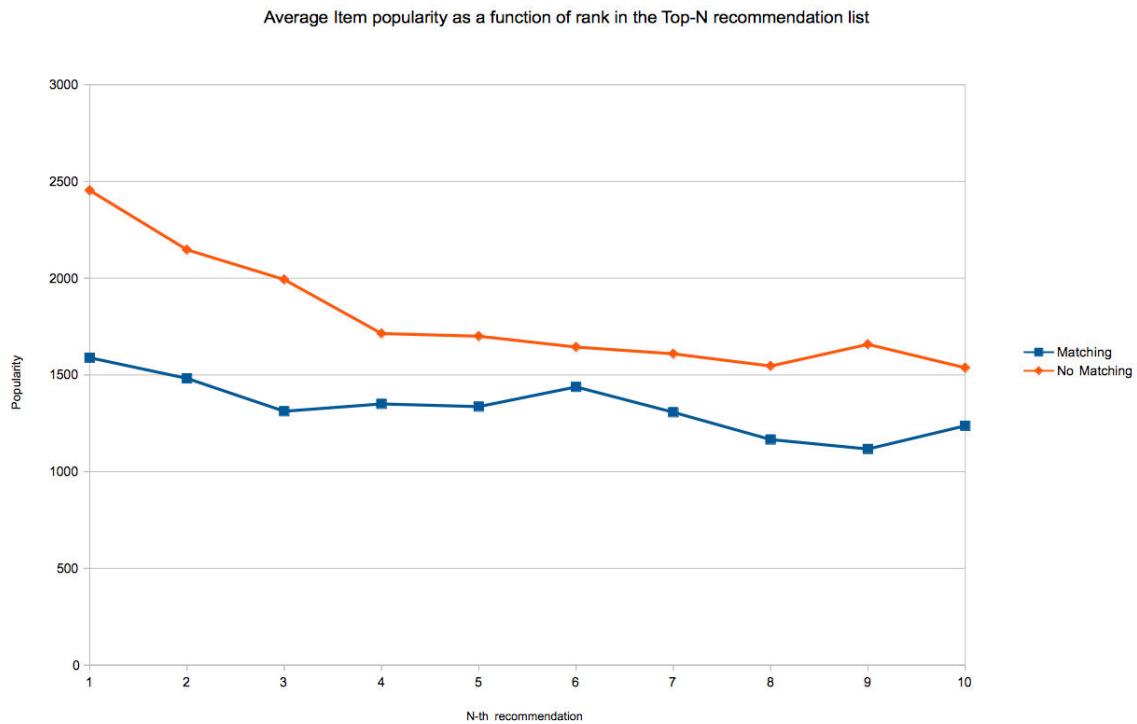
Figure 6.1: *Average Item popularity as a function of rank in the Top-N recommendation list*

6.7 Conclusion

We have presented a new approach to generate top-N recommendation list by applying a Matching Game algorithm. We show that this methods leads to a loss of performance of 10% according to the MAP metric. However, it has the interesting property of making the top-N recommendation list less "systematic", by introducing less popular items, that wouldn't have been selected without the Matching Algorithm.

DISCUSSION AND CONCLUSION

Discussion

We can not content ourselves with rating predictions only, no matter how accurate results are, from a numerical point of view. However small the margin error may be, it could put the user at risk of disappointment. This risk is even larger, especially as we are not certain to have found the evaluation criteria measuring the ability of systems to satisfy the expectations of a passive user.

The alternative we propose with argued recommendations is aimed to give relevant elements, so that the user's decision may be active, participative, collaborative, well-reasoned and based on actual real contents. This innovative approach enables us to tone down the controversial issue about evaluation.

If our system's results were poor, one could suspect us of trying to hide any weakness. But this is really not what the bill is all about. The results of current systems in the literature (including ours) are good, even very good. But we do know that perfection is clearly out of reach and the implementation of ideal criteria is impractical.

We want to include the end-user in the loop, give him the best position to auto-recommend himself what he is most likely to appreciate. This is the way we propose to adopt. This approach borrows some notions in Information Retrieval (IR) and Question Answering (QA) areas.

Navigation inside the system will be a main feature as the user will be an actor of his recommendations. And a good thing is that our system provides an accurate ranked list of items, in which the user could just click on the item and see why he ought to or not to choose it.

Conclusion

We have proposed an Adaptive Matrix Completion method that allows Recommender Systems to be highly dynamic. Experimental results showed that this method improves significantly the accuracy of predicted ratings, even if there is still a residual noise which seems unavoidable when using only rating data and no other features. Future works should now focus on extending this scheme to time-varying user- and item- features, but also investigate other matrix regularizers to automatically determine the optimal reduced rank K. Ideally, we should also introduce some meta-adaptation that allows the adaptation rates $(\alpha_1, \beta_1, \alpha_2, \beta_2)$ to vary over time.

We have also presented an innovative approach to recommendation by designing a domain-independent semantic recommender system, relying on a word based similarity function (WBS), providing textually well-argued recommendations to users. Moreover, the aesthetic preferences of users are now taken into account, not only their ratings. Additionally, the system runs in a dynamic and adaptive framework that makes it very efficient in catching instantaneously every single change.

We have described a new optimization process based on randomness through which we obtain very good results and a strong robustness. We proposed a game-theoretic approach that allows less popular items to be more represented in recommendation lists.

Finally, we have discussed how the production of textual arguments helps the end user to understand the reasons that led to the recommendation.
We do believe that calling forth the user's free-will allows us to put things into perspective about the importance of benchmarking.

As a future work, we are currently working on generating a full sentence based on extracted relevant arguments.
This is coherent with our conception of where the intelligence has to be located in the relationship between human beings and "intelligent" systems.

APPENDIX

A

QUESTIONNAIRE RESULTS - VODKASTER

194 réponses

Résumé Afficher les réponses complètes

**Consultez-vous les avis émis sur Vodkaster avant d'aller voir un film ?**
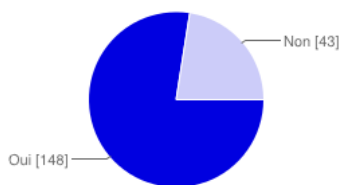
Oui	**148**	76%
Non	**43**	22%

Si oui, pour quelle(s) raison(s) ?
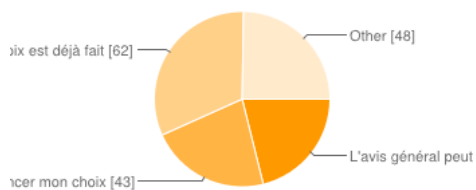
L'avis général peut influencer mon choix	**41**	21%
L'avis de mes amis Vodkaster peuvent influencer mon choix	**43**	22%
Je consulte à titre informatif mais mon choix est déjà fait	**62**	32%
Other	**48**	25%

Combien de temps s'écoule en moyenne entre le visionnage d'un film et la publication de votre micro-critique ?
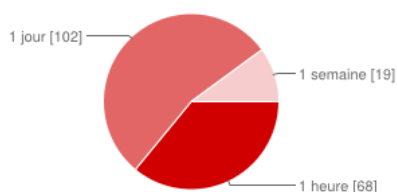
1 heure	**68**	35%
1 jour	**102**	53%
1 semaine	**19**	10%

Allez-vous regarder d'autres avis (ou notes) sur Vodkaster concernant un film avant de publier votre propre micro-critique ?
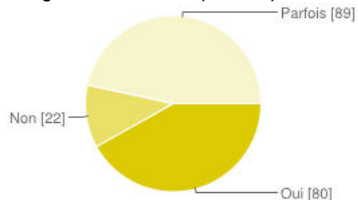
Oui	**80**	41%
Non	**22**	11%
Parfois	**89**	46%

Lors dela rédaction d'une micro-critique, de quelle manière procédez-vous ?

En un seul jet et sans ré-écriture	**14**	7%
Spontanée puis corrigée	**109**	56%
Longuement réfléchie	**67**	35%

Longuement réfléc
En un seul jet et sa
puis corrigée [109]
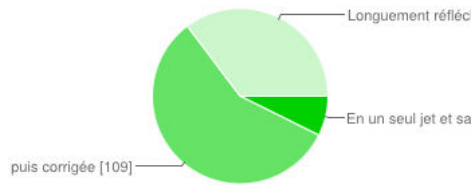
Lors de la rédaction d'une micro-critique, quel style cherchez-vous à adopter ?

La question ne se p
Other [29]
t naturellement [95]
Je cherche la form
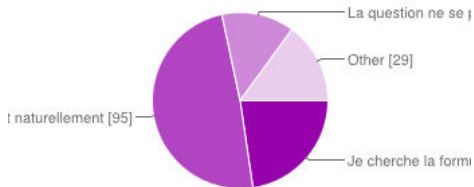
Je cherche la formule qui "fait mouche"	**44**	23%
J'écris ce qui me vient naturellement	**95**	49%
La question ne se pose pas	**26**	13%
Other	**29**	15%

De manière générale, êtes-vous gêné par la limite de 140 caractères ?
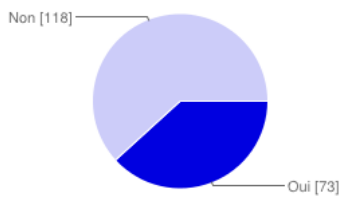
Non [118]
Oui [73]

Oui	**73**	38%
Non	**118**	61%

Comment contournez-vous cette contrainte ?

Other [82]
e micro-critique [22]
rs micro-critique [2]
Abréviations, suppr
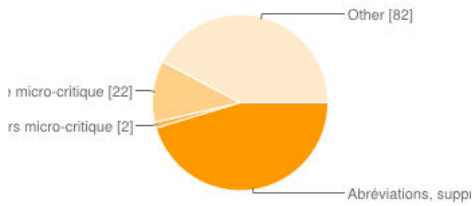
Abréviations, suppressions de certains mots	**88**	45%
Fragmenter en plusieurs micro-critique	**2**	1%
Commenter ma propre micro-critique	**22**	11%
Other	**82**	42%

Pour vous, une micro-critique est bonne si :

Elle génère un déb
Elle peut me faire c
rêté attention [118]
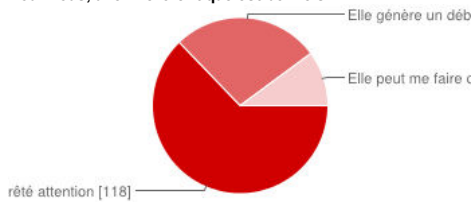
Elle permet de découvrir des aspects importants auxquels je n'aurais pas prêté attention	**118**	61%
Elle génère un débat (en commentaires)	**51**	26%
Elle peut me faire changer d'opinion	**19**	10%

Avant d'aller voir un film, vous êtes influencé par :
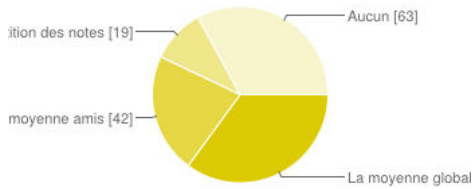
La moyenne globale	**67**	35%
La moyenne amis	**42**	22%
La répartition des notes	**19**	10%
Aucun	**63**	32%

Hors du contexte Vodkaster, vous êtes influencé par :
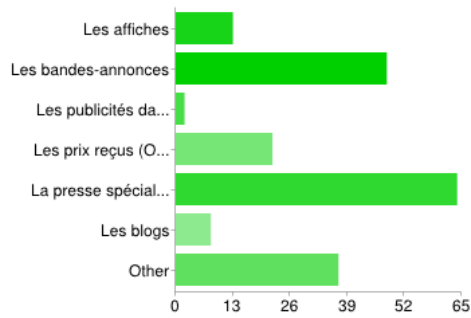
Les affiches	**13**	7%
Les bandes-annonces	**48**	25%
Les publicités dans la presse	**2**	1%
Les prix reçus (Oscar, César, Palme d'Or...)	**22**	11%
La presse spécialisée	**64**	33%
Les blogs	**8**	4%
Other	**37**	19%

Le plus souvent, vous allez sur le site Vodkaster depuis :
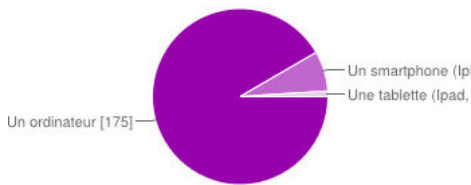
Un ordinateur	**175**	90%
Un smartphone (Iphone, Android, Blackberry...)	**14**	7%
Une tablette (Ipad, Android...)	**2**	1%

Lorsque vous êtes sur Vodkaster, vous êtes :
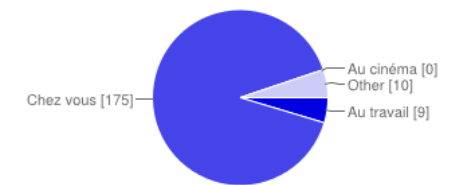
Au travail	**9**	5%
Chez vous	**175**	90%
Au cinéma	**0**	0%
Other	**10**	5%

Parlez-vous des micro-critiques en dehors de votre cercle d'amis Vodkaster ?
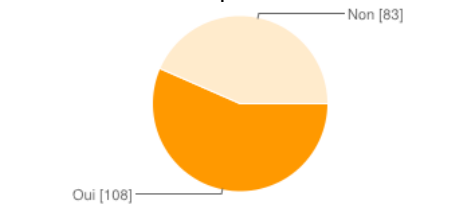
Oui	**108**	56%
Non	**83**	43%

Si oui, vous en parlez à :

Vos collègues de travail	**11**	6%
Vos amis non-adhérents de Vodkaster	**77**	40%
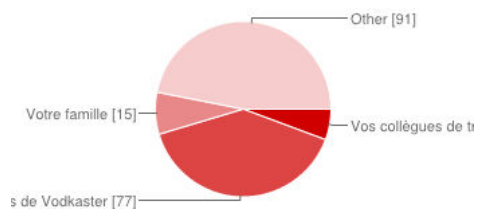
Votre famille	**15**	8%
Other	**91**	47%

Combien de films voyez-vous par an ?

654764   250   150   100   150   Un peu plus d'une centaine environ   200   180   82 en 2012   120   200   50   80   250   300   entre 10 et 30   50   100   220   150   200   150   300   250   365,6   Environ 300   200   environ 250   200   100   plus de 100 sorties en salles, plus de 10   ...

Lisez-vous des magazines de Cinéma ?
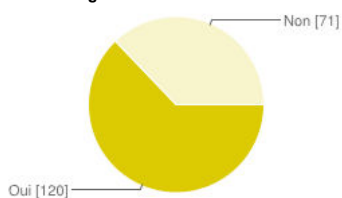
Oui	**120**	62%
Non	**71**	37%

Si oui, lesquels ?

Cahiers du cinéma   So Film   Studio Cine Live, Premiere, So Film, Cinema Teaser   Positif et les Cahiers   So Film, les Cahiers, Positif.   Cahiers du CInéma   Abonné à Studio Ciné Live, So Film, Les Cahiers du Ciné   ...

Quel est LE critique que vous préférez ?

Cyril Béghin   Je ne lis pas de critiques cinéma (seulement des dossiers, reportages et interviews)   Moi. Pas le meilleur, mais le plus fiable.   Aucun   Michel Ciment   Serge Daney   Joachim Lepastier   Aucune idée   Phi   ...

Combien de fois par an participez-vous à des manifestations culturelles autres que cinématographiques (expositions, théâtre, concerts, danse...) ?

5   10   25   10   Une dizaine   20   30   52   25   15   2   10   10   2   2   15   10   20   Entre 12 et 24 fois   30   50   10 ~   Environ 20   5   10   2   10   environ une dizaine   1   5   0   20   10   0   8   1 à 2 fois grand maxi   15   10 fois   12   2   Je ne sais pas   Au moins 50 soirées/concerts   20   Rarement   ...


Catégorie socio-professionnelle

Agriculteur exploitant	**0**	0%
Artisan, commerçant et chef d'entreprises	**4**	2%
Cadre, professions intellectuelles supérieures	**36**	20%
Professions intermédiaires	**3**	2%
Employé	**28**	16%
Ouvrier	**0**	0%
Retraité	**1**	1%
Étudiant	**99**	55%
Other	**14**	8%

Les utilisateurs peuvent cocher plusieurs cases, donc les pourcentages peuvent être
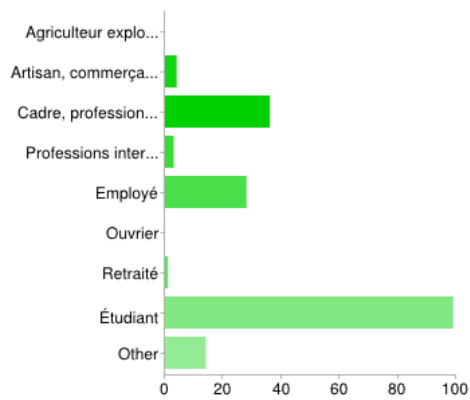
supérieurs à 100 %.

Niveau de diplôme
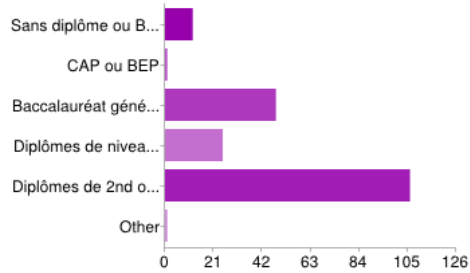
Sans diplôme ou Brevet des collèges
CAP ou BEP
Baccalauréat général, technologique ou professionnel
Diplômes de niveau Bac plus 2 (DUT, BTS, DEUG, écoles des formations sanitaires ou sociales,...)
Diplômes de 2nd ou 3e cycle universitaire (licence, maîtrise, master, DEA, DESS, doctorat) ou diplômes de gran
Other

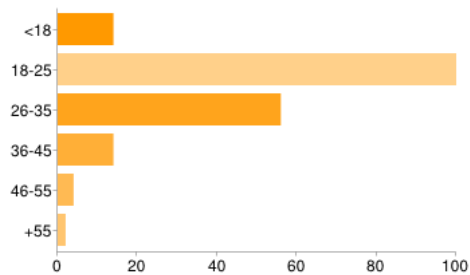Les utilisateurs peuvent cocher plusieurs cases, donc les pourcentages peuvent être supérieurs à 100 %.

Sexe
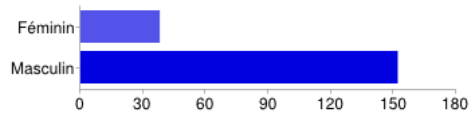
Féminin	**38**	20%
Masculin	**152**	80%

Les utilisateurs peuvent cocher plusieurs cases, donc les pourcentages peuvent être supérieurs à 100 %.

Âge

<18	**14**	7%
18-25	**100**	53%
26-35	**56**	29%
36-45	**14**	7%
46-55	**4**	2%
+55	**2**	1%

Les utilisateurs peuvent cocher plusieurs cases, donc les pourcentages peuvent être supérieurs à 100 %.

Nombre de réponses quotidiennes

LIST OF FIGURES

LIST OF TABLES

BIBLIOGRAPHY

(Adomavicius et Tuzhilin., 2005) G. Adomavicius et A. Tuzhilin., 2005. Toward the next generation of recommender systems: A survey of the state of the art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, pp. 734–749. 9, 14, 21

(Agarwal et al., 2010) D. Agarwal, B.-C. Chen, et P. Elango, 2010. Fast online learning through offline initialization for time-sensitive recommendation. Dans les actes de *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, 703–712. ACM. 50

(Anderson, 1953) R. L. Anderson, 1953. Recent advances in finding best operating conditions. Dans les actes de *J. Amer. Statist. Assoc*, 789–798. 25

(Armstrong, 2001) J. S. Armstrong, 2001. *Principles of Forecasting, A Handbook for Researchers and Practitioners.* Kluwer Academic Publishers. 5

(Bell et Koren, 2007) R. Bell et Y. Koren, 2007. Lessons from the netflix prize challenge. *SIGKDD Explorations*. 15

(Bell et al., 2007) R. Bell, Y. Koren, et C. Volinsky., 2007. The bellkor 2008 solution to the netflix prize. *The Netflix Prize*. 24, 28

(Bilgic et Mooney, 2005) M. Bilgic et R. Mooney, 2005. Explaining recommendations: Satisfaction vs. promotion. Dans les actes de *Proceedings of the Workshop Beyond Personalization 2005*. 17

(B.L. Bowerman, 2004) A. K. B.L. Bowerman, R.T. O'Connell, 2004. *Forecasting, time series and regression: An applied approach*. 28

(Bonhard, 2004) P. Bonhard, 2004. Improving recommender systems with social networking. Dans les actes de *Proceedings Addendum of CSCW*. 13

(Breese et Kadie,) H. D. Breese, J. et C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. Dans les actes de *14th Conference on Uncertainty in Artificial Intelligence*. 21

(Bridge et al., 2006) G. Bridge, M. Göker, L. McGinty, et B. Smyth, 2006. Case-based recommender systems. *The Knowledge Engineering review*. 12

(Brusilovsky, 1996) P. Brusilovsky, 1996. Methods and techniques of adaptive hypermedia. *User Modeling and User-Adapted Interaction.*. 7

(Buchanan et al., 2005) G. Buchanan, S. Cunningham, A. Blandford, J. Rimmer, et C. Warwick., 2005. Information seeking by humanities scholars. Dans les actes de *Proceedings of the European Conference on Digital Libraries*. 18

(Burke, 2002) R. Burke, 2002. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction.*. 6, 9, 13

(Burke, 2007) R. Burke, 2007. Hybrid web recommender systems. *The Adaptive Web*, 377–408. 7

(Codina et Ceccaroni, 2010) V. Codina et L. Ceccaroni, 2010. Taking advantage of semantics in recommendation systems. Dans les actes de *Proceedings of the 13th International Conference of the Catalan Association for A.I*, 163–172. 70

(Degemmis et al., 2007) Degemmis, P. M., Lops, et G. Semeraro, 2007. A content-collaborative recommender that exploits wordnet-based user profiles for neighborhood formation. user modeling and user- adapted interaction. *The Journal of Personalization Research (UMUAI) 17*(3), 217–255. 11

(Ethis, 2004) E. Ethis, 2004. *Pour une po(i)etique du questionnaire en sociologie de la culture. Le spectateur imaginé*. Logiques sociales SOCIOLOGIE EUROPE France. 54

(Funk, 2006) S. Funk, 2006. Netflix update: Try this at home. 16

(Gaillard et al., 2013a) J. Gaillard, M. El-Beze, E. Altman, et E. Ethis, 2013a. Flash reactivity: adaptive models in recommender systems. Dans les actes de *Proceedings of the 2013 International Conference on Data Mining (DMIN), WORLDCOMP*. 51, 72

(Gaillard et al., 2013b) J. Gaillard, M. El-Beze, E. Altman, et E. Ethis, 2013b. Well-argued recommendation: adaptive models based on words in recommender systems. Dans les actes de *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 70, 71

(Gale et Shapley, 1962) D. Gale et L. Shapley, 1962. College admissions and the stability of marriage. *The American Mathematical Monthly 68*(1). 82

(Golub et Reinsch, 1970) G. Golub et C. Reinsch, 1970. Singular value decomposition and least squares solutions. Dans les actes de *Numerische Mathematik*. 16

(Herlocker et J., 2001) J. Herlocker et K. J., 2001. Content-independent, task-focused recommendation. *IEEE Internet Computing*. 10

(Herlocker et al., 1999) J. Herlocker, J. Konstan, A. Borchers, et J. Riedl, 1999. An algorithmic framework for performing collaborative filtering. Dans les actes de *In proceedings of the 1999 Conference on Research and Development in Information Retrieval*. 18

(Herlocker et al., 2000) J. Herlocker, J. Konstan, et J. Riedl, 2000. Explaining collaborative filtering recommendations. Dans les actes de *ACM Conference on CSCW*. 6, 10, 12, 13

(Herlocker et al., 2002) J. Herlocker, J. Konstan, et J. Riedl, 2002. An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Information Retrieval*. 10

(J.E. Hanke, 1995) A. R. J.E. Hanke, 1995. *Business forecasting*. 28

(Johnson et Johnson, 1993) H. Johnson et P. Johnson, 1993. Explanation facilities and interactive systems. Dans les actes de *In proceedings of International Workshop on Intelligent Interfaces*. ACM Press. 18

(Karnopp, 1963) D. C. Karnopp, 1963. Random search techniques for optimization problems. Dans les actes de *Automatica*, 111–121. 25

(Koenemann et Belkin, 1996) J. Koenemann et N. Belkin, 1996. A case for interaction: A study of interactive information retrieval behavior and effectiveness. Dans les actes de *Proceedings of the Human Factors in Computing Systems Conference*. 18

(Konstan et Riedl., 2002) J. A. Konstan et J. Riedl., 2002. *Collaborative filtering: Supporting social navigation in large, crowded infospaces.* Springer Verlag. 13

(Koren, 2008) Y. Koren, 2008. Factorization meets the neighborhood: A multifaceted collaborative filtering model. Dans A. Press (Ed.), *4th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, pp. 426–434. 16

(Koren, 2010) Y. Koren, 2010. Collaborative filtering with temporal dynamics. *Communications of the ACM 53*(4), 89–97. 50

(Krause, 1987) E. F. Krause, 1987. *Taxicab Geometry.* Dover. 52

(Linden et al., 2003) G. Linden, B. Smith, et J. York, 2003. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*. 11, 13

(Lu et al., 2009) Z. Lu, D. Agarwal, et I. S. Dhillon, 2009. A spatio-temporal approach to collaborative filtering. Dans les actes de *Proceedings of the third ACM conference on Recommender systems (RecSys)*, 13–20. ACM. 50

(Magnini et Strapparava, 2001)   B. Magnini et C. Strapparava, 2001. Improving user modelling with content-based techniques. Dans les actes de *Proceedings of the 8th International Conference of User Modeling*. 11

(Maidel et al., 2008)   V. Maidel, P. Shoval, B. Shapira, et M. Taieb-Maimon, 2008. Evaluation of an ontology-content based filtering method for a personalized newspaper. Dans les actes de *RecSys'08: Proceedings*, 91–98. 70

(McNee et al., 2006)   S. McNee, J. Riedl, et J. Konstan, 2006. Making recommendations better: An analytic model for human-recommender interaction. Dans les actes de *Proceedings of the ACM Human Factors in Computing Systems Conference*. 18

(Meyer., 2012)   F. Meyer., 2012. *Recommender systems in industrial contexts.* Thèse de Doctorat, University of Grenoble, France. 6, 21

(Nelson, 1970)   P. Nelson, 1970. Information and consumer behavior. *The Journal of Political Economy.*. 5

(Pacitti et al., 2011a)   E. Pacitti, F. Draidi, et B. Kemme, 2011a. P2prec: A p2p recommendation system for large-scale data sharing. *T. Large-Scale Data- and Knowledge-Centered Systems 3*, 87–116. 13

(Pacitti et al., 2011b)   E. Pacitti, F. Draidi, D. Parigot, et G. Verger, 2011b. P2prec: a social-based p2p recommendation system. Dans les actes de *Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, October 24-28, 2011*, 2593–2596. 13

(Patarek, 2007)   A. Patarek, 2007. Improving regularized singular value decomposition for collaborative filtering,. Dans les actes de *KDD Cup and Workshop,*, pp. 39–42. ACM Press. 16

(Pine, 1993)   J. Pine, 1993. *Mass Customization.* Harvard Business School Press. 17

(Rao et Talwar, 2008)   N. Rao et V. Talwar, 2008. Application domain and functionnal classification of recommender systems a survey. *Journal of library and information technology (Desidoc)*. 21

(Rastrigin, 1963)   L. A. Rastrigin, 1963. The convergence of the random search method in the extremal control of a many-parameter system. Dans les actes de *Automat. Remote Control*, Numéro 24, 1337–1342. 25

(Recht et al., 2010)   B. Recht, M. Fazel, et P. Parrilo., 2010. Guaranteed minimum rank solutions of matrix equations via nuclear norm minimization. *SIAM Review*. 55

(Recht et Re, 2011)   B. Recht et C. Re, 2011. Parallel stochastic gradient algorithms for large-scale matrix completion. 17

(Recht et Ré, 2013)   B. Recht et C. Ré, 2013. Parallel stochastic gradient algorithms for large-scale matrix completion. *Mathematical Programming Computation 5*(2), 201–226. 55

(Rendle et Schmidt-thieme, 2008) S. Rendle et L. Schmidt-thieme, 2008. Online-updating reg-ularized kernel matrix factorization models for large-scale recommender systems. Dans les actes de *Proceedings of the 2008 ACM conference on Recommender systems (RecSys) ACM*. 50

(Resnick et Hal., 1997) P. Resnick et R. V. Hal., 1997. Recommender systems (introduction to special section.). *Communications of the ACM*. 21

(Resnick et al., 1994) P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, et J. Riedl, 1994. Grou-plens: An open architecture for collaborative filtering of netnews. Dans les actes de *Proceed-ings of ACM 1994 Conference on Computer Supported Cooperative Work*. 10

(Resnick et Varian, 1997) P. Resnick et H. R. Varian, 1997. Recommender systems. *Communi-cations of the ACM*. 6, 8

(Ricci, 2002) F. Ricci, 2002. Travel recommender systems. *IEEE Intelligent Systems*. 6

(Rich, 1979) E. Rich, 1979. User modeling via stereotypes. *Cognitive Science*, 328–353. 5

(Riedl et Konstan, 2002) J. Riedl et J. A. Konstan, 2002. *Word of Mouse: The Hidden Marketing Power of Collaborative Filtering*. Warner Business Books. 13

(Said et al., 2013) A. Said, S. Berkovsky, et E. D. Luca, 2013. Movie recommendation in context. *ACM Trans. Intell. Syst. Technol. (TIST)* 4(1). 28

(Salakhutdinov et Mnih, 2008) R. Salakhutdinov et A. Mnih, 2008. Probabilistic matrix fac-torization. *Advances in Neural Information Processing Systems 20 (NIPS 07)*, pp. 1257–1264. 16

(Salton, 1989) G. Salton, 1989. Automatic text processing. *Addison-Wesley*. 5

(Sarma, 1990) M. Sarma, 1990. On the convergence of the baba and dorea random optimization methods. *Journal of Optimization Theory and Applications*. 26

(Sarwar, 2000) B. Sarwar, 2000. Application of dimensionality reduc- tion in recommender system—a case study. Dans A. Press (Ed.), *Workshop on Web Mining for e-Commerce: Challenges and Opportunities (WebKDD)*. 16, 18

(Sarwar et al., 2001) B. Sarwar, G. Karypis, J. Konstan, et J. Reidl, 2001. Item-based collaborative filtering recommendation algorithms. Dans les actes de *Proceedings of the 10th International Conference on World Wide Web*. 21

(Sarwar et al., 2002) B. Sarwar, G. Karypis, J. Konstan, et J. Riedl, 2002. Incremental singular value decomposition algorithms for highly scalable recommender systems. Dans les actes de *5th International Conference on Computer and Information Technology (ICCIT)*. 17

(Schafer et al., 1999) J. Schafer, J. Konstan, et J. Riedl, 1999. Recommender systems in electronic commerce. Dans les actes de *Proceedings of the ACM Conference on Electronic Commerce*. 8

(Schafer et al., 2001)   J. Schafer, J. Konstan, et J. Riedl, 2001. E-commerce recommender applications. *Data Mining and Knowledge Discovery*. 8

(Semeraro et al., 2009)   G. Semeraro, P. Basile, de Gemmis M., et P. Lops, 2009.   *Handbook of Research on Digital Libraries: Design, Development and Impact*, Chapter User Profiles for Personalizing Digital Libraries. Y.L. Theng and S. Foo and D.G.H. Lian and J.C. Na. 11

(Setten, 2005)   M. Setten, 2005.   *Supporting People in Finding Information-Hybrid Recommender Systems and Goal-Based Structuring.* Thèse de Doctorat, Telematica Instituut. 9

(Shani et al., 2005)   G. Shani, D. Hackerman, et R. Brafman, 2005. An mdp-based recommender system. *Journal of Machine Learning Research*. 7

(Shardanand et Maes, 1995)   U. Shardanand et P. Maes, 1995. Social information filtering: Algorithms for automated "word of mouth". Dans les actes de *Proceedings of the Human Factors in Computing Systems Conference*. 10, 18

(Stefani et Strapparava, 1998)   A. Stefani et C. Strapparava, 1998. Personalizing access toweb sites: The siteif project. Dans les actes de *Proc. of second Workshop on Adaptive Hypertext and Hypermedia*. 11

(Stern et al., 2009)   D. H. Stern, R. Herbrich, et T. Graepel, 2009. Matchbox: large scale online bayesian recommendations. Dans les actes de *Proceedings of the 18th international conference on World wide web (WWW)*, 111–120. ACM. 50

(Su et Khoshgoftaar., 2009)   X. Su et T. M. Khoshgoftaar., 2009. A survey of collaborative filtering techniques. *Adv. Artificial Intellegence*. 28

(Tan et al., 2005)   P. Tan, M. Steinbach, et V. Kumar., 2005. *Introduction to Data Mining.* Addison-Wesley. 70

(Xiong et al., 2010)   L. Xiong, X. Chen, T.-K. Huang, J. G. Schneider, et J. G. Carbonell, 2010. Temporal collaborative filtering with bayesian probabilistic tensor factorization. Dans les actes de *Proceedings of the SIAM International Conference on Data Mining (SDM)*, Volume 10, 211–222. SIAM. 50

(Ziegler et al., 2005)   C. Ziegler, S. McNee, J. Konstan, et G. Lausen., 2005. Improving recommendation lists through topic diversification. Dans les actes de *Fourteenth International World Wide Web Conference*. 21