Universidad Autónoma de Madrid
Escuela Politécnica Superior
Departamento de Ingeniería Informática

Temporal Models in Recommender Systems: An Exploratory Study on Different Evaluation Dimensions

Pedro G. Campos Soto
Supervisor: Fernando Díez Rubio
Co-supervisor: Manuel Sánchez-Montañés

Trabajo Fin de Máster

Programa Oficial de Posgrado en Ingeniería Informática y de Telecomunicación
Universidad Autónoma de Madrid
Junio de 2011

Abstract

A Recommender System (RS) is a computer program able to identify specific objects for different user interests. Given that many RS have been operating for years, temporal information as a source to obtain better recommendations is acquiring more importance. The currently active research field of RS, has tried to incorporate this information in the form of new recommendation algorithms. However, a common evaluation framework for testing improvements on this area is still missing; most proposals have been developed for and tested under specific (and different) datasets, circumstances and metrics, making it difficult to fairly compare them.

This work aims to help establishing a better perspective of the impact of techniques that deal with temporal information on RS. An extensive review of published work on the subject was carried out, including not only time aware techniques, but also the basic techniques upon which time aware extensions are built. A considerable proportion of the techniques under study and metrics for their evaluation were implemented, and a rigorous evaluation protocol scheme was developed, in order to allow the usage of the different techniques under a common experimental setting, which included two common recommendation tasks (rating prediction and top-N recommendation). We assessed the recommendations' results obtained with the different implemented recommendation algorithms on five different evaluation dimensions (statistical accuracy, decision support accuracy, novelty, diversity and coverage), including six different metrics (RMSE, Precision, AUC, Self-Information, Intra List Similarity and Interest Coverage).

Results show that, differently to what could be expected, not all time-aware algorithms were able to outperform their time-unaware counterparts, in particular with respect to accuracy on rating prediction (statistical accuracy), which is somewhat unexpected given that, in general, the main motivation for the elaboration of such extensions is accuracy increase. Moreover, the behavior of algorithms on the assessed metrics varies notably, and no particular technique can be considered as "the best" across the different evaluation dimensions. These findings remarks 1) the importance of establishing a common and rigorous evaluation scheme when different algorithms are to be compared and 2) the most suitable recommendation algorithm will depend on the particular task at hand and evaluation dimension of interest.

To finalize, we identify interesting research topics related to this work, some of which we hope to address in the near future.

Contents

List of Figures

Chapter 1. Introduction

1.1 Motivation

A recommender system (RS) is a computer program able to identify specific objects for different user interests, based on several information sources which the system can access. Nowadays the creation of new and better recommendation algorithms is an active field of research and development, as shown by the existence of scientific conferences dedicated exclusively to this topic as, for example, the *ACM Conference on Recommender Systems*[1]. There have been also competitions sharing big money prizes, as the *Netflix Prize*[2] one, emphasizing practical aspects of the recommendation process itself. The analysis and development of RS is a very active research area at the present time, encompassing a diversity of aspects as, for example, those related with data modeling, those related with the accuracy and effectiveness of recommendation results, or with temporal aspects, as will be detailed in this work.

Generally speaking, the most valued goal in this field has been the effectiveness of recommendations, measured usually in terms of the accuracy of ratings that the system is able to predict, with respect to ratings given by the user. In fact, the most used metrics for the evaluation of RS in the literature correspond to variants of the difference between those values, as *MAE* (*Mean Absolute Error*) and *RMSE* (*Root Mean Squared Error*)[Herlocker et al., 2004]. Because of this, most of the research has focused on techniques that allow RS to increase their accuracy, as for example Matrix Factorization [Koren et al., 2009]. However, there are other aspects that may be equally important when the goal is that users find useful recommendations, as the novelty of recommendations (showing items unknown but interesting to users). Therefore, there exists the need of improving other metrics different to accuracy, as the aforementioned novelty, or the diversity of recommendations [Herlocker et al., 2004].

One characteristic with increasing importance is the data volume to be processed by the RS. For example, the dataset used in the *Netflix Prize* contains more than 100M ratings given by 480.189 users to a set of 17.770 movies (and this is just a part of the total dataset of *Netflix*). Traditional algorithms as *k- Nearest Neighbors* (kNN) [Herlocker et al., 2002] require excessive execution time. There are many proposals to deal with this scalability problem, related mainly with dimensional reduction of data [Sarwar et al., 2000]. Other tested technique is clustering [Ungar and Foster, 1998], which has high scalability, but with more trade-offs on accuracy as reported on literature [Su and Khoshgoftaar, 2009].

Given that many RS have been operating for years, the temporal dimension is acquiring more importance. One example of that is the change of tastes (evolution) of users through time. Many new algorithms try to incorporate this information effectively [Ding and Li,

[1] http://recsys.acm.org/
[2] http://www.netflixprize.com/

2005; Ding et al., 2006; Lee et al., 2008; Lee et al., 2009; Koren, 2009a]. The existence of multiple temporal data mining algorithms give the possibility to perform additional improvements, taking advantage of the potential of this dimension of information. On the other hand, most proposals of time aware RS have been developed for and tested under specific (and different) datasets, circumstances and metrics, making it difficult to fairly compare them; a common evaluation framework for testing improvements on this area is still missing.

From the foregoing it can be observed the need of rigorously studying how the application of different (and novel) techniques that aim to deal properly with temporal information, as clustering or matrix factorization, impacts on RS results not only in terms of accuracy, but on other less studied metrics as coverage, novelty or diversity. The present work analyses the state of the art on different techniques used on RS which incorporates temporal data, studying their influence on a variety of metrics, some of them not used commonly in the evaluation of recommender systems under a common evaluation protocol, in order to have a better perspective of the impact of the techniques over different aspects of interest to deal with. In doing so, the advantages and restrictions of applications of each technique are detailed, which allow readers to have a better understanding of the implications of using each specific technique.

1.2 Problem definition

Traditionally, the recommendation problem has been formulated as the estimation of *ratings* for *items* that have not been *used* [3] by a *user* using, as primary input, ratings given by this user to other items, and maybe some other information if available and the system is able to take advantage of it. Such estimates for unrated items allow to recommend the item(s) with the highest estimated rating(s) to the user [Adomavicius and Tuzhilin, 2005; Linden et al., 2003]. In this section we formalize these concepts, and the related *time-aware* RS notion.

As elegantly described in [Adomavicius and Tuzhilin, 2005], in general a utility function can be estimated that measures the usefulness of an item to a user. Let $\mathcal{U}$ be the set of users and $\mathcal{I}$ the set of items in the system. Let $\mathcal{F}$ be the utility function:

$$\mathcal{F}: \mathcal{U} \times \mathcal{I} \rightarrow \mathcal{R} \quad \textbf{(1)}$$

where $\mathcal{R}$ is a totally ordered set ($\mathcal{R}$ is usually but not necessarily a subset of $\mathbb{R}$).

Definition: The *rating prediction* problem consists in the estimation of $\mathcal{F}$ for each $u \in \mathcal{U}$ and each $i \in \mathcal{I}$ whose utility is unknown a-priori:

$$\forall u \in \mathcal{U} \wedge \forall i \in \mathcal{I}, \hat{r}_{u,i} = \mathcal{F}(u,i) \quad \textbf{(2)}$$

where $\hat{r}_{u,i}$ is the predicted utility (predicted rating).

[3] Here *used* involves several different senses or activities like, for example, seen, heard, tasted, purchased, etc. depending on the type of the item being recommended.

Conceptually, if the function $\mathcal{F}$ can be computed for the whole pairs of the $\mathcal{U} \times \mathcal{I}$ domain, a RS is able to select for recommendation the item with the highest utility for each user:

$$\forall u \in \mathcal{U}, i^*(u) = \arg\max_{i \in \mathcal{I}} \mathcal{F}(u, i) \quad \textbf{(3)}$$

This notion can be extended to generate a *top-N* recommendation list, that is, selecting for recommendation an ordered set consisting on the *N* items with highest utility.

Definition: The *top-N recommendation* problem consists in determining, for each user u, the set of items $\mathcal{I}_N^*(u)$ such that[4,5]:

$$\forall u \in U, \mathcal{I}_N^*(u) = \bigcup_{j=1}^{N} i_j^*(u) : i_j^*(u) = \arg\max_{i \in \mathcal{I} - \mathcal{I}_{j-1}^*(u)} \mathcal{F}(u, i) \quad \textbf{(4)}$$

with:

$$\mathcal{I}_0^*(u) = \emptyset$$

The main source of information for computing $\mathcal{F}$ is the set of utility values (ratings) that users have previously assigned to some items (the so called rating matrix R). Different configurations of how R is analyzed, the kind of data in R, and what additional sources of information are used, give rise to different types of RS. For example, some RS use information about declared characteristics of items (e.g. genre and director in case of movies) to find items with similar characteristics to those highly rated by a user, assuming users like items with similar characteristics, whilst others focus on pure rating information, which may be *explicit* or *implicit*.

Typically, the additional information is managed as *profiles*. For instance, each element of the user space can be defined with a *user profile* that includes various user characteristics, such as age, gender, income, marital status, etc. In the simplest case, the profile can contain only a single (unique) element, commonly the user ID. Similarly, each element of the item space can be defined with a set of their declared characteristics. In this work we focus on RS that make use of temporal information, generally in the form of time-stamped ratings, that is, taking advantage of knowing *when* ratings were done. This information can be managed as a *rating profile*, which contains the rating value and additional *rating context* information (information about the context in which the rating was done) which in our case will contain the rating *time*, although other contextual information may be added, e.g. the *place* of rating, the user *mood* at rating time, the *company*, etc. This way recommendation algorithms may, for example, discard old rating data, find similarities between users or items taking into account how ratings evolve through time, etc.

When using this additional contextual information, not only the way the estimations are computed can be modified, but the whole utility notion may be affected, as the RS could

[4] In these definitions we are assuming implicitly that items already used by the target user have lower utility (or no utility at all) than items not already used. Indeed, in many practical implementations, items already used by u are discarded for selection of $\mathcal{I}_N^*$. However, it should be noted that this assumption may not hold on particular domains and situations, e.g. after a long time period since the user used the items.

[5] There are other ways of computing the recommendation list, e.g. [Rendle et al., 2009b].

be able to estimate the utility of items not only considering the particular user who needs the recommendation. In fact, now the utility may be differentiated according to the *time*, *place*, etc. for which the recommendation is asked[6]. Adomavicious et al. tackles this as a *multidimensional* recommendation problem (as opposed to the traditional two-dimensional $\mathcal{U} \times \mathcal{I}$ problem)[Adomavicius et al., 2005], in which the *recommendation space* $\mathcal{S}$ is modeled as a set of dimensions $\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_n$, each dimension $\mathcal{D}_i$ being a subset of a Cartesian product of some attributes. Examples of dimensions $\mathcal{D}_i$ are a user profile (e.g. $user = \{user\_id, gender, age\}$), an item profile (e.g. in the case of movies $item = \{movie\_id, genre, director\}$), time of recommendation (e.g. $time = \{hour\_of\_day, day\_of\_week, month, year\}$), location (e.g. $location = \{city, country\}$), or mood of the user (e.g. $mood = \{mood\_id\}$).

This way, the utility function $\mathcal{F}$ can be defined over the space $\mathcal{S}$ as [Adomavicius et al., 2005]:

$$\mathcal{F}: \mathcal{D}_1 \times \mathcal{D}_2 \times \cdots \times \mathcal{D}_n \rightarrow \mathcal{R} \quad \textbf{(5)}$$

The above implies that the utility (rating) a user assigns to an item depends, for example, on the location where the item is used, at what time, etc., and thus, the estimation of utilities should also depend of such dimensions. Particularly, a RS that takes into account the *time* dimension for recommendation will use a recommendation space of the form $\mathcal{S} = \mathcal{U} \times \mathcal{I} \times \mathcal{T}$, where $\mathcal{T}$ is the time dimension.

Definition: The *time-dependant rating prediction* problem consists in the estimation of the utility of items for users at particular times:

$$\forall u \in \mathcal{U} \wedge \forall i \in \mathcal{I} \wedge \forall t \in \mathcal{T}, \hat{r}_{u,i,t} = \mathcal{F}(u, i, t) \quad \textbf{(6)}$$

Similarly, the *time-dependant top-N recommendation* problem consists in the recommendation of the highest utility items for users at particular times:

$$\forall u \in U \wedge t \in \mathcal{T}, \mathcal{I}_N^*(u,t) = \underset{i \in \mathcal{I} - \mathcal{I}_{j-1}^*(u,t)}{\arg\max} \mathcal{F}(u,i,t) \quad \textbf{(7)}$$

with:

$$\mathcal{I}_0^*(u,t) = \emptyset$$

Definition: A *time-aware* RS is a RS which uses any form of temporal information to solve a recommendation problem.

Note that recommenders defined by (2)-(4) as much as those defined by (6)-(7) can be considered time-aware RS, if they take advantage of any kind of temporal information for utility estimation, either as data in profiles for estimation computation and/or as target recommendation context.

[6] Note there is a difference between using contextual information in profiles during utility estimation, and estimating utility for a particular recommendation context.

1.3 Research goals

The main goal of this work is to carry out an exploratory study of state-of-the-art techniques in data mining and machine learning fields applied to recommender systems, emphasizing on techniques allowing to deal with temporal information. We aim to establish their main characteristics and restrictions, and the possible advantages from their usage, on different evaluation perspectives. This way, this work has the following specific research goals:

- To carry out a state-of-the-art revision of techniques used in recommender systems, especially those able to handle temporal information.
- To apply some of the state-of-the-art data mining techniques for recommendation.
- To analyze possible benefits derived from the application of clustering, as a traditional data mining technique, on the elaboration of recommendations.
- To study the impact of techniques on different metrics related with the outcome of recommender systems.

1.4 Outline

This work is structured as follows: This chapter establishes the motivation and problem definition, and includes a list of publications related with Master's work of the candidate. Chapter 2 presents a review of the state-of-the-art on RS and related techniques, with focus on time-aware recommendation. Also a brief review of recommendation metrics is included. Chapter 3 details the different techniques implemented for this work, some of them with a particular emphasis on the time-dependant recommendation problem. Chapter 4 describes the experimental setting and results obtained from experimentation with the techniques presented in Chapter 3. Finally, 4.4.4 establishes main conclusions derived from this work and devises related future lines of research.

1.5 Publications

The following publications are related to this work:

1. Pedro G. Campos, Fernando Díez. La Temporalidad en los Sistemas de Recomendación: Una revisión actualizada de propuestas teóricas. *I Congreso Español de Recuperación de Información (CERI 2010)*. Madrid, España, 15 y 16 de junio.
2. Pedro G. Campos, Alejandro Bellogín, Fernando Díez, J. Enrique Chavarriaga. Simple Time-Biased KNN-based Recommendations. *Proceedings of the Workshop on Challenge on Context-aware Movie Recommendation, CAMRA 2010*, part of the *4th ACM Conference on Recommender Systems, RecSys2010*. Barcelona, Spain, September 30th, 2010.
3. Pedro G. Campos, Fernando Díez, Manuel Sánchez-Montañés. Towards a More Realistic Evaluation: Testing the Ability to Predict Future Tastes of Matrix Factori-

zation-based Recommenders. Submitted as short paper to *5th ACM Conference on Recommender Systems, RecSys2011*.

There are also other publications that were completed during the time period of the Master. Although they are not directly within the scope of this work, all of them were developed within the scope of subjects of the Master, which reflects an adequate degree of assimilation of general topics covered in the program:

1. Pedro G. Campos, Ruth Cobos. Towards Awareness Services Usage Characterization: Clustering Sessions in a Knowledge Building Environment. In Y. Bi and M.-A. Williams (Eds.): *KSEM 2010, LNAI 6291*, pp. 270-281. Springer, Heidelberg (2010).
2. Fernando Díez, J. Enrique Chavarriaga, Pedro G. Campos, Alejandro Bellogín. A. Movie Recommendations based in explicit and implicit features extracted from the Filmtipset dataset. *Proceedings of the Workshop on Challenge on Context-aware Movie Recommendation, CAMRA 2010*, part of the *4th ACM Conference on Recommender Systems, RecSys2010*. Barcelona, Spain, September 30th, 2010.
3. Pedro G. Campos, Silvia Acuña, José A. Macías. Implementación de Propiedades de Usabilidad con Impacto en el Diseño Mediante la Programación Orientada a Aspectos. *Actas del XI Congreso Internacional de Interacción Persona-Ordenador, INTERACCION 2010*, parte del *Congreso Español de Informática 2010, CEDI 2010*. Valencia, España, 7-10 de septiembre de 2010. **Jesús Lóres Best Paper Award**.

We would like to acknowledge computing time support from the Scientific Computing Institute at UAM (CCC at UAM), which was fundamental for the elaboration of some of these publications, and in particular for experimentation during this work.

Chapter 2. Recommender Systems and State-of-the-Art

2.1 Introduction

Nowadays Internet and particularly the World Wide Web bring access to almost non-limited information resources. However, this phenomenon involves some problems, being one of the most important ones the information overload problem. A proposal to deal with this difficulty is to use a Recommender System (RS), aiming to give personalized user guidance about interesting and/or useful items.

This chapter explains some basic concepts about RS, and then presents a review of the state of the art on recommender techniques, analyzing particularly techniques suitable for including temporal information on recommendation. A revision of evaluation metrics used to compare recommender results is also included.

2.2 Recommender Systems

RS apply varying techniques from statistics and knowledge discovery to the problem of recommending items to users of a system [Sarwar et al., 2000]. Most known RS are used in e-commerce; within this context, RS suggest products to customers. Products can be recommended based on best sellers on a site, based on the demographics of the customers, or based on an analysis of the past buying behavior of the customer as a prediction for future buying behavior. Broadly, these techniques are part of personalization on a site, because they help the site to adapt itself to each customer. RS automate personalization of the Web, enabling individual personalization for each customer [Schafer et al., 1999].

As noted by [Adomavicius and Tuzhilin, 2005], although the roots of RS can be traced back to the work on cognitive science, approximation theory, information retrieval or forecasting theories, RS emerged as an independent research area in the mid-1990s when researchers started focusing on recommendations problems that explicitly rely on the ratings structure.

One of the first known RS was the Tapestry system [Goldberg et al., 1992], developed at Xerox Parc. This was a filtering system for electronic documents, primarily e-mail and Usenet postings. A user could create filtering rules like "show me documents that are replied to by other members of my research group" or "don't show me any messages which Joe said were a waste-of-time". Non-automated filtering systems such as Tapestry required the user to determine the relevant predictive relationships within the community, placing a large cognitive load on the user [Herlocker, 2000]. Automating the process of recommendation allow recommendations for large communities of users. In this sense, one of the first automated RS was GroupLens [Resnick et al., 1994]which used a neighborhood-based algorithm.

Several issues have been addressed though the last decade and a half. According to [Lathia et al., 2007], traditionally RS face two conflicting challenges: on one hand accuracy (the generation of recommendations that closely match the user's actual taste), and on the other hand, scalability, since generating such recommendations requires a lot of computational power. With a broader vision, [Kumar et al., 2001] argue that most research on RS have been focused on three areas: (i) the design of algorithms that, given the past preferences of the user, will make useful recommendations; (ii) how to gather the information on user preferences as conveniently and unobtrusively as possible; and (iii) privacy issues, i.e., how to combine the information gathered from a group of users to the advantage of an individual user, without divulging information about other users.

Other topics covered to a lesser extent are how to evaluate recommendation results [Herlocker et al., 2004], usability aspects related with RS [Santos and Boticario, 2008] and integration to software development processes [Rojas et al., 2009], to name a few. Multiple and diverse ideas and techniques have been used with the goal of improving RS performance, as for example machine learning techniques such as neural networks [Pazzani and Billsus, 1997] or Bayesian classifiers [Mooney and Bennett, 1998], using contextual information [Adomavicius et al., 2005], genetic algorithms [Kim and Ahn, 2008], ontologies [Mylonas et al., 2008], matrix factorization techniques [Takács et al., 2008] or temporal information [Lee et al., 2009; Koren, 2009a; Lathia et al., 2009a].

Among the different ideas and techniques, it is notable that one of the most popular techniques for developing recommender algorithms is *Collaborative Filtering* [Su and Khoshgoftaar, 2009; Symeonidis et al., 2008b] or a mixture including it (as in a hybrid recommender). Multiple information sources may be used, but we must remark that the most valuable one is explicit rating information, which is the primary input for collaborative filtering. In order to introduce and structure appropriately the different techniques, a classic RS classification scheme is presented.

2.3 Classical Recommender Algorithms Classification

RS are usually classified into the following categories, according to the algorithms used to generate recommendations [Adomavicius and Tuzhilin, 2005], i.e., how the information filtering task is performed:

- *Content-based filtering*: The system will recommend items with similar declared characteristics to the ones the *target* user[7] preferred in the past;
- *Collaborative filtering*: The system will recommend items that people with similar tastes and preferences to the target user liked in the past;
- *Hybrid approaches*: These methods combine collaborative and content-based filtering elements.

As stated in section 1.2, the common task for RS is to recommend to the target user those items with the highest predicted utility, that is, the RS uses its filtering approach to establish

[7] User for whom the recommendation is made, also called the *active* user.

individual items' utilities, and then recommending those with highest utility. However, there exist another variant whose aim is to determine the relative order that the user would gave to items instead of their individual utilities. This variant is sometimes called *Preference Based Filtering* [Jin et al., 2003], although the techniques used for the ordering generation can also fall into the three aforementioned categories.

2.3.1 Content-Based Filtering

Content-based filtering (CBF) algorithms search for items *similar* to other items that the user liked in the past. That is, the predicted utility $\mathcal{F}(u,i)$ of item i for user u is estimated based on the known utilities (ratings) $R_{u,i'} \subset R$ assigned by user u to the set of items $i' \in \mathcal{I}$ that are *similar* to item i. To estimate such *similarity*, the RS uses stored information about the items, e.g. in the case of movies, genre, director, etc. This approach has its roots in the classical Information Retrieval (IR) [Baeza-Yates and Ribeiro-Neto, 1999] and Information Filtering [Belkin and Croft, 1992] research fields. The improvement over traditional IR approaches come from the use of *user profiles* that contain information about users' tastes, preferences, and needs. The profiling information can be elicited from users explicitly, e.g., through questionnaires, or implicitly—learned from their transactional behavior over time [Adomavicius and Tuzhilin, 2005]. Accordingly, the item information stored by the RS is known as *item profile*. It is usually computed by extracting a set of features of the item (possibly from external sources), and is used to determine the appropriateness of the item for recommendation purposes.

Given the significant advances on the IR field, typically item and user profiles are described as sets of *keywords* which allow the subsequent use of traditional techniques of IR to determine matching documents (in this case item profiles) to a predefined query or set of terms (in this case user profile) [Adomavicius and Tuzhilin, 2005]. In this kind of RS, the utility function $\mathcal{F}(u,i)$ can be described as a computation on the user and item profiles. Following the notation of [Adomavicius and Tuzhilin, 2005], we can formalize this computation as follows.

Definition: Let $CB\_user\_profile(u)$ be a content based user profile, consisting in a set of (weighted) keywords describing tastes and preferences of u, and let $item\_profile(i)$ be an item profile, consisting in a set of (weighted) keywords describing characteristics of item i. The utility function $\mathcal{F}(u,i)$ on a content-based RS is calculated as a scoring function on the user and item profiles:

$$\mathcal{F}(u,i) = score\big(CB\_user\_profile(u), item\_profile(i)\big) \quad \textbf{(8)}$$

Using a keyword based representation for profiles and, for example, the well-known keyword weighting scheme *term frequency-inverse document frequency* (TF-IDF) measure [Baeza-Yates and Ribeiro-Neto, 1999], $user\_profile(u)$ and $item\_profile(i)$ can be represented as TF-IDF vectors, and thus, $score(\cdot,\cdot)$ can be calculated with known IR scoring heuristics such as the cosine similarity measure [Baeza-Yates and Ribeiro-Neto, 1999]. Other techniques for content-based recommendations have also been used, such as Bayesian classifi-

ers or machine learning techniques including clustering, decision trees, and artificial neural networks [Adomavicius and Tuzhilin, 2005].

Given its nature strongly dependent on users' activities and stored information, content-based RS have a number of limitations [Adomavicius and Tuzhilin, 2005]:

- Limited content analysis: content-based RS are limited by the features that are explicitly associated with the objects that these systems recommend. This means that the content (item profiles) must either be in a form that can be parsed automatically by the system (e.g., text) or the features should be assigned to items manually. Automatic feature extraction can be particularly difficult for certain item domains (e.g. music), and the assignation of attributes by hand may be unpractical due to limitations of resources. Moreover, different items represented by the same (limited) set of features may be indistinguishable.
- Overspecialization: When the system can only recommend items that score highly against a user's profile, the user is limited to being recommended items that are similar to those already rated. This problem is not only that content-based RS cannot recommend items that are different from anything the user has seen before, but sometimes recommend items too similar to something the user has already used, such as different news articles describing the same event. Thus, this problem is directly related with the *diversity* of recommendations presented to users.
- New user problem: A RS needs enough information in the user profile before it can generate reliable recommendations. Therefore, a new user, which has entered very few information to the system, would not be able to get accurate recommendations.

2.3.2 Collaborative Filtering algorithms

Collaborative Filtering (CF) algorithms try to predict the utility of items for a particular user based on the items previously rated by *other users* (as opposed to CB RS which base their recommendations on items previously rated by the *same user*). This way, a CF RS is not limited to recommend items similar to those that the target user already know, enabling the recommendation of items completely unknown by him/her, taking advantage of information from other users (that is why this kind of filtering is known as *collaborative*). In firsts formulations, the utility $\mathcal{F}(u,i)$ of item i for user u is estimated based on the known utilities (ratings) $R_{u',i}$ assigned to item i by those users $u' \in \mathcal{U}$ that are related to user u. To estimate the existence of a relation between users, the RS compares the profiles of users, which in this case includes (or consists primarily on) ratings given to items by the user, identifying users with similar tastes and preferences to the target user (deduced from similar ratings for instance). Following the notation of [Adomavicius and Tuzhilin, 2005], we can formalize this computation as follows.

Definition: Let $N^{user}(u)$ be a function which returns a set of users related to u (the neighborhood of u). The utility function $\mathcal{F}(u,i)$ on a *user based* collaborative filtering RS is calculated as a computation on the known utilities that other users have informed about the same item:

$$\mathcal{F}(u,i) = \underset{u' \in N^{user}(u)}{\text{aggr}} R_{u',i} \quad \textbf{(9)}$$

Here we emphasize the *user based* characteristic, because the computation relies on (implicit) relations between users. There are many proposals in the RS literature to compute the aggregation function, and particularly $N^{user}(u)$, usually taking into account the degree of similarity among the target user u and the possible related users u'. The most common approach is to select the k most similar users (in terms of rating behavior), leading to the traditional kNN algorithm (see section 3.1 for further details about kNN algorithms). As a way to overcome with huge computational effort in domains where the number of users far exceeds the number of items, [Sarwar et al., 2001] proposed to compute similarities between *items* and thus obtain ratings from items similar to the *target item* i, leading to *top-N* item recommendations [Deshpande and Karypis, 2004](a.k.a. *item based* collaborative RS):

Definition: Let $N^{item}(i)$ be a function which returns a set of items related to i (the neighborhood of i). The utility function $\mathcal{F}(u,i)$ on an *item based* collaborative filtering RS is calculated as a computation on the known utilities that the user has informed about other items:

$$\mathcal{F}(u,i) = \underset{i' \in N^{item}(i)}{\text{aggr}} R_{u,i'} \quad \textbf{(10)}$$

It is important to note here that in the computation of $N^{item}(i)$ ratings given by other users to item i are used, thus maintaining the premise of collaborative filtering of take advantage of ratings from users distinct from the target user. This approach has been shown to provide better computational performance than user-based CF on environments with more users than items.

The previous approaches have been identified as *memory based* (a.k.a. *heuristic based*) CF, due to the fact that in the computation of $N^{user}(u)$ (and similarly in the computation of $N^{item}(i)$) the entire collection of known utility values is used. An additional approach for CF exists, known as *model based* CF, in which the known utility values are used to learn a *model* which is then used to compute $\mathcal{F}(u,i)$ [Breese et al., 1998]. For model based algorithms, different approaches are described in the RS literature, such as probabilistic models [Getoor and Sahami, 1999], neural networks [Pazzani and Billsus, 1997], clustering [Breese et al., 1998], to name a few. A method combining both memory and model based algorithms have been also published [Pennock et al., 2000], showing better results than pure memory based and model based CF algorithms.

As mentioned earlier, collaborative RS are able to recommend items dissimilar to those used in the past by the target user, because the recommendation relies on other users' (items') recommendations. Due to this, it is possible also for this kind of RS to deal with any kind of content (items), because it does not need (textual) descriptions of the items. However, collaborative RS have their own limitations:

- New user problem: Similarly as with content based RS, the system needs enough information about tastes and preferences of the user in order to make accurate recommendations.
- New item problem: If no enough users rate new items added to the system, it would not be able to recommend those items. New user and new item problems are also known as *cold start* and *ramp-up*.
- Sparsity: Usually, the number of ratings stored in a RS is very small compared to the number of ratings that need to be estimated, which difficult a correct estimation of unknown ratings.
- Grey sheep: If there are users whose tastes are unusual compared to the rest of the population, there will not be (enough) other users who are similar.

To overcome in part the two last problems, it has been proposed to use additional information present in the user profile, particularly demographics (i.e. gender, age, location, etc.). This kind of collaborative RS is sometimes called *demographic filtering*. To deal with cold start, a common approach is to use hybrid algorithms

2.3.3 Hybrid algorithms

In RS context, a hybrid recommender is a combination of content based and collaborative filtering algorithms, which helps to avoid some limitations of such algorithms alone. [Adomavicius and Tuzhilin, 2005] classify hybrid RS as follows:

- RS with separate implementations of collaborative and content based methods, combining their predictions. The mixture can be made using a linear combination of ratings or a voting scheme. Alternatively, at a given moment one of the individual recommenders (the "best" according to some metric) can be chosen.
- Collaborative RS incorporating some content based characteristics. For example, using content based information in the user profile to calculate user similarity.
- Content based RS incorporating some collaborative characteristics. For example, using dimensionality reduction techniques (used to reduce dimensionality of sparse rating matrix on CF RS) on content based profiles.
- RS with a general unifying model that incorporates both content based and collaborative characteristics. That is, using content based and collaborative characteristics in a single method which is able to generate recommendations taking advantage of all these characteristics. There have been some proposals of such unifying models in literature, for example [Ansari et al., 2000] in which the profile information of users and items are used in a single statistical model.

[Burke, 2002] presented a more detailed taxonomy of hybrid RS, which include:

- Weighted hybrids: The utility of an item is computed from the result of all available recommendation techniques (equivalent to the "separate implementations combining their predictions" class mentioned before).
- Switching hybrids: The RS uses some criterion to switch between recommendation techniques (equivalent to the "best" chosen individual recommender).

- Mixed hybrids: Recommendations from more than one technique are presented together. Strictly speaking, this is not a hybrid, but a mix of recommendations.
- Feature combination hybrids: In this case collaborative information is treated as additional feature data associated with each example and use content based techniques over this augmented data set. It is a particular case of "Content based RS incorporating collaborative characteristics".
- Cascade hybrids: Recommendation techniques are employed one after another, in a staged process, breaking ties and refining results from the previous technique. The output of one recommender is not the input for the next, but the results of the recommenders involved are combined in a prioritized manner.
- Feature augmentation hybrids: One technique is employed to produce a rating or classification of an item, and that information is then incorporated into the processing of the next recommendation technique.
- Meta-level hybrids: The entire model generated by one recommender algorithm is used as the input for another.

In summary, hybridization is used to alleviate some of the problems associated with content based or collaborative filtering techniques alone (although not all—new user problem will be present in content based, collaborative and hybrid recommenders as well). Anyhow, several papers empirically compare the performance of hybrid vs. pure collaborative and content based methods, showing that hybrid recommenders can provide more accurate recommendations [Balabanovic and Shoham, 1997; Melville et al., 2002; Pazzani, 1999; Soboroff and Nicholas, 1999].

2.3.4 Preference Based Filtering

The most common method to recommend items to users have been to predict the absolute value of ratings (utility) that individual users would give to the yet unused items, using some of the filtering techniques mentioned earlier [Adomavicius and Tuzhilin, 2005]. However, another approach that has been studied is the prediction of the relative preferences of the users, that is, to estimate the correct relative order of preference that a user would give to a set of items.

A simple approach to generate ranked list of items (expecting to reflect an ordering of user preferences) is to use the predicted utility computed with some of the above-mentioned methods, and then rank items sorting them according to that rating (see for example [Deshpande and Karypis, 2004; Breese et al., 1998; Billsus and Pazzani, 1998]). While such approach might work well in practice, it still do not guarantee that the generated ranking match effectively the preference order of users [Freund et al., 2003], i.e., a user might prefer (use) an item i_a with a lower computed utility than another item i_b. This seemly weird behavior may be due to the fact that frequently in CF the prediction of the utility of an item for a user is computed as an aggregation of utilities manifested by many users who often use completely different ranges of utility to express identical preferences [Freund et al., 2003]. Another problem with such approach is that elements that should be ranked may be considered as negative feedback during training (user have not manifested utility for such items) [Rendle et al., 2009b; Pan et al., 2008].

Considering the above, another approach more suitable for this task is to model relative ordering between items instead of the absolute utility, assuming that relative orderings between items are more consistent than the values of utilities within the class of users with similar interests [Jin et al., 2003]. For example, if a user u manifest that $\mathcal{F}(u, i_a) > \mathcal{F}(u, i_b)$, then the only information considered is that i_a is preferred to i_b, leaving aside the utility values. With this purpose [Cohen et al., 1999] propose a two-stage method, learning a *preference function* $Pref(i_a, i_b)$ returning a measure of how certainty is that i_a should be ranked before i_b in the first stage, and using the learned preference function to order a set of items[8]. [Jin et al., 2003] use a probabilistic approach to generate a graphic model only modeling relative orderings, and computing the probability for each user to rate pairs of items with a particular ordering.

The problem of generating a ranked list of preferences instead predicting utility (in the form of ratings) has also been tackled on CF recommending environments that do not collect ratings, depending on implicit information to make recommendations (e.g. bookmarked web pages), which correspond to positive only feedback (commonly there is no way to make a "negative" bookmark); this kind of CF is sometimes called *One Class CF* [Pan et al., 2008]. In such environments it makes no sense to predict a rating, as users do not rate items, thus the recommendation is a prediction of the probability that the user will prefer the item. The problem is how to determine which elements are less preferred by users, in the absence of negative feedback; in order to compute the preference, the missing values may be treated as negative or unknown values, or a combination of them. For example, [Pan et al., 2008] use different strategies to balance and tune the interpretation of missing values as negative ones.

Based on the before-mentioned considerations, Rendle et al.[Rendle et al., 2009b] propose a Bayesian Personalized Ranking criterion for optimization of personalized rankings, and a learning algorithm based on stochastic gradient descent which are applicable to recommender models such kNN or matrix factorization, outperforming equivalent models optimized for error minimization on the area under the ROC curve, which is a metric used to evaluate ranking classifiers [Ling et al., 2003].

2.4 Temporality in Recommender Systems Literature

2.4.1 First approaches

Temporality has attracted little attention in the RS literature untill recently. One of the firsts works mentioning this dimension is the proposal of Zimdars et al. [Zimdars et al., 2001], in which authors treat the CF problem as a time-series prediction task testing two approaches: i) transforming the implicit preference data (web page visits) to encode the ordering of visits through a *data expansion* scheme; and ii) binning the data. This work showed improvements over non-temporal approaches tested, although increased data sparsity[9] arose

[8] The work of [Cohen et al., 1999] is not directly devoted for RS, so they use the concept of instances instead of items, but within the context of the present work these terms may be used interchangeably.

[9] The data expansion scheme, though did not truncate data, indeed incremented considerably the number of parameters to be optimized by the learning algorithm.

as a challenging problem. In 2002 Terveen et al. used personal history of users to determine their present musical preferences, in a visual interactive system called "HistView" [Terveen et al., 2002]. In 2003, Tang et al. used the production year of movies to scale down the candidate set of a CF RS of movies by pruning "old" movies, which lead to a "truncation" of ratings considered on computations and thus reducing dimensionality. Moreover they obtained improved recommendation accuracy [Tang et al., 2003]. Though only one particular temporal aspect was considered, this work shows the importance of temporal dimension. Another related work is from Sugiyama et al. who in 2004 explored a temporal CF approach in which a detailed analysis of the internet navigation history during a day is performed [Sugiyama et al., 2004].

2.4.2 Weighting schemes

In 2005, Ding and Li incorporated a time based weight into a memory based CF, increasingly penalizing old ratings, i.e., decreasing the importance of known ratings as time distance from recommendation time increases [Ding and Li, 2005]. Upon this strategy, in 2006 they developed a *recency* based CF algorithm [Ding et al., 2006]. In 2007, Ma et al. [Ma et al., 2007] extended this strategy incorporating a preprocessing step which consisted in the usage of a CBF algorithm to calculate unknown rating values (alleviating sparsity), thus obtaining a full virtual rating matrix, which is then used with the CF weighting algorithms.

In 2008, Lee et al. proposed a CF algorithm based upon implicit feedback (purchase data) with a somewhat more general temporal model as it includes two temporal aspects: the "launch" time of an item (the moment at which it is incorporated into the catalog of the RS) and the purchase time. The purchase data is converted into "pseudo-rating" data, and obtained ratings are increasingly weighted as launch time of the item or purchase time is more recent, based on the assumptions that i) more recent purchases better reflect a user's current preference, and ii) recently launched items appeal more to users. With a very simple weighting scheme, they obtain substantial increments (up to 47%) in the number of items recommended that are also purchased by users (the RS recommended mobile wallpapers) [Lee et al., 2008]. In 2009, the same research team presents a more detailed analysis with different weighting approaches according to temporality of data [Lee et al., 2009]; these results show that taking into consideration any of the two temporal aspects separately leads to better item sales, and using both temporal aspects further improves sales, though not additively. Although these results were promising, the time weighting scheme seems limited as valuable information becomes undervalued, similarly to the information lost occurring in rating "truncation".

2.4.3 Time influence in kNN algorithms

Another approach for time inclusion in RS was given by Neal Lathia and collaborators [Lathia et al., 2009a; Lathia et al., 2008; Lathia et al., 2009b]. In 2008 they analyzed kNN algorithm's behavior from a temporal perspective, considering the algorithm as a process generating an implicit social network in which nodes represent users and edges represent relationships between users [Lathia et al., 2008]. Some interesting findings in this work are: i) neighborhoods formed by kNN are dynamic, i.e., they change as time passes though finally converge into a stable set; ii) the convergence velocity (of the neighborhood) depends on the similarity measure (which determines the similarity between pairs of users); and iii) it

is possible to identify "power users"—users frequently selected to form other users' neighborhood, and so having stronger influence on predictions made by the algorithm. In 2009 these researchers pose that production RS must continually adjust parameters as new information is incorporated into the *dataset*. However, observing the results of applying kNN on a dataset *iteratively* (simulating data actualizations from time to time), results are not as good as those from applying kNN on a static dataset. From this, in [Lathia et al., 2009a] a method for updating the neighborhood size for each user is proposed, which leads to lower prediction error as showed in their experimentation. Later on, in [Lathia et al., 2009b] an analysis about evaluation in RS field is performed, criticizing an excessive focus in accuracy[10], proposing a new metric for evaluating CF through time called Time Averaged RMSE (Root Mean Squared Error), which is the average RMSE[11] among predictions and ratings until a particular time (instead of calculating it over the full dataset). Applying this metric allows seeing differences in accuracy through time. Moreover, they also propose to evaluate diversity among recommendation lists generated in different moments, using a metric based on Jaccard similarity. Their experimentation showed that methods with better accuracy have also poorer diversity. One step further, in [Lathia et al., 2010] authors argue that temporal diversity is a need, as reproducing similar recommendations through time may generate a negative perception on user (this point is held based upon a user survey). This work also presents a simple metric for evaluating diversity among recommendation lists, and simple (but effective) approaches to increment temporal diversity, which seems to have no impact on accuracy.

2.4.4 Incorporation of time parameters into models

The methods reviewed so far in this section can be considered as time-aware heuristics, in the sense that all of them perform direct computations on the full dataset, varying only the particular computation performed according to the additional time information. However, (as is the case with general CF data) it is also possible to learn models from the dataset, which are then used to make recommendations[12]. One way to make such models time-aware is to incorporate into them additional parameters reflecting dynamic changes in data. The incorporation of parameters reflecting systematic effects associated to users or items, known as *biases* [Bell and Koren, 2007; Bell et al., 2008; Koren, 2008] in the context of the *Netflix Prize* competition[13] motivated this line of research. For example, typical CF data exhibits large systematic tendencies for some users to give higher ratings than others, and for some items to receive higher ratings than others. Incorporating parameters which isolate such tendencies help researches to improve model's results [Koren et al., 2009]. Similarly, additional parameters reflecting dynamic tendencies such as users' taste change or items' popularity drift allows models to obtain better results. For instance Xiang and Yang [Xiang and Yang, 2009] considers four main types of *temporal effects*, incorporating the corresponding parameters into Matrix Factorization (MF) models: i) *time bias*, reflecting changes in society's interests and habits (e.g. fashion changes); ii) *user bias shifting*, reflecting individual change in user habits (e.g. a user may become more demanding as he/she ac-

[10] In fact other authors have also criticized the only-accuracy evaluation approach, e.g. Adomavicius and Tuzhilin [Adomavicius and Tuzhilin, 2005] and Herlocker et al. [Herlocker et al., 2004].
[11] See section 2.5.2.2 for details.
[12] These algorithms were introduced as model based previously in this chapter.
[13] www.netflixprize.com

quires more experience with items, lowering ratings to all items); iii) *item bias shifting*, reflecting individual change in item popularity (e.g. a movie may become more popular for a short period of time after it wins an Oscar award); and iv) *user preference shifting*, reflecting individual change in user preferences (e.g., a user may like cartoons when he/she is young, but when he grows up may dislike them). Other time effects are also mentioned, as the *year/month effect*, which reflect particular tendencies associated with the moment of recommendation (e.g. periodicity on a year/month basis) and the *loyalty, activity and popularity effect* which reflects the amount (duration) of activity of users/items in the RS. Their experimentation show better RMSE results on Netflix data as more time effects are incorporated into models. This work also note that time decay (i.e. time weighting) is not optimum due to lost (or undervaluation) of old though important data.

As mentioned earlier, the Netflix Prize competition encouraged an enormous amount of research and improvements in the CF field. Some major findings listed by the competition's winning team led by Yehuda Koren are the usage of MF models, the ensemble of different methods including variants of neighborhood based CF and MF based CF, and the incorporation of time information [Koren et al., 2009; Koren, 2009a; Bell and Koren, 2007; Bell et al., 2008; Koren, 2008; Koren, 2009b; Bell et al., 2009]. In 2007, Bell and Koren [Bell and Koren, 2007] considers temporal effects as global effects, considering that interactions between time and users (or items) vary linearly with the square root of the number of days since the first rating from the user (or to the item) untill the date of the actual rating being predicted. In 2008 they propose the use of parameters associated with time intervals, grouping ratings according to the rating date, aiming to better model temporal dynamics in user and item tendencies. Given that tendencies change more slowly on items, with few parameters of this kind it is possible to adequately learn and model item temporal tendencies; however the modeling of users becomes more complex as they may have many kinds of dynamics, e.g. sudden (during one day due to bad humor) and long term (slow change of interests as the user becomes older), thus requiring more parameters and training data to appropriate learning [Bell et al., 2008]. With a detailed explanation, Koren [Koren, 2009a] fully describes their proposal in 2009, which includes many temporal effects that may be included in MF models or neighborhood based ones, and arguing the use of temporal information as a key to win the Netflix Prize competition. He also notes clear temporal effects on Netflix data, emphasizing that the inclusion of temporal dynamics proved more useful than various algorithmic enhancements in order to improve accuracy of rating predictions.

2.4.5 Evaluation of interest drift

Another line of research in this context has been the explicit detection of drift in user interests. In 2005 Min and Han [Min and Han, 2005] propose two simple methods in the context of a user kNN recommendation algorithm to detect changes in user interests: i) a clustering based method which detects change of user's cluster in different timeframes (each cluster correspond to a set of users with similar interests), and thus a change of cluster is interpreted as interest drift; and ii) a similarity based method (named auto similarity), which compares ratings of the active user by the classic Pearson correlation coefficient in different timeframes, thus interpreting a negative correlation as a change in user interests. If

an interest change is detected, then the neighborhood formation is affected by weighting differentiation on the computation of similarity among users, according to the timeframe at which the interest change was detected. To deal with data sparsity, a dimensionality reduction technique based on item hierarchy is used, by which the ratings are treated at item category level.

In 2009 Cao et al. [Cao et al., 2009] presents a graph based method to identify interest drift. They identify four types of user interest patterns, based in the category of items rated by the user: i) *Single Interest Pattern* (SIP), which correspond to users with one interest (rates only one category of items) during a time span; ii) *Multiple Interest Pattern* (MIP), which reflects users with more than one interest during a time span; iii) *Interest Drift Pattern* (IDP), which reflects users with identifiable interests but not lasting all the time span; and iv) *Casual Noise Pattern* (CNP), which reflects users with very short duration interests. In order to establish the pattern of a user, a *rating graph* and *rating chain* are built. The rating graph has nodes representing items rated by a user, and there is an edge between items if their similarity (according to some predefined measure) is bigger than a certain threshold. The density of this graph allows to observe how similar are the items rated by the user. On the other hand, the rating chain also has nodes representing items rated by a user (identifying them according to the rating time), and there is a link between a pair of nodes (i.e. items rated consecutively) only if those two items are similar. With a simple analysis on those structures it is possible to determine simple interest and casual noise patterns. Users whose rating graph have high density and whose rating chain have high continuity have a single interest patterns, meanwhile low density rating graphs and low continuity rating chains reflect a casual noise pattern. Determining multiple interest and interest drift patterns require a more careful analysis, being need to identify whether there is any drift point. With this goal, an algorithm called *Density Based Segmentation* is proposed, which is able to identify such drift points based on the rating graph and rating chain characteristics. A multiple interest pattern has no drift point, whilst an interest drift pattern may have one or more drift points. With this information, the authors propose a general framework to improve a RS, which in summary consist in identifying the corresponding pattern of a user's rating series, and depending upon the type of pattern detected, apply one of the following: a) make recommendation using the full rating series (for SIP and MIP); b) recommend the most popular items discarding the user's rating series (for CNP); and c) use only ratings after the last interest drift of the user (for IDP). Results on Movielens dataset and synthetic data show improvements in recommendation performance. The authors argue that a wide range of RS can be enhanced with this approach, as it can be considered a *wrapper* which can be added as an interest pattern detection module into an existing RS.

2.4.6 Periodicity in tastes

Another view of the time influence on the recommendation problem is given by Baltrunas and Amatriain in 2009 [Baltrunas and Amatriain, 2009], who assumes that user preferences change over time but have temporal repetition in the music domain (e.g. a user listens to one type of music while working and another type before going to sleep). They tested some time segmentation of the rating data (which corresponded to transformed implicit feedback as in [Celma, 2008]) in order to create *micro-profiles* of users containing ratings of such time

segments (e.g. morning/evening, working day/weekend, cold season/hot season). Using data from the corresponding micro-profile they improved accuracy (MAE) on about 2% compared to the usage of the full rating data with an ad-hoc dataset extracted from last.fm[14] RS using a factorization CF algorithm. One faced problem was that of how to define a time segment, e.g. *morning*, as it may differ for different users; they try with different partitioning schemes, and interestingly tested cross-validation, information gain and explained variance schemes to estimate the best split.

2.4.7 Inclusion of Temporal Dimension as Contextual Information

A general framework for extending the traditional scheme of recommendation based only on (user, item, rating) pieces of information have been developed by Gediminas Adomavicious and collaborators [Adomavicius et al., 2005; Adomavicius and Tuzhilin, 2011; Adomavicius and Tuzhilin, 2001a; Adomavicius and Tuzhilin, 2001b; Adomavicius and Tuzhilin, 2008]. With this purpose, in [Adomavicius and Tuzhilin, 2001a] propose to incorporate multiple dimensions of information (besides the classical user and item dimensions), one of which is time, store it in a *recommendation warehouse* and use OLAP[15]-like aggregation capabilities in order to be able to aggregate information from this more complex recommendation space. Building on this idea, in [Adomavicius and Tuzhilin, 2001b] the authors extend their previous proposal incorporating a *Recommendation Query Language* with the purpose of allowing to express a wide range of specific recommendations "on the fly". Later on, in [Adomavicius et al., 2005] they gave a further description of this model, and the additional information that the model allows to incorporate is defined as contextual information (some details of this model are depicted in section 1.2). Figure 1 shows a graphical representation of the proposed model. This paper also describes reduction-based, heuristic-based and model-based rating estimation approaches based on multidimensional information. The former is deeper discussed, suggesting to do a pre-processing of data and selecting from the multidimensional space the information that matches the contextual values (i.e. for a recommendation during a weekday select only ratings related with users and items that were done during weekdays), thus reducing the recommendation space to the classical $\mathcal{U} \times \mathcal{I}$ domain, and enabling the usage of existing RS.

[14] www.last.fm

[15] On-Line Analytical Processing: An approach for fast answering of multi-dimensional database queries.

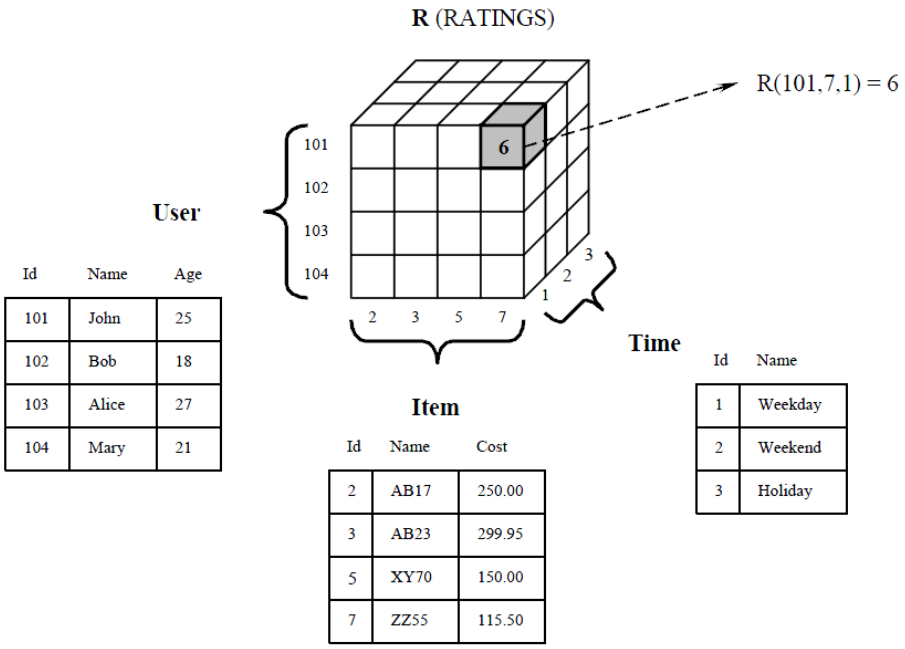


Figure 1. Multidimensional model for a recommendation space (from [Adomavicius et al., 2005]).

This scheme, as noted by the authors, may help improving recommendation accuracy in *some* cases, due to incremented data sparcity. Using as contextual dimensions *time*, *place* and *companion*, they built a movie RS for evaluation purposes, obtaining better results with a hybrid of reduce-based and classic CF over classic CF alone. Afterwards in a tutorial during the 2nd ACM Conference on Recommender Systems, Adomavicius and Tuzhilin [Adomavicius and Tuzhilin, 2008] propose a more general scheme for the context-aware recommendation process, which was later on expanded in a book chapter [Adomavicius and Tuzhilin, 2011], identifying three stages for context exploitation: i) contextual pre-filtering, which is equivalent to the reduce-based approach previously discussed; ii) contextual post-filtering, which consists on predict ratings in traditional fashion (with the full rating data available) and then using contextual information to adjust the results; and iii) contextual modeling, which uses contextual information directly in the modeling technique. Figure 2 shows a schematic view of the proposed approaches.

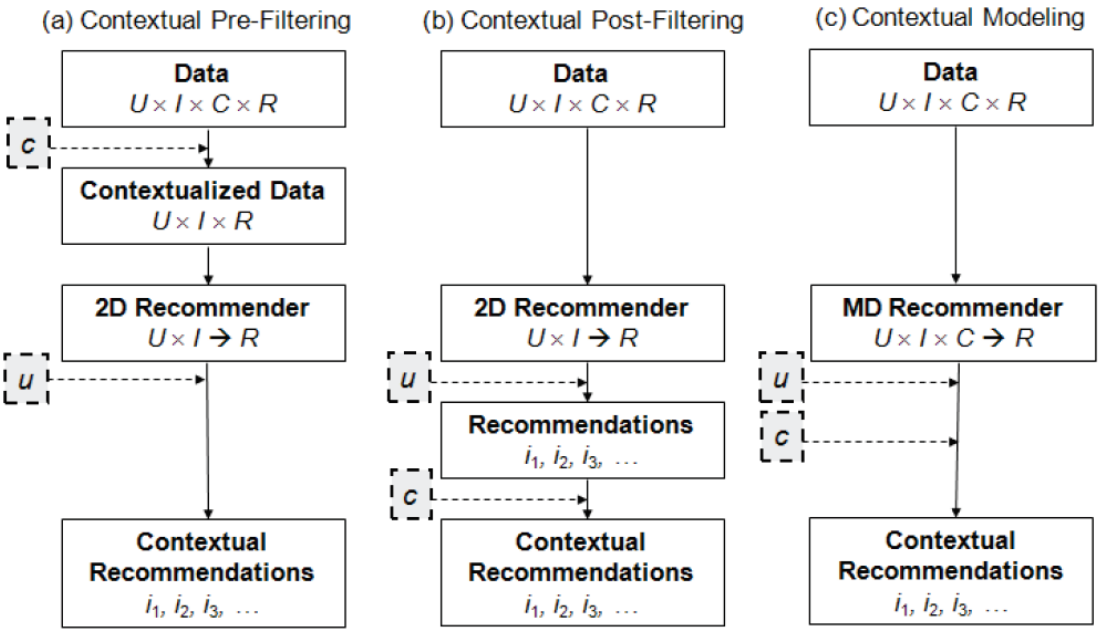


Figure 2. Paradigms for incorporating context in RS (from [Adomavicius and Tuzhilin, 2011]).

In [Panniello et al., 2009] a comparison among pre and post-filtering approaches is presented, using time information as part of contextual information, establishing that neither of the approaches outperforms the other consistently among different datasets. Also they argue that selecting a good post-filtering strategy is time-consuming, proposing a simple heuristic for approach selection. Authors claim improvements up to 30% on F-measure over classic recommendation in their experimentation.

2.4.8 Other proposals

In 2005, Zhang et al. [Zhang et al., 2005] propose an extension to the pLSA model for CF [Hofmann, 1999; Hofmann, 2001; Hofmann, 2004] which allows an incremental learning of the model, taking into consideration the variation of ratings through time and thus allowing the algorithm to track factor drift as argued by the authors. Their experiments show improvements of ~10% over original pLSA in MAE.

In 2007, Ricci and Nguyen [Ricci and Quang Nhat Nguyen, 2007] incorporate long-term and short-term preferences in a critique-based mobile RS. The long-term preferences corresponded to mined past interactions and users' explicitly defined stable preferences, whilst short-term preferences corresponded to session-specific preferences expressed by the user.

In 2009, Lu et al. [Lu et al., 2009] propose a spatio-temporal model for CF based on MF, which includes a Markov random field to incorporate spatial correlations across users and/or items, and a state space framework to model temporal structure estimated by a linear Kalman Filter. They obtained an improvement of 1.4% over standard MF in RMSE in their experimentation.

More recently, motivated by the Challenge on Context-Aware Movie Recommendation (CAMRa 2010[16]) performed jointly with the 4th ACM Conference on Recommender Systems (RecSys 2010[17]), which included a temporal track (w*eekly track*), Gantner et al. [Gantner et al., 2010] propose to use Pairwise Interaction Tensor Factorization [Rendle and Schmidt-Thie Lars, 2010], a tensor factorization model [Lathauwer et al., 2000] initially developed for tag prediction [Symeonidis et al., 2008a; Rendle et al., 2009a] which incorporates an optimization criterion based on preference filtering instead of prediction error minimization[18]. This work won the weekly track of CAMRa 2010. Another proposal presented in this challenge was from Brenner et al. [Brenner et al., 2010], which used a popularity measure (based upon high rated movies) in order to forecast popularity of movies in a short-term period. Interesting results were found on new users (cold start problem) recommended with movies rated during the first two weeks of previously signed-in users.

As well during 2010, Xiang et al. [Xiang et al., 2010] propose the construction of *Session-based Temporal Graphs* (STG), which simultaneously models users' long-term and short-term preferences over time. The authors argue that time dimension is a local effect and should not be compared across all users arbitrarily. They exemplify indicating that users bought different items at the same time triggered by completely different external events, and so acknowledging a relation among those items or users on time is typically not useful because the occurrence is purely coincident, but in the other hand, it is more likely due to the same external event that a user bought multiple items in a short period; that is, user-specific time dimension is more likely to capture the "true" correlation over time [Xiang et al., 2010]. Thus, STG allows modeling such local effects. STG is a bi-partite graph composed of user, item and user-session nodes, allowing links between user and item or user-session and item nodes reflecting items viewed by a user at any time (long-term preferences) and items viewed by a user during a specific time bin or session (short-term preferences) respectively. Moreover, they propose an *Injected Preference Fusion* algorithm, which allows injecting user preferences into nodes of STG and propagate them to item nodes in a weighted arrangement depending upon the type of relation (long or short-term). On this graph authors perform a Temporal Personalized Random Walk, based on personalized Page Rank [Haveliwala, 2002; Scarselli et al., 2004], obtaining improvements on Hit Ratio metric over Delicious and Citeulike datasets[19].

Finally, it is important to mention the existence of a recent US Patent related to temporal recommendation [Zhao and Fukushima, 2010]. This invention proposes the use of temporal rating models to predict how the user would rate an item at different times, and thus estimate the optimal time to recommend such item to the user. The authors exemplify some strategies, as for example determining which hours of the day (or which days of the week) users rate a restaurant highly (e.g. before dinner), and so recommend the restaurant at peak rating time (or some time before the peak).

[16] http://www.dai-labor.de/camra2010/
[17] http://recsys.acm.org/2010/
[18] See section 2.3.4 for details.
[19] Public datasets available through e-mail contact to CiteULike (www.citeulike.org) and DAI-Labor (www.dai-labor.de) respectively

2.5 Metrics for recommendation evaluation

RS have a variety of properties that may affect its results (and therefore user experience) such as accuracy, robustness, scalability and so forth [Shani and Gunawardana, 2011]. Given the scope of the present work, this section is mainly devoted to metrics that allow comparison of RS algorithms in an offline setting, i.e. without user interaction, using for this purpose rating data in existing datasets.

The most widely studied evaluation dimension for RS in literature has been *accuracy* [Herlocker et al., 2004]. Accuracy metrics can be classified as either *statistical* or *decision-support* [Adomavicius and Tuzhilin, 2005; Herlocker et al., 1999]. Statistical accuracy metrics compare estimated utilities by the RS against the actual utilities collected from the user (typically ratings). Thus, they are commonly used to evaluate rating prediction task' results. Examples of such metrics are Mean Absolute Error, Root Mean Squared Error, and correlation between predictions and ratings [Adomavicius and Tuzhilin, 2005]. On the other hand, decision-support metrics determine how well a RS can predict (recommend) high-relevance items[20]. These metrics are more related to *top-N* recommendation task. Examples of such metrics are classical IR metrics of precision (percentage of truly relevant items among those recommended by the system), recall (percentage of correctly recommended items among all the ratings known to be relevant), F-measure (a harmonic mean of precision and recall) and Receiver Operating Characteristic (ROC) metric demonstrating the trade-off between *true positive* and *false positive* recommendations in RS [Adomavicius and Tuzhilin, 2005; Herlocker et al., 1999]. Other studied evaluation measure has been *coverage*. Coverage measures the percentage of items for which a RS is capable of estimate utility [Adomavicius and Tuzhilin, 2005; Herlocker et al., 1999].

As pointed by [Adomavicius and Tuzhilin, 2005], these metrics do not adequately capture other dimensions such as *usefulness*, *quality*, *novelty* or *diversity* of recommendations, which indeed constitute important dimensions for users (see for example [Lathia et al., 2010; Yang and Padmanabhan, 2001]). Thus novel metrics aiming to evaluate these dimensions are being proposed and used to evaluate RS. In this section some of them are reviewed. We will consider that users express the utility in form of ratings, and that the RS computed utility is a predicted rating. Prior to the description of the metrics, the experimental setting needed to apply such metrics is briefly introduced.

2.5.1 Experimental Setting for Metric Application and Notation

The experimental setting in the present work considers the usage of existing datasets that allow low-cost comparison of algorithms in an offline manner [Shani and Gunawardana, 2011]. Thus, it is necessary to divide such datasets into a *train* and a *test* set, the former used in the *training phase* where the algorithms *learn* about users' tastes and items' being preferred, and the latter used during the *testing phase*, where the metrics are applied, simulating the online process where the system makes recommendations and the users uses (or not) those recommendations. Let $TestSet = \{(u, i, t, r_{u,i,t}) : u \in \mathcal{U}, i \in \mathcal{I}, t \in \mathcal{T}, r_{u,i,t} \in R\}$ be the test

[20] A common interpretation of relevant items in RS field has been "items that would be rated highly by user" [Adomavicius and Tuzhilin, 2005]. However, the concept of "high rating" is not equivalent for all users, as noted by [Freund et al., 2003].

set, which correspond to a 4-tuple consisting of user, item, time and the corresponding rating. It is important to note that a rating pertaining to the test set is not present in the train set (ussually, a part of the known ratings in a dataset is hidden, forming the test set, and the unhidden ratings form the train set). It should also be noted that most metrics are not time-aware, thus the time component is not considered. In such cases the time component is ommited. Additionaly, let $\mathcal{U}_{TestSet} = \{u: u \in TestSet\}$.

2.5.2 Statistical accuracy metrics

These metrics measure the difference between predicted and real utility values. Accordingly, these metrics are better suited in cases where it is important to determine the level of satisfaction that a particular item will cause to a particular user, e.g. in systems which the predicted rating will be displayed to the user [Herlocker et al., 2004].

2.5.2.1 Mean Absolute Error

Mean Absolute Error (MAE) measures the average absolute deviation between a predicted rating and the user's true rating:

$$MAE = \frac{\sum_{r_{u,i} \in TestSet} \left| \hat{r}_{u,i} - r_{u,i} \right|}{|TestSet|} \quad \textbf{(11)}$$

where $\hat{r}_{u,i}$ is the predicted rating value (computed as $\mathcal{F}(u,i)$). In this case, lower values of MAE indicate better accuracy.

2.5.2.2 Root Mean Squared Error

Root Mean Squared Error (RMSE) puts more emphasis on large errors between predicted and true ratings compared with MAE:

$$RMSE = \sqrt{\frac{\sum_{r_{u,i} \in TestSet} \left(\hat{r}_{u,i} - r_{u,i} \right)^2}{|TestSet|}} \quad \textbf{(12)}$$

As the case of MAE, lower values of $RMSE$ indicate better accuracy.

2.5.2.3 Correlation

Correlation is a statistical measure of agreement between two vectors of data. Thus, the correlation between predicted and real ratings can be used to evaluate the accuracy of a RS, in terms of agreement between those values. A correlation coefficient such as Pearson product-moment correlation coefficient can be used with this purpose [Sarwar et al., 1998]. Normally, higher correlation values imply better accuracy.

2.5.2.4 Other Statistical Accuracy Metrics

Other statistical accuracy metrics, or different ways to apply them, can be found in RS literature. For example, *normalized mean absolute error* [Goldberg et al., 2001] is mean absolute error normalized with respect to the range of rating values, theoretically allowing comparison between prediction runs on different datasets. On the other hand, [Shardanand and Maes, 1995] measured separately mean absolute error over items to which users gave ex-

treme ratings. They partitioned their items into two groups, based on user rating (a scale of 1 to 7). Items rated below three or greater than five were considered extremes. The intuition was that users would be much more aware of a recommender system's performance on items that they felt strongly about. The mean absolute error of the extremes did provide a different ranking of algorithms than the common mean absolute error.

It is important to note that these measures are typically performed on test data that the users choose to rate, so test data are likely to constitute a skewed sample (users tend to rate only items they find useful) [Adomavicius and Tuzhilin, 2005].

2.5.3 Decision-support accuracy metrics

These metrics takes into consideration how many relevant items are recommended by the RS. Usually, the ranking of recommended items is also considered in the metric values. Thus, these metrics are better suited in cases where it is important to test if lists of recommended items contain items that are valuable for the user [Herlocker et al., 2004].

2.5.3.1 Precision

Precision of an IR system is defined as the ratio of *documents* retrieved by the system that are relevant (for a user petition or *query*) [Baeza-Yates and Ribeiro-Neto, 1999]. In the context of RS, items take the place of documents. Moreover, there is no explicit query from the user, so user tastes and preferences (in the user profile) take the place of the query. Regarding relevance, a common interpretation of relevant items in RS field has been "items that would be rated highly by user" [Adomavicius and Tuzhilin, 2005]. Thus, RS Precision is the ratio of recommended items that are relevant. Being $\mathcal{I}^*(u)$ the ordered set of all recommended items to user u, let $Rels_u$ be the true set of items considered relevant by user u (also called *ground truth*), and $RelsRecomm_u$ the set of items recommended that are relevant, that is $RelsRecomm_u = \mathcal{I}^*(u) \cap Rels_u$, then Precision of recommendation for user u is:

$$P_u = \frac{|RelsRecomm_u|}{|\mathcal{I}^*(u)|} \quad \textbf{(13)}$$

The Precision of the RS among all (tested) users may be computed by:

$$P = \frac{\sum_{u \in \mathcal{U}_{TestSet}} P_u}{|\mathcal{U}_{TestSet}|} \quad \textbf{(14)}$$

If only the *top* N recommended items are taken into consideration, this ratio is called Precision at N or $P@N$. Let $RelsRecomm_u@N = \mathcal{I}_N^*(u) \cap Rels_u$, then:

$$P_u@N = \frac{|RelsRecomm_u@N|}{|\mathcal{I}_N^*(u)|} \quad \textbf{(15)}$$

Equivalently, $P@N$ of the RS among all (tested) users may be computed by:

$$P@N = \frac{\sum_{u \in \mathcal{U}_{TestSet}} P_u@N}{|\mathcal{U}_{TestSet}|} \quad \textbf{(16)}$$

2.5.3.2 Recall

Recall of an IR system is defined as the ratio of relevant documents that are retrieved. As in the case of Precision, here items replace documents, and the retrieved set of documents correspond to the recommended items by the RS. Thus RS Recall is the ratio of relevant items recommended. Following the notation defined previously, the Recall of recommendation for user u is:

$$Recall_u = \frac{|RelsRecomm_u|}{|Rels_u|} \quad \textbf{(17)}$$

Similarly, Recall at *N*, Recall of the RS among all (tested) users, and Recall at N among all (tested) users may be computed by:

$$Recall_u@N = \frac{|RelsRecomm_u@N|}{|Rels_u|} \quad \textbf{(18)}$$

$$Recall = \frac{\sum_{u \in \mathcal{U}_{TestSet}} Recall_u}{|\mathcal{U}_{TestSet}|} \quad \textbf{(19)}$$

$$Recall@N = \frac{\sum_{u \in \mathcal{U}_{TestSet}} Recall_u@N}{|\mathcal{U}_{TestSet}|} \quad \textbf{(20)}$$

2.5.3.3 F-measure

Several approaches have been taken to combine Precision and Recall into a single metric. One approach is the F1 metric, which combines Precision and Recall into a single number, as a harmonic mean:

$$F1_u = \frac{2P_u Recall_u}{P_u + Recall_u} \quad \textbf{(21)}$$

2.5.3.4 Receiver Operating Characteristic

The Receiver Operating Characteristic (ROC) model [Swets, 1969] attempts to measure the extent to which an information filtering system can successfully distinguish between signal (relevance) and noise [Herlocker et al., 2004]. The model requires that the system predicts the level of relevance of every potential item. It measures the *true positive rate* (TPR; percentage of relevant items predicted as relevant, that is, $Recall$ in the IR context) versus the *false positive rate* (FPR; percentage of irrelevant items predicted as relevant). Considering that typically items are presented to the user in a ranked list, the ROC curve plots this proportion at different cutoffs levels (search lengths). Figure 3 shows an example of an ROC curve. As larger TPR values and lower FPR values means that the system is able to effec-

tively predict relevant items as relevant, a better predictive system will have a curve most near to the upper left corner of the plot. A random predictor will have a straight line from the origin to the upper right corner.

In the computation of ROC curve, items not rated (i.e. whose relevance is not known) are discarded and do not affect the curve negatively or positively [Herlocker et al., 2004]. However [Schein et al., 2002] states that a perfect recommender may not produce a perfect ROC graph, because some recommender systems may display more recommendations than there exist "relevant" items to the recommender, and that these additional recommendations should be counted as false-positives.

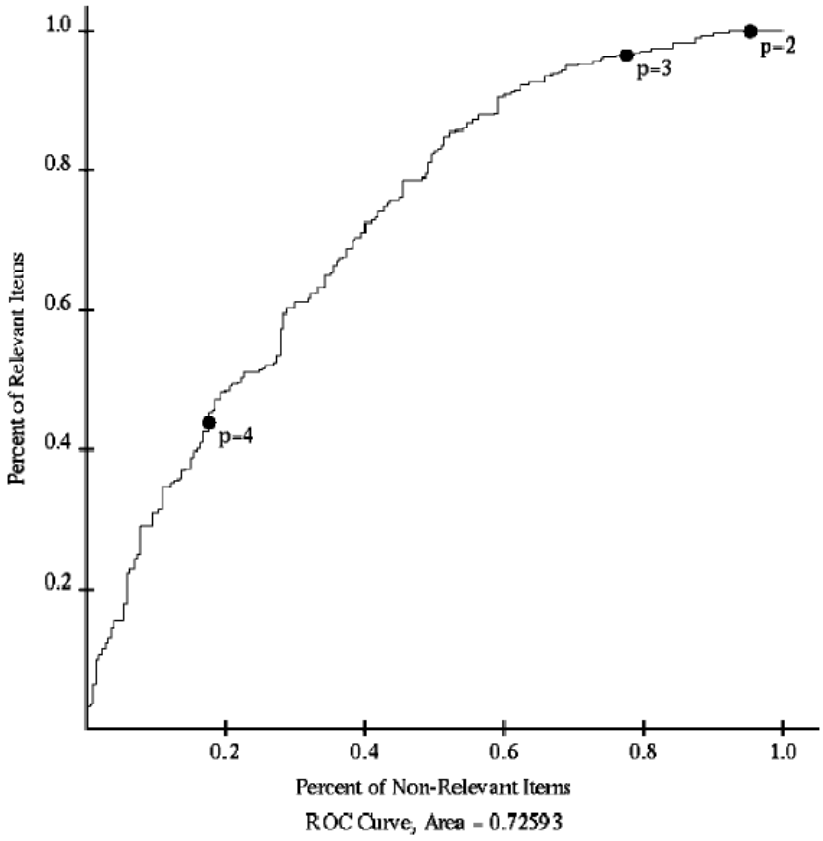


Figure 3. Example of a ROC curve (taken from [Herlocker et al., 2004])

2.5.3.5 Area under the ROC curve

The area underneath a ROC curve can be used as a single metric of the system's ability to discriminate between good and bad items, independent of the search length. According to [Hanley and McNeil, 1982], the area underneath the ROC curve is equivalent to the probability that the system will be able to choose correctly between two items, one randomly selected from the set of bad items, and one randomly selected from the set of good items. Intuitively, the area underneath the ROC curve captures the recall of the system at many different levels of fallout. One form to compute this metric is to build the ROC curve and compute the area. Alternatively it can be computed directly from a ranked list by [Ling et al., 2003]:

$$AUC_u = \frac{S_0 - positive(positive+1)/2}{positive \times negative} \quad \textbf{(22)}$$

where $S_0 = \sum_{i \in Rels_u} rank(i)$, $rank(i)$ is the rank position of item i, $positive$ is the number of positive (relevant) items for u (i.e. $|Rels_u|$) and $negative$ is the number of negative (non relevant) items for u (i.e. $|\mathcal{I} - Rels_u|$).

It also can be computed considering directly the relative ordering assigned by the RS [Rendle et al., 2009b]:

$$AUC_u = \frac{1}{|Rels_u||\mathcal{I} - Rels_u|} \sum_{i \in Rels_u} \sum_{j \in \mathcal{I} - Rels_u} (x_{u,i,j}) \quad \textbf{(23)}$$

where $x_{u,i,j} = 1$ if item i is ranked before (in top of) item j in $\mathcal{I}^*(u)$, and $x_{u,i,j} = 0$ in other case.

2.5.3.6 Discounted Cumulative Gain

Precision and Recall do not take into account the usefulness of an item based on its position in a result list. The premise of Discounted Cumulative Gain (DCG) metric is that highly relevant documents appearing lower in a search result list should be penalized, so the relevance value is reduced logarithmically proportional to the position of the result. The DCG accumulated at a particular rank position N is [Jarvelin and Kekalainen, 2002]:

$$DCG_u@N = rel_{u,1} + \sum_{pos=2}^{N} \frac{rel_{u,pos}}{log_2 pos} \quad \textbf{(24)}$$

where $rel_{u,pos}$ is the relevance value for user u of the item at position pos[21]. In order to allow fair comparison among different users, the DCG should be normalized. This is done by sorting recommended items by known relevance, producing an *ideal* DCG at position N, thus obtaining the normalized DCG:

$$nDCG_u@N = \frac{DCG_u@N}{IDCG_u@N} \quad \textbf{(25)}$$

The nDCG of the RS among all (tested) users may be computed by:

$$nDCG@N = \frac{\sum_{u \in \mathcal{U}_{TestSet}} nDCG_u@N}{|\mathcal{U}_{TestSet}|} \quad \textbf{(26)}$$

[21] There are actually many variants for this computation. For instance, the *trec_eval* utility used for evaluation in TREC (http://trec.nist.gov/) compute it as $DCG_u@N = \sum_{pos=1}^{N} \frac{rel_{u,pos}}{log_2(pos+1)}$.

2.5.3.7 Considerations on Decision-support Accuracy Metrics

The major problem with decision-support accuracy metrics is that the true set of relevant items $Rels_u$ (according to the heuristic of "high rated items") is unknown, but only a subset of them, as users only rate a small subset of the full item set. Indeed, the idea of a RS is to recommend items unknown to the users (otherwise why would the user need a RS). Thus, it is possible to have items actually relevant to a user and recommended by the RS being not considered by the metrics.

A second, apparently minor problem is that of determining which items are indeed relevant to the users. If items are rated binary (e.g. "dislike" and "like" or 0 and 1), it is easy to determine which items are found relevant by the users (and predicted as relevant by the system). The former may lead to think that in the case of multi-valued ratings (e.g. 1-10 scale) higher ratings correspond directly to higher relevance. However, as stated by [Freund et al., 2003], given that a numeric rating can have a different meaning for different users (some users tend to give higher ratings for instance), and that in CF predictions are usually build upon ratings from many different users, it is not clear that between two items i_1, i_2, $\hat{r}_{u,i_1} > \hat{r}_{u,i_2}$ implies that i_1 is more relevant than i_2 for user u. The latter may be particularly important in computing metrics which depend on the top N ranked list of recommended items, such as $P@N$ or $R@N$.

2.5.4 Coverage Metrics

Coverage in RS measures what proportion of items the system is able to compute predictions to.

2.5.4.1 Prediction Coverage

This metric measures directly the percentage of items for which the recommender can form predictions [Herlocker et al., 2004]. Consider again the set $\mathcal{I}^*(u)$, the set of item for which the system is able to compute $\mathcal{F}(u,i)$. The coverage for a user u can be computed as:

$$Cov_u^{pred} = \frac{|\mathcal{I}^*(u)|}{|\mathcal{I}|} \tag{27}$$

It is important to note that different recommendation algorithms may be unable to generate a prediction for different users on the same item. To calculate the prediction coverage among all users, we can compute the average by:

$$Cov^{pred} = \frac{\sum_{u \in \mathcal{U}} Cov_u^{pred}}{|\mathcal{U}|} \tag{28}$$

2.5.4.2 Catalog Coverage

This metric measures what percentage of available items does the RS ever recommend to users. This form of coverage may be more important for e-commerce sites [Herlocker et al., 2004]. Following prior notation in this chapter, let $\mathcal{I}^* = \{i : i \in \bigcup_{u \in \mathcal{U}} \mathcal{I}^*(u)\}$, i.e., $\mathcal{I}^*$ is the

set of items among all recommendation list generated by the RS. Then the general catalog coverage of a RS can be computed as:

$$Cov^{catalog} = \frac{|\mathcal{I}^*|}{|\mathcal{I}|} \quad \textbf{(29)}$$

Catalog coverage is usually measured on a set of recommendations formed at a single point in time. For instance, it might be measured by taking the union of the top 10 recommendations for each user in the population [Herlocker et al., 2004].

2.5.4.3 Interest and Relevance Coverage

An alternative way of computing coverage considers only coverage over items in which a user may have some interest. Coverage of this type is not usually measured over all items, but only over those items a user is known to have examined. For instance, when the predictive accuracy is computed by hiding a selection of ratings and having the RS computed predictions for those ratings (e.g. ratings in a test set), the coverage can be measured as the percentage of covered items for which a prediction can be formed [Herlocker et al., 2004]. Let $\mathcal{I}_{u,TestSet} = \{i: r_{u,i} \in TestSet\}$ be the set of items in which u has shown interest (i.e. have rated such items), and let $\mathcal{I}^*_{u,TestSet} = \{i: i \in \mathcal{I}_{u,TestSet} \wedge \mathcal{F}(u,i) \neq \emptyset\}$ be the set of items of interest for user u for which the RS is able to calculate predictions of utility. Then the interest coverage for a user u can be computed as:

$$Cov_u^{interest} = \frac{|\mathcal{I}^*_{u,TestSet}|}{|\mathcal{I}_{u,TestSet}|} \quad \textbf{(30)}$$

The interest coverage among all users can be computed as:

$$Cov^{interest} = \frac{\sum_{u \in \mathcal{U}_{TestSet}} Cov_u^{interest}}{|\mathcal{U}_{TestSet}|} \quad \textbf{(31)}$$

Similarly, [Bellogín et al., 2010] propose a relevance coverage, which is coverage over relevant items. It can be computed as:

$$cov^{rel} = \frac{|\bigcup_{u \in \mathcal{U}} RelsRecomm_u|}{|\bigcup_{u \in \mathcal{U}} Rels_u|} \quad \textbf{(32)}$$

It should be noted that the interest coverage metric is another form of relevance coverage, if $TestSet$ is constituted only by positive ratings.

2.5.5 Novelty and Diversity Metrics

As pointed by [Herlocker et al., 2004], some RS produce recommendations that are highly accurate and have reasonably coverage—and are yet useless for practical purposes. As example, they mention that in a grocery store context, recommending bananas would be statistically highly accurate (almost everyone buys bananas), but useless, as most people knows bananas, and knows whether or not they want to buy some, ignoring such recommendation. Much more valuable would be a recommendation for the new frozen food the customer has never heard of—but would love. For this purpose, new dimensions for analyzing RS that consider the "nonobviousness" of the recommendation are being considered in the literature. Examples of such dimension are novelty (and serendipity), which could be defined as the proportion of items unknown to the user that are recommended (and liked), and diversity, which tries to measure the differentiation among items recommended. The measurement of such dimensions is not easy, given that commonly for evaluation there is only a test set corresponding to (some) items evaluated by the user, but in most cases this is not the full set of known items (users do not rate all items they know due to time or other constraints). Moreover, it is not possible to know whether an item will be of interest for a user if he/she does not know it, so there is no information about the full set of interesting items. Despite of these problems, there are different proposals in the literature of metrics to measure these important dimensions. Here we review some of them.

2.5.5.1 Self information-based Novelty

[Zhou et al., 2010] note that novelty in RS is concerned with suggesting items a user is unlikely to know about already, and propose to use the self-information or surprisal, a measure of unespectedness of items relative to their popularity. Let $Rels_i$ be the set of users that consider relevant the item i, then the self-information based novelty of the recommendation set $\mathcal{I}_N^*(u)$ is:

$$nov_u^{self-information}\left(\mathcal{I}_N^*(u)\right) = \frac{\sum_{i \in \mathcal{I}_N^*(u)} \log_2\left(\frac{|\mathcal{U}|}{Rels_i}\right)}{N} \quad \textbf{(33)}$$

And the mean self-information based novelty of a RS on *top*-N recommendation lists is:

$$nov^{self-information}@N = \frac{\sum_{u \in \mathcal{U}_{TestSet}} nov_u^{self-information}\left(\mathcal{I}_N^*(u)\right)@N}{|\mathcal{U}_{TestSet}|} \quad \textbf{(34)}$$

2.5.5.2 Unpopularity-based Novelty

[Fouss and Saerens,] propose a novelty metric which is based in the idea that "best-seller" items (those items frequently rated) are the opposite of novel items (it is expectable that "best-seller" items are mostly known by users). Thus, they measure the rating frequency of the top N items recommended. Being $R_{.,i}$ the set of known ratings assigned to item i (in this case, ratings in train set), the unpopularity-based novelty for a user u among the *top* N items recommended can be calculated by:

$$nov_u^{unpopularity}\left(\mathcal{I}_N^*(u)\right) = \underset{i \in \mathcal{I}_N^*(u)}{\text{median}}\left(\left|R_{\cdot,i}\right|\right) \quad \textbf{(35)}$$

The unpopularity-based novelty of a RS on *top-N* recommendation lists can be calculated by:

$$nov^{unpopularity}@N = \frac{\sum_{u \in \mathcal{U}_{TestSet}} nov_u^{unpopularity}\left(\mathcal{I}_N^*(u)\right)@@N}{|\mathcal{U}_{TestSet}|} \quad \textbf{(36)}$$

2.5.5.3 Intra-list Similarity

With the purpose of balancing *top-N* recommendation lists according to the user's full range of interest, [Ziegler et al., 2005] introduce an intra-list similarity metric that intends to capture the diversity of a list. In their work, diversity may refer to all kinds of features, e.g., genre, author, and other discerning characteristics. Based upon an arbitrary similarity function $sim: \mathcal{I} \times \mathcal{I} \rightarrow \mathbb{R}$ with values of sim incrementing as the similarity between items increment, [Ziegler et al., 2005] define intra-list similarity for a user's ranked recommendation list $\mathcal{I}^*(u)$ as:

$$ILS\left(\mathcal{I}^*(u)\right) = \frac{\sum_{i_j \in \mathcal{I}^*(u)} \sum_{i_k \in \mathcal{I}^*(u), i_j \neq i_k} sim\left(i_j, i_k\right)}{2} \quad \textbf{(37)}$$

Lower values of $ILS\left(\mathcal{I}^*(u)\right)$ means more dissimilar elements in $\mathcal{I}^*(u)$, i.e., better diversity. As noted in their paper, this definition of $ILS\left(\mathcal{I}^*(u)\right)$ is permutation-insensitive, i.e., rearranging positions of items in $\mathcal{I}^*(u)$ does not affect $ILS\left(\mathcal{I}^*(u)\right)$.

2.5.5.4 Set Diversity and Item Novelty

Aiming to identify the difficulty of a recommendation task based on the novelty of items in the user profile, [Zhang and Hurley, 2008; Zhang and Hurley, 2009] propose an item novelty metric which is based upon a diversity measure of items. The diversity function depends on a distance function $d: \mathcal{I} \times \mathcal{I} \rightarrow \mathbb{R}$, which stands for the distance or *dissimilarity* between a pair of items. Thus, the diversity of a set of recommended items $\mathcal{I}^*(u)$ is:

$$div\left(\mathcal{I}^*(u)\right) = \frac{1}{|\mathcal{I}^*(u)|(|\mathcal{I}^*(u)| - 1)} \sum_{i_j \in \mathcal{I}^*(u)} \sum_{i_k \in \mathcal{I}^*(u), i_j \neq i_k} d\left(i_j, i_k\right) \quad \textbf{(38)}$$

A simple distance function may be build using a similarity metric such as cosine similary, thus making $d\left(i_j, i_k\right) = 1 - sim\left(i_j, i_k\right)$. They also propose a computation for measuring the novelty of an item i, as the amount of additional diversity that i brings to the set $\mathcal{I}^*(u)$. Thus, the item novelty of a recommended item is computed by:

$$nov_i^{\mathcal{I}^*(u)} \triangleq |\mathcal{I}^*(u)| \left(div(\mathcal{I}^*(u)) - div(\mathcal{I}^*(u) - \{i\}) \right) = \frac{1}{|\mathcal{I}^*(u)| - 1} \sum_{i_j \in \mathcal{I}^*(u)} d(i, i_j) \quad (39)$$

2.5.6 Time-aware Recommendation Metrics

Untill this point, all metrics reviewed are time-unaware. However, in recent RS literature some time-aware recommendation metrics have been proposed, due to work of Lathia and collaborators. These metrics are described below:

2.5.6.1 Time-averaged RMSE

In 2009 [Lathia et al., 2009b] pose a temporal accuracy metric based on RMSE, which they call *Time-averaged RMSE*, consisting simply in the RMSE computed on ratings made until a particular point of time t:

$$TA\_RMSE_t = \sqrt{\frac{\sum_{r_{u,i} \in TestSet_t} (\hat{r}_{u,i})}{|TestSet_t|}} \quad (40)$$

where $TestSet_t$ is the set of test ratings made until time t, i.e. $TestSet_t = \{r_{u,i,t'} : r_{u,i,,t} \in TestSet \ \wedge t' \leq t\}$.

2.5.6.2 Temporal Novelty and Diversity

In 2010 [Lathia et al., 2010] consider the problem of recommendation through time. A user which assiduously uses a RS may receive the same recommendations over and over, thus devaluating user interest in recommendations. As a way to overcome this problem, the authors argue that diversity should be introduced into recommendations. In order to measure the degree of diversity, they simple measure the level of difference between consecutive recommendation lists presented to a user (at a level N) at different times t_1 and t_2 ($t_1 < t_2$), as the size of their set theoretic difference:

$$diversity@N \left(\mathcal{I}^*_{N,t_1}(u), \mathcal{I}^*_{N,t_2}(u) \right) = \frac{\left| \mathcal{I}^*_{N,t_2}(u) \backslash \mathcal{I}^*_{N,t_1}(u) \right|}{N} \quad (41)$$

where $\mathcal{I}^*_{N,t}(u)$ is the set of *top-N* items recommended at time t. As noted by the authors, one limitation of this metric is that it measures the diversity between two lists, highlighting the extent that users are being sequentially offered the same recommendations, but no clue of how recommendations change in term of new items is given. Thus, they also introduce a novelty metric, which compares the recommendation list ($\mathcal{I}^*_{N,t}(u)$) to the set of all items that have been recommended untill time t, $\mathcal{I}^*_t$:

$$novelty@N \left(\mathcal{I}^*_{N,t}(u) \right) = \frac{\left| \mathcal{I}^*_{N,t}(u) \backslash \mathcal{I}^*_t \right|}{N} \quad (42)$$

Chapter 3. Using Time Information in Recommendation

As described in the previous chapter, there are a number of proposals for incorporating the time dimension into RS. However, the majority of such proposals have been tested only against accuracy metrics (in particular using only MAE or RMSE), leaving other important evaluation dimensions aside. Hence there is the need of make a systematic study of these techniques, establishing how the accuracy improvement (if any) impacts on other metrics, and looking for additional improvements in other recommendation dimensions. This chapter details techniques that have been implemented throughout this work, which includes time-unaware algorithms and time-aware extensions for them, with the purpose of allow a better understanding of how the time dimension have been incorporated. We also suggest some possible novel time-aware extensions.

3.1 K-Nearest Neighbors

The kNN family of algorithms is one of the most widely used techniques in RS. The key idea behind it is to establish a set of similar users (or items), called the nearest neighbors, whose ratings over the target item (user) are then extrapolated in order to compute a rating prediction. As pointed out in Chapter 2, the firsts formulations of this technique in the RS context were user based, in which a neighborhood of users similar to u ($N^{user}(u)$) is determined, and then their ratings on item i are aggregated:

$$\mathcal{F}(u,i) = \underset{u' \in N^{user}(u)}{\text{aggr}} R_{u',i} \tag{43}$$

Many variants for the computation of $N^{user}(u)$ as well as for the aggregation function have been proposed. In general, the set of neighbors of u is determined from a similarity measure, usually assessed from the set of common ratings among users in a pair basis, selecting the k users most similar to u (from there its name kNN). That is:

$$N_k^{user}(u) = \bigcup_{j=1}^{k} u_j' : u_j' = \underset{u' \in \mathcal{U} - N_{j-1}^{user}(u), u \neq u'}{\arg\max} sim(u,u') \quad \text{with} \quad N_0^{user}(u) = \emptyset \tag{44}$$

Additionally, instead of selecting the k nearest neighbors, it is also possible to select all users whose similarity with u is greater than a pre-specified threshold, i.e. $N^{user}(u) = \cup_{u' \in \mathcal{U}, u' \neq u} u' : sim(u,u') \geq \tau_{sim}$. One of the most used approaches for computation of $sim(\cdot,\cdot)$ is based on *correlation* among co-ratings [Adomavicius and Tuzhilin, 2005; Herlocker et al., 1999]. If $\mathcal{I}_{uv}$ represent the set of items co-rated by both user u and v, i.e. $\mathcal{I}_{uv} =$

$\{i \in \mathcal{I} | r_{u,i} \neq \emptyset \wedge r_{v,i} \neq \emptyset\}$, then the Pearson Correlation similarity can be computed by [Adomavicius and Tuzhilin, 2005]:

$$sim(u,v) = \frac{\sum_{i \in \mathcal{I}_{uv}} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in \mathcal{I}_{uv}} (r_{u,i} - \bar{r}_u)^2 \sum_{i \in \mathcal{I}_{uv}} (r_{v,i} - \bar{r}_v)^2}} \quad (45)$$

where $\bar{r}_u$ is the average rating value of user u. This similarity is used as a weight in the aggregation computation. The most common aggregation approach is to use a weighted sum [Adomavicius and Tuzhilin, 2005]:

$$\mathcal{F}(u,i) = \hat{r}_{u,i} = c + \sum_{u' \in N^{user}(u)} sim(u,u') \times r_{u',i} \quad (46)$$

where c is a normalizing factor, usually computed as $c = 1/\sum_{u' \in N^{user}(u)} sim(u,u')$. In order to avoid the selection of users based on too few co-ratings, [Herlocker et al., 1999] introduced a *significance* weighting, in which if the two users u and v had fewer than 50 co-rated items, i.e. $|\mathcal{I}_{uv}| \leq 50$ the computed value of c is multiplied by $|\mathcal{I}_{uv}|/50$.

3.2 Time Decay and Truncation

The basic assumption of these approaches is that users change their preferences as time goes by, so recent ratings better reflect their present tastes. The common way to model temporal concept drift had been by means of incrementally penalizing past data as they become older, in the form of an associated weight, or even eliminating them. One of the first proposals was the "truncation" of the recommendable item set based on a temporal feature of items (e.g. movie production year in the case of a movie RS, as in [Tang et al., 2003]). In a more general conception, this idea is to eliminate from the training data any information not meeting a predefined temporal condition. In the simplest case, where available information consists in the rating matrix and temporal information, this would be:

$$\forall r \in R: r \leftarrow \emptyset \; iif \; temporal\_condition(r) = false \quad (47)$$

where $temporal\_condition(\cdot)$ may be for example $year(r) > 2000$ or $days\_ellapsed(r,t) < \tau_{time}$, where $days\_ellapsed(r,t)$ is the number of days elapsed since r was done untill time t, and τ_{time} is a time threshold (this last scheme is also known as using a *time window*). A less strict approximation is to use an exponential decay factor on the temporal feature of interest (the so-called *time-decay* approach). For example, [Ding and Li, 2005] modified the common rating prediction computation used in kNN (eq. (46)) incorporating a time weighting factor $w(m)$:

$$\mathcal{F}(u,i,t) = \hat{r}_{u,i} \quad (48)$$
$$= c + \sum_{u' \in N^{user}(u)} sim(u,u') \times w\left(days\_ellapsed(r_{u',i}, t)\right) \times r_{u',i}$$

with

$$w(m) = e^{-\delta \times m} \tag{49}$$

where δ is the *decay rate*, set to $\delta = 1/M_0$. M_0 is known as the *half-life* of $w(M)$, a value such that $w(M_0) = (1/2)w(0)$. That is, the weight reduces by $1/2$ in M_0 days. This way, older ratings have less weight in prediction computation.

Note that the normalizing factor c must be accordingly recomputed, i.e. $c = 1/\sum_{u' \in N^{user}(u)} sim(u,u') \times w\left(days\_ellapsed\left(r_{u',i}, t\right)\right)$.

3.3 Temporal CF with Adaptive Neighborhoods

In this kNN extension proposed by [Lathia et al., 2009a], authors note that real, production RS data are constantly updated, but CF algorithms are not designed to adapt to these changes. To address this problem, they propose a method to automatically assign and update per-user neighborhood sizes, which is supposed to outperform a global, static k parameter value (under TA_RMSE metric[22]). The k values are computed by [Lathia et al., 2009a; Lathia, 2010]:

$$\forall u \in \mathcal{U}: k_{u,t+1} = \max_{k \in V}\left(e_{u,t} - TA\_RMSE_{u,t,k}\right) \tag{50}$$

where $k_{u,t+1}$ is the k value selected for predicting ratings of user u in the time interval $]t, t+1]$ (such a time interval may correspond to a few days), V is a set of potential k values to be tested (authors used $V = \{0,20,35,50\}$), $e_{u,t}$ is the TA_RMSE achieved until time t between the ratings in the user profile and predictions actually made (with the k values selected on time intervals before t), and $TA\_RMSE_{u,t,k}$ is the TA_RMSE on the user profile that would be achieved with parameter value k. Thus, (50) selects the parameter value k that maximize the improvement on the current user error. This model can be further generalized to consider, instead of only different kNN parameter values, different CF algorithms [Lathia, 2010]:

$$\forall u \in \mathcal{U}: Alg_{u,t+1} = \max_{Alg \in V}\left(e_{u,t} - TA\_RMSE_{u,t,L}\right) \tag{51}$$

In (51), V is a set of algorithms. A main drawback of this last approximation noted by the authors, is the computation overload of maintaining several CF algorithms running in parallel, and moreover being retrained for each user at each time t. It should be noted however that, to increase accuracy to her highest level, the main approach consists on blending several recommenders [Koren, 2009b]. A drawback for the purposes of this work is the difficulty for testing this method, because its computation considers values obtained at different moments. It would be unfeasible to calculate metrics such as *Precision* or *Recall,* which requires one ranking of recommended items (to the best of our knowledge there are no definitions of how to compute *Precision*, *Recall* and related metrics on successive ranking

[22] Details in section 2.5.6.1.

lists). This problem can be easily solved by using eq. (50) or eq. (51) iteratively for obtaining the best parameter (or algorithm) for each user *at the end of the training period*. Then, the selected value (or algorithm) can be used to compute every rating prediction. We note however that, this approach seems not to have consideration of users' taste change through time, as the k parameter value selected is computed measuring the error over *all* past ratings of the user, i.e. past tastes not currently present have the same importance (for the computation of k value) as users' current tastes.

3.3.1 Instantaneous Adaptive Neighborhoods

Based on Lathia et al.'s work, we propose to use the adaptive neighborhood scheme with a slight modification, in order to speed up the training phase of the algorithm. Our idea is to consider only two temporal bins, the first standing for train set, and the second for test set. We define t_{test} as the time instant that splits both sets, leaving all rating data time-stamped after t_{test} as test set, and the remainder as train set. Then, we apply the adaptive neighborhood formula, but directly minimizing the TA_RMSE on the users' profile, as there will not be different error computations prior to t_{test}:

$$\forall u \in \mathcal{U}: k_{t>t_{test}} = \min_{k \in V} \left(TA\_RMSE_{u,t_{test},k} \right) \quad \textbf{(52)}$$

Then, with the computed $k_{t>t_{test}}$ we generate predictions for items in test set, and further rank items based on such predicted ratings.

3.4 Bias Baseline estimates

As noted by [Koren, 2008], typical CF data exhibit large user and item effects, for example, some users are more exigent and tend to rate movies below their average ratings. One of the lessons learned during the Netflix Prize competition was that these effects must be taken into consideration in order to obtain accurate models for rating prediction [Koren, 2009b]. So, although they might seem simple, baseline estimates based on bias from users and items' ratings may become a powerful tool in the recommending task, moreover combined with other techniques. A simple bias baseline estimate of the rating may be computed by [Koren, 2008]:

$$\hat{r}_{u,i} = \mu + b_u + b_i \quad \textbf{(53)}$$

where μ stands for the global mean rating, b_u is the mean rating bias of user u, and b_i is the mean rating bias of item i. The error of this rating prediction rule is:

$$e_{u,i} = r_{u,i} - \mu + b_u + b_i \quad \textbf{(54)}$$

These parameters can estimated from data using the least squares method with regularization [Koren, 2008; Koren, 2009b]:

$$\min_{b*} J = \sum_{u,i} \left(r_{u,i} - \mu - b_u - b_i\right)^2 + \lambda \left(\sum_u b_u^2 + \sum_i b_i^2 \right) \quad \textbf{(55)}$$

The first term of J looks for finding parameters b_u and b_i that minimize the quadratic error on the known ratings. The second (regularization) term controls the magnitude of these parameters, aiming to avoid overfitting. The minimization problem (55) can be solved using a stochastic gradient descent approach, were the gradient of J is used to determine the direction of steepest minimization (gradient's opposite direction), and the parameter values are updated iterating over each value $r_{u,i}$. The updating equations for (55) are:

$$\begin{aligned} b_u' &= b_u - \gamma \cdot \frac{\partial J}{\partial b_u} \\ b_i' &= q_i - \gamma \cdot \frac{\partial J}{\partial b_i} \end{aligned} \quad \textbf{(56)}$$

with

$$\begin{aligned} \frac{\partial J}{\partial b_u} &= -2e_{u,i} + 2\lambda b_u \\ \frac{\partial J}{\partial b_i} &= -2e_{u,i} + 2\lambda b_i \end{aligned} \quad \textbf{(57)}$$

3.4.1 Bias Baseline Extension: Incorporating temporal biases

It is important to note, as noted in Chapter 2, that such biases can be very time dependent. For example, an item may become popular on a particular season or because an external event (e.g. movies nominated to Oscar). On the other hand, users may change their taste as they become older, thus changing their rating behavior. Based on this idea, additional bias parameters have been considered by Koren [Koren, 2009a; Koren, 2009b]. A baseline estimate of the rating incorporating *time changing* bias can be:

$$\hat{r}_{u,i}(t) = \mu + b_u(t) + b_i(t) \quad \textbf{(58)}$$

Following the analysis of [Koren, 2009a; Koren, 2009b] for the items' case, temporal-aware bias information can be added using specific parameters on a predefined temporal span (i.e. using temporal bins) considering that items' bias change slowly over time:

$$b_i(t) = b_i + b_{i,Bin(t)} \quad \textbf{(59)}$$

In the users' case, Koren considers two biases definitions, one accounting for gradual concept drift, and the second accounting for sudden, day-specific drifts. The latter may model a change in a specific day of user's mood for example, or even the fact that sometimes more than one user uses the same account in a RS. Thus, the user's bias becomes [Koren, 2009a; Koren, 2009b]:

$$b_u(t) = b_u + \alpha_u \cdot dev_u(t) + b_{u,t} \quad \textbf{(60)}$$

The first term correspond to the stationary part of the user bias (i.e. its value holds for the whole time interval analyzed). The second term represents the gradual concept drift of the user, where $dev_u(t) = \text{sign}(t - t_u) \cdot |t - t_u|^\beta$ stands for the time deviation of a rating from the mean rating date of u, t_u, with $|t - t_u|$ measuring the temporal distance in days between the rating date t and t_u [Koren, 2009a; Koren, 2009b]. The parameter β was set by the authors to best fit the data they were using (Netflix dataset), whilst the parameter α_u is learned for each user. The third term in (60) corresponds to the bias on users' ratings on a specific day.

3.5 Matrix Factorization

The Matrix Factorization (MF) technique corresponds to an extension of the Singular Value Decomposition (SVD) approach. As explained in [Lathia, 2010] and [Amatriain et al., 2011], in classical SVD a $n \times m$ data matrix R is decomposed (factorized) into new matrices $N\Sigma M$, with N a $n \times f$ matrix, M a $m \times f$ matrix, and Σ a $f \times f$ diagonal matrix, such that:

$$R = N\Sigma M^T \tag{61}$$

The values of Σ are known as *singular values* or *eigenvalues*, and the columns of N and the columns of M are known as *singular vectors* or *eigenvectors*. An interesting property of the SVD algorithm is that the new matrices split the original values of R into f linearly independent components or factors. This fact was exploited by [Deerwester et al., 1990], where they take a term×document matrix M and used SVD to find latent relations among documents and terms (based on the factors computed by the technique). As noted by the authors, the dimensionality reduction given by SVD (usually $f \ll n, m$) makes it possible to "relate" documents and terms even by terms not present in some documents. Thus, it seems a natural choice to use this technique to find relations among users and items. However, a problem with this approach is that SVD is not well defined for sparse matrices. [Sarwar et al., 2000] used imputation (filling missing values of R with some predefined value, e.g. user mean rating or item mean rating), to allow posterior usage of SVD.

On the other hand, N and M can be used to approximate R. For instance, if R is a rating matrix, the user's rating given to an item can be computed as the dot product between the user's factor vector ($N_{u,\cdot}$ or n_u) and the item's factor vector ($M_{i,\cdot}$ or m_i) [Lathia, 2010]. That is:

$$\hat{r}_{u,i} = \sum_{j=0}^{f} N_{u,j} \times M_{j,i} = n_u^T m_i \tag{62}$$

From the above, N and M can be approximated iteratively to fit R, for example minimizing the Frobenius Norm between the difference on them: $\min \ \|R - NM\|^2$. This have been commonly known as matrix factorization. This way, it is not necessary to use imputation,

as it is possible to use only the known values of R to estimate N and M. However, overfitting can become a huge problem, which can be alleviated using regularization, i.e., penalizing the magnitude of the approximated vectors. The common regularized formulation for collaborative filtering is inspired in minimizing the squared error on the set of ratings [Koren et al., 2009]:

$$\min_{n*,m*} J = \sum_{(u,i \in R)} \left(r_{u,i} - n_u^T m_i\right)^2 + \lambda(\|n_u\|^2 + \|m_i\|^2) \quad \textbf{(63)}$$

Different algorithms exist to compute this kind of factorization, with prominence of the *alternating least squares* and the *stochastic gradient descent* approaches [Koren et al., 2009]. A widely used implementation of stochastic gradient descent was published by Simon Funk[23] in the context of the Netflix Prize. In this implementation, for each known rating, the parameters are optimized by updating them in the opposite direction of the gradient of the optimization criterion, using a *learning rate* parameter γ which controls the amount of update [Koren et al., 2009; Takács et al., 2008]:

$$\begin{aligned} n_u' &\leftarrow n_u - \gamma \cdot \frac{\partial J}{\partial n_u} \\ m_i' &\leftarrow m_i - \gamma \cdot \frac{\partial J}{\partial m_i} \end{aligned} \quad \textbf{(64)}$$

3.5.1 MF Extension: Adding Temporal Biases and Temporal Factors

In an outstanding paper selected for Best Research Paper Award at KDD[24] 2009, Koren [Koren, 2009a] describes the incorporation of biases, temporal biases and temporal user-item interaction factors into a MF model (he also discusses the incorporation of temporal dynamics into a neighbor model). The biases and temporal biases incorporated are the same discussed in section 3.4, only being needed to adapt the learning rates and regularization constants. The factorization model with temporal biases, leads to the following rating prediction:

$$\hat{r}_{u,i}(t) = \mu + b_i(t) + b_u(t) + n_u^T m_i \quad \textbf{(65)}$$

with $b_i(t)$ and $b_u(t)$ as defined in (59) and (60) respectively. The discussion in the paper about temporal dynamics in user-item interactions highlights that as users are due to personal changes in their tastes, factors describing their rating behavior are more likely to be prone to temporal effects than factors related with items. Thus, they apply a modeling similar to those used on the user bias effects on the user factors' modeling, leading to [Koren, 2009a]:

[23] http://sifter.org/~simon/journal/20061211.html
[24] ACM SIGKDD Conference on Knowledge Discovery and Data Mining

$$\hat{r}_{u,i}(t) = \mu + b_i(t) + b_u(t) + n_u^T(t)m_i \quad \textbf{(66)}$$

with:

$$n_{u,j}(t) = n_{u,j} + \alpha'_{u,j} \cdot dev_u(t) + n_{u,j,t} \quad \textbf{(67)}$$

The full model leads to the following minimization problem:

$$\begin{aligned} \min J = \sum_{(u,i,t \in R)} & \left(r_{u,i,t} - \mu - b_i - b_{i,bin(t)} - b_u - \alpha_u dev_u(t)^{\beta} - b_{u,t} \right. \\ & \left. - (n_u^T + \alpha'_u \cdot dev_u(t) + n_t^T)m_i\right)^2 \\ & + \lambda \left(\sum_i b_i^2 + \sum_i b_{i,bin(t)}^2 + \sum_u b_u^2 + \sum_u \alpha_u^2 + \sum_u b_{u,t}^2 \right. \\ & \left. + \|n_u(t)\|^2 + \|m_i\|^2 \right) \end{aligned} \quad \textbf{(68)}$$

which can be solved using the stochastic gradient descent algorithm discussed previously. It is important to note that, although the author in [Koren, 2009a] do not give further details about parameter values, it is argued in another publications of the Netflix Prize winning team [Koren, 2009b; Koren and Bell, 2011] that each regularization term (as $\sum_i b_i^2$, $\sum_i b_{i,bin(t)}^2$, etc.) should be associated with a different regularization constant (λ_1, λ_2, etc.). Moreover, even the learning rates should be individually adjusted in order to obtain the better improvements in the minimization. We also remark that authors used a slightly different factorization model than (63), which they called SVD++ [Koren, 2008], that includes a set of factors accounting for "implicit" information (which items had been rated by each user instead the rating values given). In this work we are left with the model as in (63) because our aim is to study the effect of the incorporation of time-aware terms into the factorization model, instead to compare different factorization models.

3.6 AutoSimilarity in Time

Having time-stamped rating data may bring the opportunity to treat this information as time series. However, a first problem with such an approach is to define which dimensions of observation should be used. Observing single user's ratings pattern does not seem as a good approach, as he/she rates different kind of items, and thus they are not comparable. The same applies when considering the pattern of a single item being rated by users with different tastes and interests. If we consider a pattern of ratings on a (user, item) pair basis, then we only have one rating (normally users rate items once). [Min and Han, 2005] faced this problem, proposing to use a (user, item category) pair basis rating pattern. They used an item hierarchy scheme to assign each item into a category, thus obtaining a temporal rating pattern for each user on each item category. An example of a simple categorization scheme in the movies domain is to relate each movie with its genre, though other more advanced schemes may be used.

Using the ratings of a user on items of a particular category enables to compute a rating value on the category by [Min and Han, 2005]:

$$cat\_r_{u,cat} = \sum_{i \in cat} \frac{r_{u,i}}{|r_{u,i}|} \quad \textbf{(69)}$$

where $cat\_r_{u,cat}$ is the categorical rating of user u to the item category cat. In order to obtain a time series, the rating data are divided into different time intervals, and for each interval the category ratings are computed[25]. Once having the time series, many of the existing methods of time series analysis may be used. In this case, we stay with the method proposed in [Min and Han, 2005], which tries to identify the moment when a concept drift occurs based on the user auto similarity, that is, analyzing how similar are the category ratings between different time intervals. With this purpose they used the Pearson correlation coefficient on category ratings of the same user, but on different time intervals:

$$AS(u, t_i, t_j) = \frac{\sum_{cat}(cat\_r_{u,cat,t_i} - \overline{cat\_r_{u,t_\iota}})(cat\_r_{u,cat,t_j} - \overline{cat\_r_{u,t_j}})}{\sqrt{\sum_{cat}(cat\_r_{u,cat,t_i} - \overline{cat\_r_{u,t_\iota}})^2} \times \sqrt{\sum_{cat}(cat\_r_{u,cat,t_j} - \overline{cat\_r_{u,t_j}})^2}} \quad \textbf{(70)}$$

where $AS(u, t_i, t_j)$ is the auto similarity of user u between time intervals t_i and t_j, $cat\_r_{u,cat,t}$ is the category rating of user u on category cat during time interval t, and $\overline{cat\_r_{u,t}}$ is the mean category rating of user u on all categories during time interval t. Given a similarity threshold value τ_{sim}, if $AS(u, t_i, t_j) < \tau_{sim}$ then it is concluded that user u changed his/her tastes on time interval t_j. Following [Min and Han, 2005] pose, if no change is detected, ratings may be predicted using e.g. the standard kNN formulas. When a change is detected, the similarity computation among users is modified, differentiating weights according to rating time. If the rating was performed before the concept drift, then they are given a lower weight. In the particular case presented in [Min and Han, 2005], ratings after concept drift are over weighted w.r.t. the rest of the ratings proportionally to the auto similarity value computed:

$$sim(u, v) = \frac{\sum_{i \in \mathcal{I}_b}(r_{u,i} - \overline{r_u})(r_{v,i} - \overline{r_v})w_b(u) + \sum_{i \in \mathcal{I}_a}(r_{u,i} - \overline{r_u})(r_{v,i} - \overline{r_v})w_a(u)}{\sqrt{\sum_i (r_{u,i} - \overline{r_u})^2} \times \sqrt{\sum_i (r_{v,i} - \overline{r_v})^2}} \quad \textbf{(71)}$$

where $\mathcal{I}_b$ and $\mathcal{I}_a$ are the set of items rated by user u before and after u's concept drift respectively (which is considered to occur at time t_j), and w_b and w_a are the weights assigned to them accordingly (assigned with values 1 and $1 + 0.5|AS(u, t_{j-1}, t_j)|$ in the original paper respectively). Note that for similarity computation, the individual item ratings are used instead of category ratings.

Using this modified similarity computation, neighbors are selected and predictions are generated using the already described kNN formulas.

[25] Hopefully, there will be enough ratings in each time interval and category as to compute a confident average value.

3.6.1 Time Series Auto Similarity Adjusted Time Decay

Taking advantage of information provided by the computation of Auto Similarity of item categories' Time Series data, which indicates at which time a user is changing his/her taste, we propose to compute a Time Decay algorithm variation, which uses such information to personalize the decay rate value used. The proposed algorithm computes AutoSimilarity of each user's category rating as explained above, detecting at which time interval each user changes his/her tastes. Neighbor selection is done with the common Pearson similarity formula (eq. (45)). Then we apply Time Decay formula (eq. (48)) to compute rating predictions, using instead a fixed δ for every user, a personalized δ_u computed by $\delta_u = 1/days\_ellapsed(r_u + \Delta t, t_i)$, where $AS(u, t_{i-1}, t_i) < \tau_{sim}$, and Δt is a weight adjusting factor used to model that, if a taste change occurs during time t, then ratings in that period should have more than the half of the weight of a recent rating. This scheme is similar on essence to the one described in [Cao et al., 2009], where graphs of similarities between movies are built, and then interest on time series rating data of related items are observed to detect changes on ratings through time, in order to discard ratings previous to a user's change (we remark anyhow that the way of generating time series data and detecting taste change differ notoriously from [Cao et al., 2009]).

3.7 Clustering

Although clustering has not been a popular approach for recommendation due to a decrease of performance on statistical accuracy metrics, some works highlight its scalability on CF, which would allow a fast recommendation generation on nowadays huge RS' databases. Considering that this is one of the most popular data mining techniques, we decided to include some CF clustering algorithms to contrast them against other techniques, as a first step to devise how they could be extended to cope with temporal rating information.

3.7.1 Clust-kNN

[Rashid et al., 2007] presents a simple though effective clustering approach for CF. They formed *l* user clusters, and used the clusters centroids as *surrogate* users. Thus each surrogate user is a vector whose components are the average rating values for each item on all users of a cluster. In order to compute a prediction, the algorithm computes the similarity between the active user and each surrogate user (instead of every other user in the system), thus enabling a faster rating prediction. From this point, the algorithm performs like a traditional kNN algorithm, using the computed surrogates. To compute the clusters, they used an improved variant of the popular k-Means algorithm called Bisecting k-Means, which performs iteratively a 2-Means clustering on the largest cluster to split until getting the desired number of clusters. In our implementation, we tested three different clustering algorithms (k-Means [MacQueen, 1967], Expectation-Maximization (EM) [Dempster et al., 1977] and Bisecting k-Means [Rashid et al., 2007]). As these algorithms depends on initialization values (initial centroids in KMeans variants, and initial parameters in EM), we test with different starting points for each algorithm, selecting the models with the lowest Davies-Bouldin Index, which attempts to measure the separation between clusters[Davies and Bouldin, 1979]:

$$DB = \frac{1}{n}\sum_{i=1}^{n}\max_{i\neq j}\left\{\frac{S_c(c_i)+S_c(c_j)}{d(c_i,c_j)}\right\} \quad (72)$$

where $S_c(c)$ is the average distance of all objects in cluster c to the cluster centroid, and $d(c_i,c_j)$ is the distance between clusters centroids. Following we briefly explains the different clustering algorithms used.

3.7.1.1 k-Means Clustering

This clustering algorithm aims to partition a set *n* of vector-valued objects $\{o_1, o_2, \cdots, o_n\}$ into k clusters ($k < n$) ($= \{c_1, c_2, \cdots, c_k\}$ so as to minimize the sum of squares of objects within clusters:

$$\arg\min_{c}\sum_{i=1}^{k}\sum_{o_j\in c_i}\|o_j-\mu_i\|^2 \quad (73)$$

where μ_i is the mean of objects in c_i. Note that the Euclidean distance is applied (although others could be used). In order to compute the partition c, the algorithm performs teo steps iteratively: 1) assign each object to the cluster with the closest centroid; 2) recompute new centroids as means of the objects in each cluster. The initial centroids can be assigned with randomly selected objects and the iteration ends when assignments no longer change. This way this algorithm performs a local optimization, whose result will depend on the initial centroids selected.

3.7.1.2 Expectation Maximization Clustering

This algorithm aims to estimate parameters of *k* distributions (representing *k* clusters) which maximize the likelihood that objects come from these distributions. Given the parameter values, and assuming particular distributions (e.g. normal distribution), it is possible to compute the probability that an object belongs to each cluster [Witten et al., 2005]:

$$p(c_j|o_i) = \frac{p(o_i|c_j)\cdot p(c_j)}{p(o_i)} = \frac{f\left(o;\mu_{c_j},\sigma_{c_j}\right)p(c_j)}{p(o_i)} \quad (74)$$

where $f(o;\mu,\sigma)$ is the normal distribution function. Similarly to k-Means, this algorithm starts with an initial guess of the parameter values, and then iteratively computes 1) probabilities of objects coming from the distributions (expectation step); and 2) computation of distribution parameters which maximizes the (log) likelihood of the distributions given the data (maximization step), which can be computed as the product of the probabilities of each object coming from each distribution (cluster). The iteration ends when increase in (log) likelihood becomes negligible.

3.7.1.3 Bisecting k-Means

This algorithm iteratively 1) picks a cluster to split; 2) applies a 2-Means clustering to produce 2 sub-clusters; and 3) repeat step 2 j times and then selects the best split. The iteration

ends when the k clusters have been found. As different selection criteria can be applied on steps 1) and 3), we follow [Rashid et al., 2007] who suggest to pick the largest remaining cluster in 1) (initially all objects are in one cluster), and to select the cluster with best (maximum) intra-cluster similarity in 3).

3.7.2 MF and Time Aware MF Clustering

Following the previous idea, it is possible to cluster the user (or item) factor matrix generated by the matrix factorization method (instead the rating matrix) in order to detect similar users. Being MF a method able to find latent relations among users, it is expectable that clustering on this reduced data could form different yet meaningful clusters compared to the clusters generated directly from rating data. Moreover, clustering may be performed on the factor matrix generated with the consideration of temporal information (see section 3.5.1), which could be considered as form of time aware clustering.

3.7.3 Cluster AutoSimilarity in Time

[Min and Han, 2005] describe a clustering-based approach for computing users' AutoSimilarity on Time(see section 3.6). This alternative scheme computes each user assigned cluster at time t, $c(u,t)$, based on rating data, and the distance between consecutive (in time) users assigned clusters, $Dist\big(c(u,t_{i-1}), c(u,t_i)\big)$. Given a similarity threshold value τ_{sim}, if $Dist\big(c(u,t_{i-1}), c(u,t_i)\big) < \tau_{sim}$ then it is concluded that user u changed his/her tastes on time interval t_i. In such a case, eq. (71) is used to compute similarities between users (items), and predictions are computed as in traditional kNN.

3.7.4 Time Series clustering

Another way of using clustering on temporal data is to cluster the time series derived from the rating data, as described in section 3.6. There are many ways to cluster time series data. For example, it is possible to compute many temporal representations e.g. Piecewise Aggregate Approximation, which in turn can be the input to a non-temporal clustering method (see Annex 3).

3.8 kNN on Factors Matrix

Similarly to the idea described in section 3.7.2, the users' (or items') factors matrix can be used to find similar users (items). Thus, a kNN like approach can be applied for selecting nearest neighbors based on factors information, and then compute ratings predictions from rating data as in traditional kNN, but using the factors' induced neighbors. This scheme have been proposed previously by [Paterek, 2007].

3.9 Time Aware CF Proposal: Time Influence

Unlike the previous approaches, in the search of interesting relations among users taking advantage of temporal information (which may lead to neighborhood formation), we theorize about the existence of a time influence relation between some users. Consider a couple of users where one of them (user u) consistently rates items after the other user (user v) have rated the same items. This may indicate that the first waits to see the rating

behavior of the second. Using this idea, we could define a time influence coefficient, which should be positive if the ratings of u (on the same movies rated by v) are done consistently after the ratings of v (u is time-influenced by v), and zero in other case.

If we analyze theoretically such coefficient, we hypothesize that its highest value should occur when 1) each rating of u_a is made after the corresponding rating of v, and 2) the ratings of u are done on the short period of time possible after v (u rates the item as he/she sees that v rated it). The latter can be expressed as the portion of time influence (*PTI*) of the ratings $r_{u,i}$ and $r_{v,i}$ (ratings of user u and v for item i respectively), that is:

$$PTI(r_{u,i}, r_{v,i}) = \begin{cases} 1 - \dfrac{TimeDiff\left(D(r_{u,i}), D(r_{v,i})\right)}{|TimeInterval|} & if\ D(r_{u,i}) \geq D(r_{v,i}) \\ 0 & if\ D(r_{u,i}) < D(r_{v,i}) \end{cases} \quad (75)$$

where $D(r)$ returns the date of rating r, $TimeDiff(d_1, d_2)$ returns the difference between dates d_1 and d_2, and $|TimeInterval|$ is the total length of the time interval analyzed. Currently we consider as time unit the *day*, but others can be used (e.g. *week* or *hour*).

Considering the above, we define the *Time Influence* coefficient between any two users that have rated some common items as:

$$TI(u, v) = \frac{1}{|I_u \cap I_v|} \times \sum_{i \in \mathcal{I}_u \cap \mathcal{I}_v} PTI(r_{u,i}, r_{v,i}) \quad (76)$$

where $\mathcal{I}_u$ are the items rated by user u.

The above expression gives an idea about the influence that a user has on another user, but the coefficient only takes into account the time at which the rating is performed, leaving out the rating value itself. If ratings are no correlated, it is possible that a high TI value is due to fortune. Thus, a more careful assessment should include this information. The previous coefficient can be extended into a *Time Influence Correlation*, incorporating an estimation of the correlation of rating values:

$$\rho_{TI}(u, v) = \rho(u, v) \times TI(u, v) \quad (77)$$

where $\rho(\,)$ is a standard correlation function, such as the Pearson product-moment correlation coefficient. Currently we are running experiments to determine the applicability of this scheme (and possible adjustments).

Chapter 4. Experiments and Results

4.1 Introduction

The different approaches reviewed so far are based on a variety of algorithmic techniques, data transformations, model assumptions, etc. These different approaches may have dissimilar effects on the multiple evaluation dimensions that a RS is subject to. Moreover, the extension of models allowing them to take advantage of temporal information, commonly with the goal of accuracy increase, may have dissimilar impacts on other evaluation dimensions (and even accuracy itself may be impacted differently, depending upon the particular techniques used). In order to bring a more complete view of how the different time aware extended recommendation models (and the corresponding "base" models) behave on classical and newer evaluation dimensions, we performed experiments comparing the models. This section describes the experimentation and results obtained, starting with a brief description of the experimental setting, including a depiction of the used dataset and the evaluation protocol. Then the results of each algorithm along evaluation dimensions are presented and discussed (a detailed description of the used metrics was presented in section 2.5).

4.2 Experimental setting

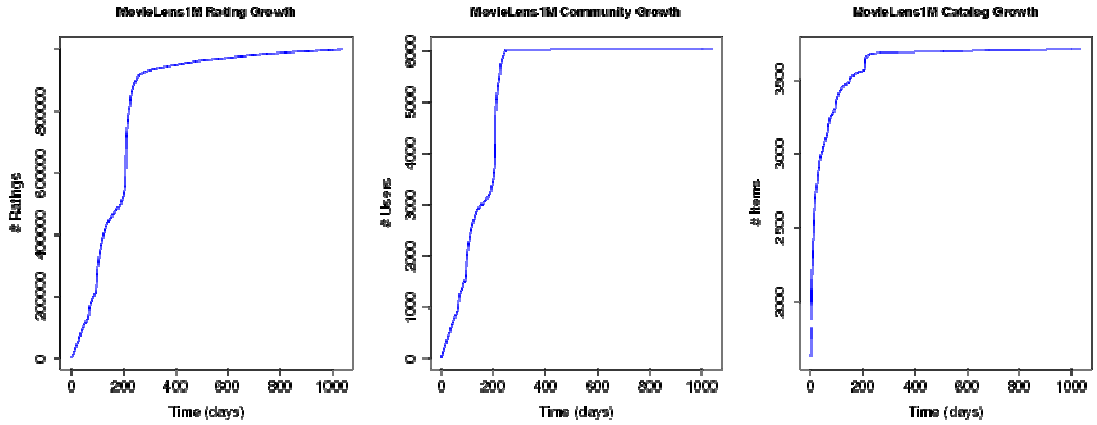
4.2.1 Dataset

As the focus of this work is to compare different time-aware recommendation models, the most basic requirement for a dataset to be used is that, besides the dataset includes the rating data with user and item identifiers, each rating must have a timestamp, in order to allow algorithms to detect temporal effects. Over this initial condition, other information may be incorporated, which could be used by the algorithms to improve their performance, e.g. demographic user data, additional information about items, other kind of user feedback different from ratings (e.g. comments) with their corresponding timestamps, etc. On the other hand, the usage of a publicly available dataset is desirable, as it facilitates the comparison with prior approaches, and allows other scientist to replicate results.

Given all the posed requirements, we selected for experimentation the MovieLens (*ML*) Datasets, a family of three datasets generated by the GroupLens Research Group, publicly available on the Internet[26]. The ML Datasets are compounded by a small-size dataset with 100.000 movie ratings (the so called ML *100K*), a medium-size one with 1.000.209 movie ratings (ML *1M*) and a large-size one with 10.000.054 movie ratings (ML *10M*). All the three have associated timestamps for each rating, and also have additional information of movies (at least title and genre). Also, every user has at least 20 ratings in each dataset. We decided to use ML1M dataset, given it is particularly well suited for our purposes. This da-

[26] http://www.grouplens.org/node/73

taset contains ratings over a timespan of three years (from April 26th, 2000 to February 28th, 2003) from users who joined ML during year 2000, that is, ratings on 2001 and 2002 correspond to a follow-up of these users; this way, this dataset have enough users, items and ratings as to detect temporal trends in metrics, as Figure 4 shows. It can be seen as well that, most items are rated during the first year, a fact also reflected in the catalog size, which after the first year stays almost fixed.



a) Rating Growth b) Community Growth c) Catalog Growth

Figure 4. Rating, Community and Catalog Growth of ML 1M Dataset

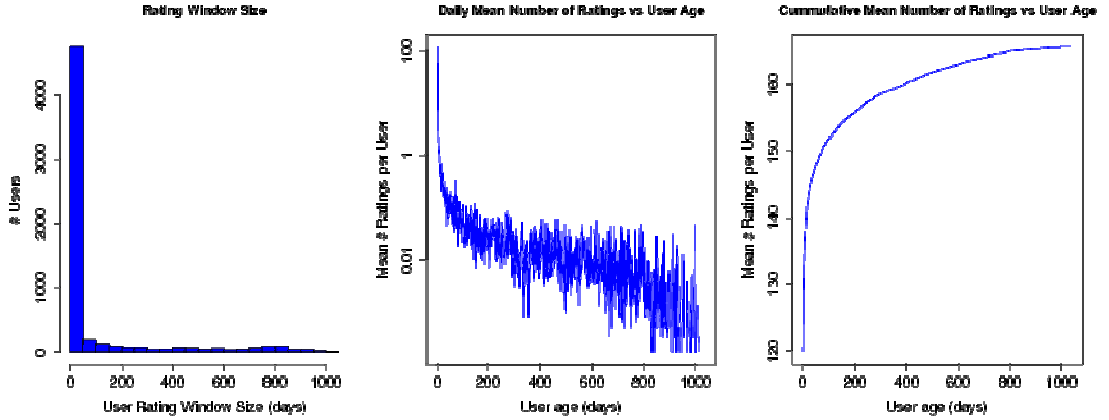
Table 1 shows some statistics about the dataset:

Dataset	ML 1M
# Ratings	1.000.209
# Users	6.040
# Items (movies)	3.706
Mean (±standard deviation) # ratings per user	165.6 (±192.75)
Mean per user rating value	3.70
Mean (±standard deviation) user rating window (days)	94.79(±221.79)
Mean (±standard deviation) # ratings per item	269.89(+-384.05)
Mean per item rating value	3.24
Mean (±standard deviation) item rating window (days)	793.15(±294.66)
Rating period (days)	1039

Table 1. Statistics of ML 1M dataset

As may be seen in Table 1, there is a considerable variation on the number of ratings given by each user and received by each item. For our work it is prominent the fact that, the user rating window (i.e. the number of days between the first and the last rating in the dataset) is very small (3 months in mean), comparing to the item rating window. This effect can be better understood by looking at Figure 5. It shows clearly that most users make the vast majority of ratings immediately after their incorporation into the system. For most users, their rating window (time elapsed between their first and last rating in the dataset) is less than 50 days (Figure 5a). In fact, an average of 120 ratings are made during a typical user's first day in the system, rapidly decreasing to less than 1 rating per day on subsequent days (Figure 5b). Consequently, the cumulative mean number of ratings per user increases more

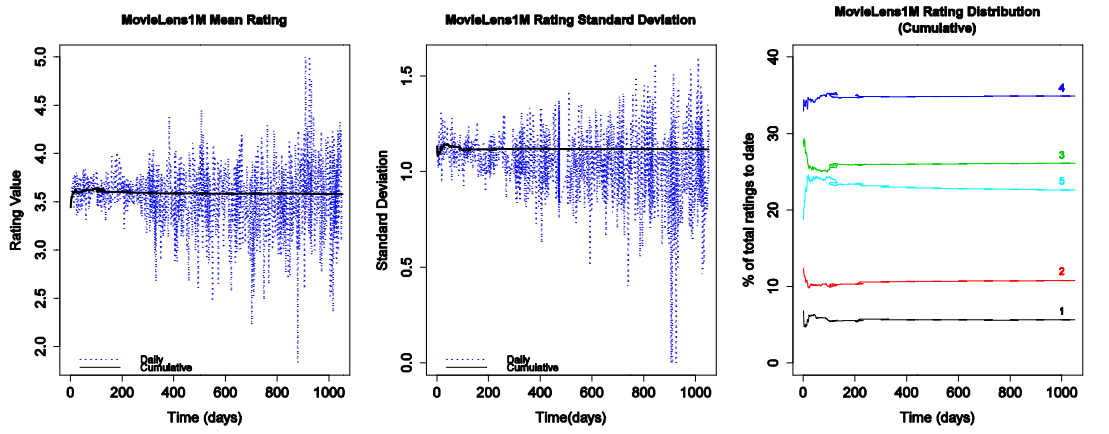
slowly as time goes by (Figure 5c). It also shows that users with longer rating windows (i.e. "older" in the system) tend to have a higher total rating count, though not much different from users with shorter rating windows.



a) Rating Window Size b) Ratings per User Age (daily) c) Ratings per User Age (Cumulative)

Figure 5. Temporal Analysis of Users' Rating Volume

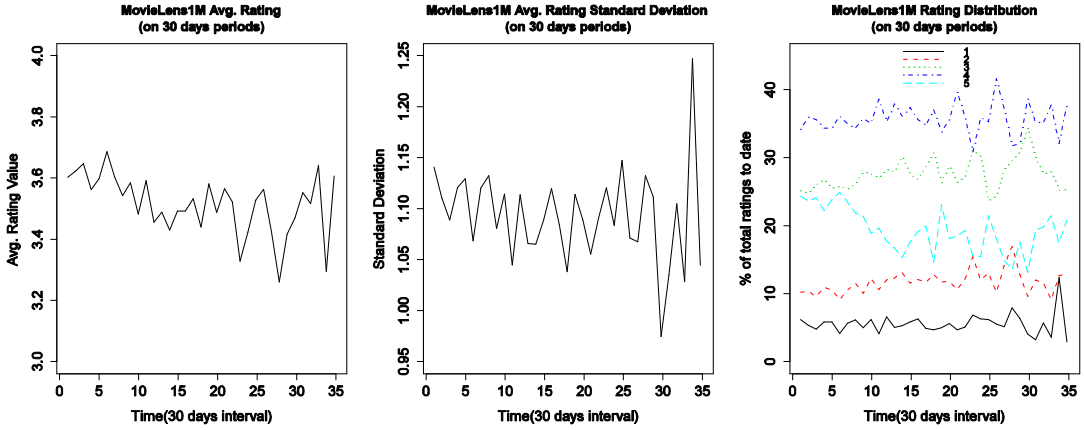
Figure 6 shows the evolution of the mean rating value through time, cumulative and non-cumulative, in a daily basis (a), the standard deviation of rating value (b) and the cumulative rating distribution (proportion of each rating value out of the total ratings per day) (c). It can be seen that, even though there is a great difference between values on one day and the next, when the computation is made on cumulative values, they tend to stabilize. Moreover, it seems a kind of decreased level of variation during the first year (Figure 6 (a) and (b)), but that is just an effect due to the dense proportion of ratings made during that period. Given that on the subsequent period there are less ratings (from different users), mean values shown a greater difference.



a) Mean Rating Value b) Rating Standard Deviation c) Rating Distribution (cumulative)

Figure 6. Temporal Analysis of User Ratings

Figure 7 is similar to the previous one, but computed over 30-days periods. This allows seeing fluctuations on different periods that the daily or cumulative plots do not admit. When analyzed on these period it is notable for example the fluctuations of the rating values distribution on final periods, which show some differences in tendencies w.r.t. initial periods. This can have an effect in recommenders' measured performance if, for instance, RS are trained with data having some particular distribution, and the test data have a different distribution.



a) Average Rating Value b) Rating Standard Deviation c) Rating Distribution (non-cumulative)

Figure 7. Temporal Analysis of User Ratings (computed over 30-days periods)

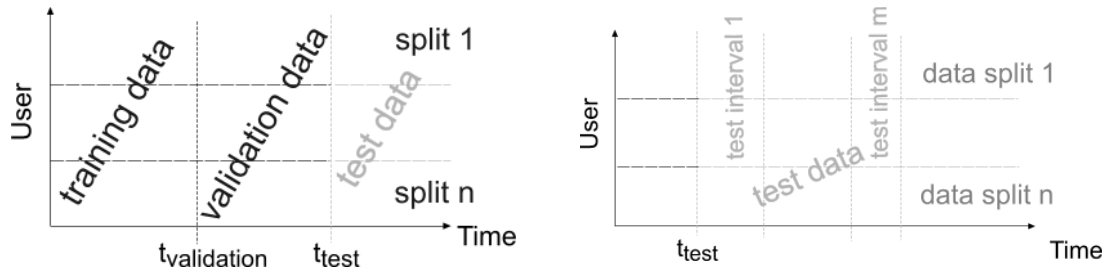
4.2.2 Evaluation Protocol

As a basic evaluation protocol, the dataset must be divided into a train and test sets. We consider that, when a RS is used in a real-world application it only can use past data to predict the present (or near future) taste of the users, the natural division scheme should be date-based, taking as training data those ratings done until a predefined date t_{test}, and leaving as test data those ratings after such a date. Although it may be seen as a basic requisite for RS performance analysis, as noted by [Lathia, 2010], most works in the area make the data division without a temporal perspective, whilst data changes as time goes by (as Figure 7 shows). Arguments for not to take into account the timestamps are the incremented data sparsity and the observed user behavior of rating most items at incorporation time (see Figure 5). For instance, the renowned and influent Netflix Prize competition used as testing data a predefined number of the most recent ratings (i.e. last ratings) of each user [Bennet and Lanning, 2007]. However, this latter approach no way ensures that the train set does not have "future data". The "last ratings" of one user may have been performed in different dates from other user's "last ratings", as considered in the training set of the Netflix Prize dataset, which has been used previously to test some of the time-aware algorithms implemented in this work. So we stay with the "date-based" data division scheme.

We also would like to perform a sort of cross validation. However, as discussed previously, the common *n-folds* splitting scheme, with ratings randomly divided into *n* splits (that is, *n-1* splits used as training data and the one left as testing data, without considering rating timestamps) is not a choice for this task, as it does not ensure a time-ordering among splits.

One possibility could be to make time-ordered splits, that is, ordering all ratings based on their timestamps, and then assigning them into each split sequentially, as in [Fu and Silver, 2004]. This option implies selecting for experimentation a number of the $m < n$ first splits (designed for simplicity with the starting time of ratings within them, e.g. $t_0, t_1, \cdots, t_{m-1}$) as training data, and use the subsequent split (t_m) as test data. A second experimental scheme could be using as training splits t_1 through t_m and t_{m+1} as test data, and so on until using split t_n as test data. Although we think this is a much better data division scheme, we discarded it as we wanted to get a scheme which allows us to cross-validate results using the same t_{test} on all splits[27].

Considering the above mentioned, we decided to make a user-based subsampling of the dataset. We selected different users in different time-ordered "folds" (or data splits), which allow forming time-ordered data splits. Within this scheme, ratings from users in each split prior to t_{test} become the training set, whilst ratings posterior to t_{test} from users in the same split form the test set, i.e. recommendations generated with the training set can be contrasted against the same users' future ratings (posterior to t_{test}). This scheme also allows building an additional validation set (required by some algorithms) whose ratings lie in between training and test ratings. Figure 8 (a) shows a schematic view of the proposed data division design.



a) General data division design. b) Test data division design.

Figure 8. Proposed data division design.

An important consideration to this scheme is that new users/items may get incorporated into the system during the test phase (i.e. their first rating date is posterior to t_{test}). In such a case, if the algorithms tested cannot make recommendations for new users (for which there is no training data), then such users are not valid for comparison purposes. Of course, it is always possible to use a recommender wrapper, which in needed cases returns a prediction making use of an alternative method (e.g. based on items' mean rating for new users), but then we would not be measuring the output from the original recommender. Conveniently, users (and items) in the selected dataset got incorporated during the first year. On the other hand, it is also possible that a user in the training set do not have data to validate or test. Moreover, some users may only have data in the validation set given the small rating window of many of them. Although this additional data may help some algo-

[27] This scheme allows such comparison selecting for training, for example, different subsets of the *m* first splits. However, we preferred to leave the study of such a scheme (and possible side effects due to split selection) as future work.

rithms to find e.g. neighbors to use during prediction, their tastes will not be contrasted. Thus, we decided to select only those users with at least 1 rating in each interval.

Considering the previous scheme (with $t_{validation}$ =January 1st, 2001 and t_{test} =June 15th, 2001), 504 users from MovieLens 1M set met the selection criteria. We divided them randomly into 5 subsets of 100 users each and form a split with 4 out of the 5 subsets. Thus we got 5 different overlapping samples of 400 users. We call these the *time-aware splits*.

Additionally to the above-mentioned, we sought a way for measuring the impact over time. At this point we identified two main designs that could satisfy this purpose. The first is based on varying the size of training data (in terms of data's encompassing time interval), i.e. to generate several train/test sets varying t_{test}, and then training each recommender on each training set, in order to evaluate their results on the corresponding test set. This scheme requires training each recommender *m* times (where *m* is the number of train/test sets generated). The second is to generate a unique training set, and generate several test sets (for example, dividing the whole test time interval into several test time sub-intervals). This way, we could assess the ability of a recommender to predict future ratings *whitout obtaining new training data*. We selected the latter for two reasons: First, from a scientific interest viewpoint, to assess recommenders output in a novel way (to the best of our knowledge, there are no works on RS field applying such an evaluation scheme). Second, from a practical viewpoint, it allows us avoid repeatedly train several recommenders (note that we have a training set for each data split). Figure 8 (b) shows schematically this design. Using 30-days intervals, we obtained 21 test intervals. In order to have a more complete view, we generated two test sets per time interval, one using test data of each test interval alone (non-cumulative data), and a second using test data from t_{test} untill the end of each test interval (cumulative data).

To finalize, there was a last decision to make: In order to perform the *top-N* recommendation task within our scheme, it is necessary to generate predictions for a set of items, whose ratings are then ordered to produce the *top-N* recommendations. So, a list of items whose ratings are to be predicted must be selected. Although the ideal approach is to consider the full training item set, in practice a common approach is to consider only a subset of it. Some research considers only the set of items with known ratings on test of each user; given this list is usually short, we consider this approach not fear (In practice, a RS do not know which subset of items will be rated by a user). On the other extreme is the approach of generating ratings for all known items, but it result on very time consuming experimentation (although is almost the only possibility in practice). We selected a compromise on both, using the set of items whose rating is known in the full test set for any user.

4.3 Implementation details

4.3.1 kNN based Models

As noted in Chapter 2 , a problem with this algorithm is given by cold start settings, or moreover with unpopular items, which are rarely rated. In such case it is hard to find a set of co-raters, hindering the computation of similarity (either in user-based or item-based approach), and thus obtaining few neighbors. It could be the case that few users had rated

a particular item, and thus even with a set of user neighbors identified, as very few (or none) neighbor had rated the item, prediction computation is unfeasible (this is similar in item-based approach). In such cases, a common heuristic is to use a default value, as the user mean rating, or item mean rating. However in this work, for algorithm comparison purposes, we consider such cases as impossible to recommend, i.e. no prediction is computed in those cases (we are confident only when the algorithm finds at least 2 neighbors that had rated the item being predicted). For the computation of ranking-based metrics, such items are placed at the end of the ranking, ordered by their identification number. We note that all ties are break by the identification number of the items, following the scheme of the *trec_eval* utility. All the above mentioned holds for all similarity-dependant algorithms.

Regarding the k value, we tested with several values, and finally set it to k=200, as it got the lowest RMSE on the test set, and provided a good coverage.

4.3.2 Time Decay and Time Truncation Models

These approaches were implemented on a Weighted Pearson Item-based kNN base scheme. We selected $M_0 = 250$ days for Time Decay model, and used the last 12 months of training data for Time Truncation model. Both of these values were selected after testing with several different values, as they provided good RMSE values.

4.3.3 Bias Baseline Models

These models are trained iteratively using the stochastic gradient descent method with regularization. This implies the need to adjust parameters needed by the method (see eqs. (55)-(56)), as different parameter values affect the final model obtained. Although some general values are detailed in the literature, such parameter values can be optimized for the particular data at hand. Thus, we used an implementation of the Nelder-Mead (Simplex) optimization method provided by the *Institut für Mathematik* of the *Technische Universität Berlin*[28].

4.3.4 Matrix Factorization Models

In general, as more factors are incorporated, the better precision a MF model deliver. However, we used only 10 factors in our tests ($f = 10$). We decided to use a low value because the dataset we are dealing with is small, and to allow the fast computation of the optimization procedure. We are aware that more extensive testing is required, particularly regarding higher dimensional models. With respect to training phase, here we also use stochastic gradient descent method with regularization. Parameter values were also optimized using the Nelder-Mead optimization method.

4.3.5 Clustering Models

We used a fixed number of 40 clusters to be formed on the different tested clustering variants. For all variants, we repeated the clustering process at least 10 times with different seed values, and kept the clustering result with lowest Davies-Bouldin Index.

[28] Available at http://www3.math.tu-berlin.de/jtem/

4.3.6 AutoSimilarity Models

These models are very sensitive to the similarity threshold value τ_{sim}. We tested several values for this parameter, and finally set it to $\tau_{sim} = -0.1$, as it allows to detect more possible taste changes of users. We also tested different weight values w_a and w_b, and finally decided to use same values as in the original proposal at light of RMSE results.

4.4 Results

This section presents the results obtained with all the tested algorithms, obtained from averaging results on 5 data splits generated from the ML 1M dataset according to the evaluation protocol detailed in section 4.2.2. Aiming to facilitate its analysis, results have been grouped according to the task and related metrics. Additionally, for each metric we first present a comparative among all tested algorithms, and then we focus on sub sets of related algorithms (kNN-variants, MF & Bias Baseline variants, and clustering variants.). All results are presented for non-cumulative and cumulative test data (test data was divided in 21 time intervals, see section 4.2.2).

In order to have a more confident analysis of results, we also computed statistical significance of differences between pairs of algorithms (on interesting cases) using the Wilcoxon Signed Rank Test [Bauer, 1972] with 95% confidence. As we seek to establish what algorithm are the best performing ones, we use a variant of the test that evaluates if one algorithm consistently obtain higher metric values than the other; thus, statistical significance differences presented imply that the first mentioned algorithm statistically have higher metric values than the second one. Statistical test were computed on results on the first test set (derived from the first test time interval), that is, first 30 days starting from t_{test} (which we call Test Interval 1, TI1), and on the full cumulative test set, that is,624 days or approximately 1 year and 10 months after training period (which we call Full Test Interval, FTI) as we consider them the most interesting ones.

4.4.1 Rating Prediction Error

Figure 9 shows a general overview of algorithms' performance through time, in terms of RMSE. This plot corresponds to non-cumulative data. Figure 10, shows the same results on cumulative data. For comparison purposes, we have included the results of a non-personalized, popularity-based recommender (Popularity, which recommends the most popular train items to all users), and a random recommender (Random).

It may be seen that, using this metric all algorithms perform far better from random or from the non-personalized, popularity based recommender. In general, clustering variants have higher RMSE than other personalized algorithms, whilst MF algorithm performs the better, particularly on the cumulative results (in the non-cumulative case, in some particular intervals MF is beaten by other algorithms, as will be detailed later).

All user-neighbors based algorithms perform worst than their item-based counterparts on kNN based variants, so we only included results from the latter. Item-based kNN and MF can be considered as baseline algorithms, as they are not time aware and have very good RMSE results.

As one of the most interesting findings, all algorithms have a very similar performance through time, even when predicting ratings made more than a year after the last training rating considered. Moreover, most of them have similar tendencies on the different intervals (i.e. most of them show a drop in performance in the same intervals). It seems that changes in test ratings' distribution affect algorithms almost equally; in other words, this appears to reflect that, after all users' taste do not change too much individually, but general rating behavior can be affected by temporal situations (e.g. changes in popularity of items), although we have not identified the particular circumstances causing it. Anyhow, it calls out attention that we do not find a major degradation on accuracy as time goes by.

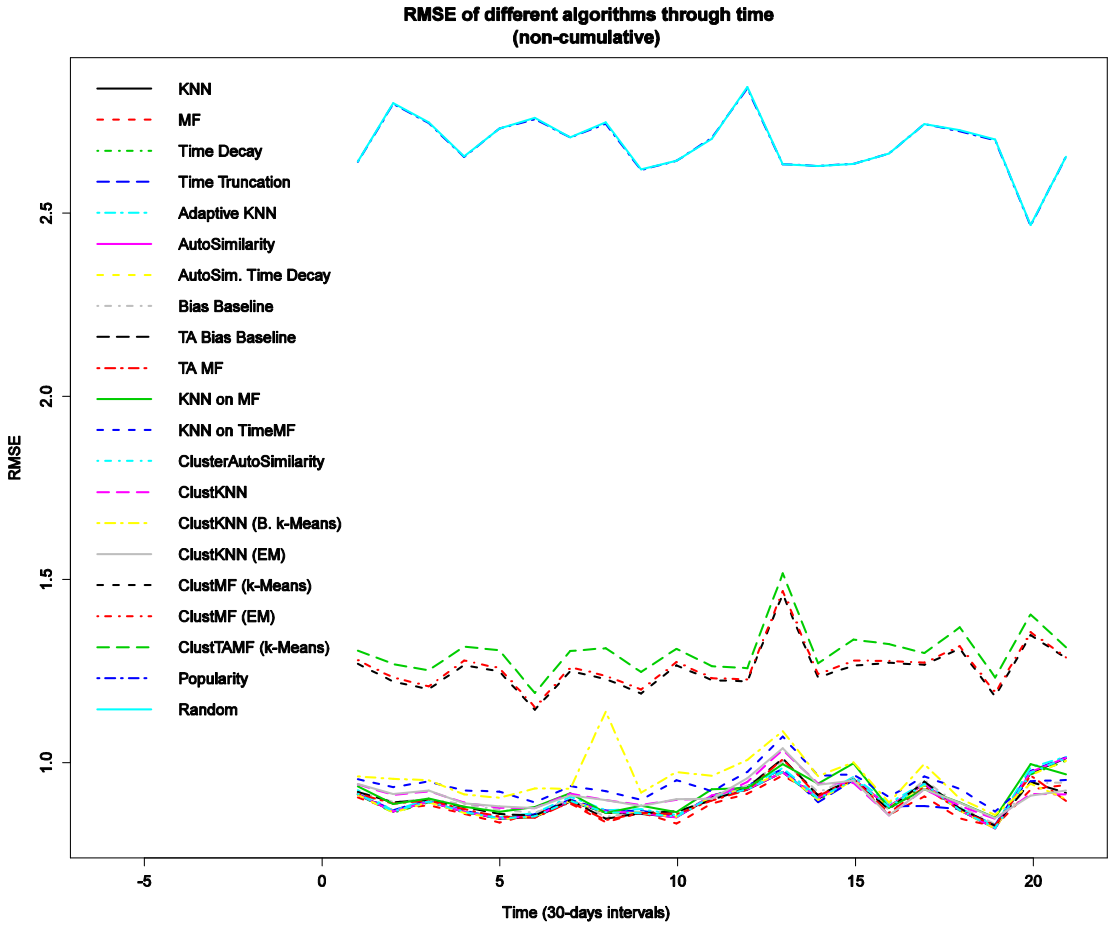


Figure 9. General overview of algorithms RMSE performance through time (non-cumulative data)

Figure 11 shows the more detailed view of RMSE performance of the different algorithms tested. The left column of the figure shows non-cumulative results, whilst right column show cumulative results. Regarding kNN variants (first file in figure), most notably the Time Decay method is able to outperform the Weighted Pearson kNN (KNN in plot legend) algorithm, although not strongly on TI1 (statistically significant differences, s.s.d. from now on, on 2 out of 5 data splits on TI1 and on 5 data splits on the FTI). Albeit average results show that Time Decay is still far from MF RMSE, the statistical test show that the difference is not so strong (s.s.d. on 1 data split on TI1 and on 3 data splits on FTI). In the non-cumulative case, there is no absolute winner algorithm, although MF is the one with lower RMSE on most intervals. The AutoSimilarity algorithm has results similar to

kNN (only 1 data split presents s.s.d. on TI1), whilst Time Truncation is behind them (3 data splits on FTI and 1 data split on TI1 present s.s.d. w.r.t. kNN), and Adaptive-kNN remains last (s.s.d on 4 data splits on TI1 and on 5 data splits on FTI w.r.t. kNN). The AutoSimilarity Adjusted Time Decay (AutoSim. Time Decay in plot legend) algorithm performs similarly to Time Decay, not being able to take advantage of the detection of the time at which users change their taste (there are no s.s.d. on any data split).

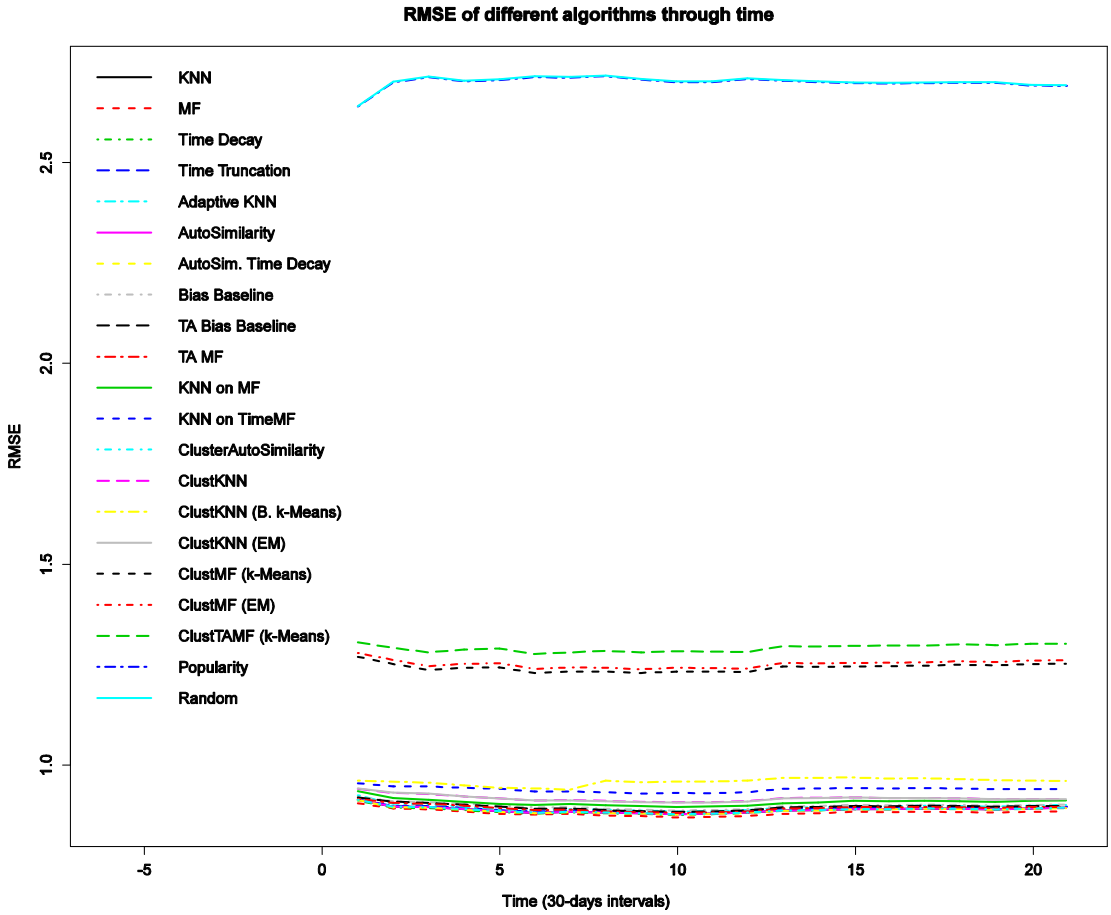
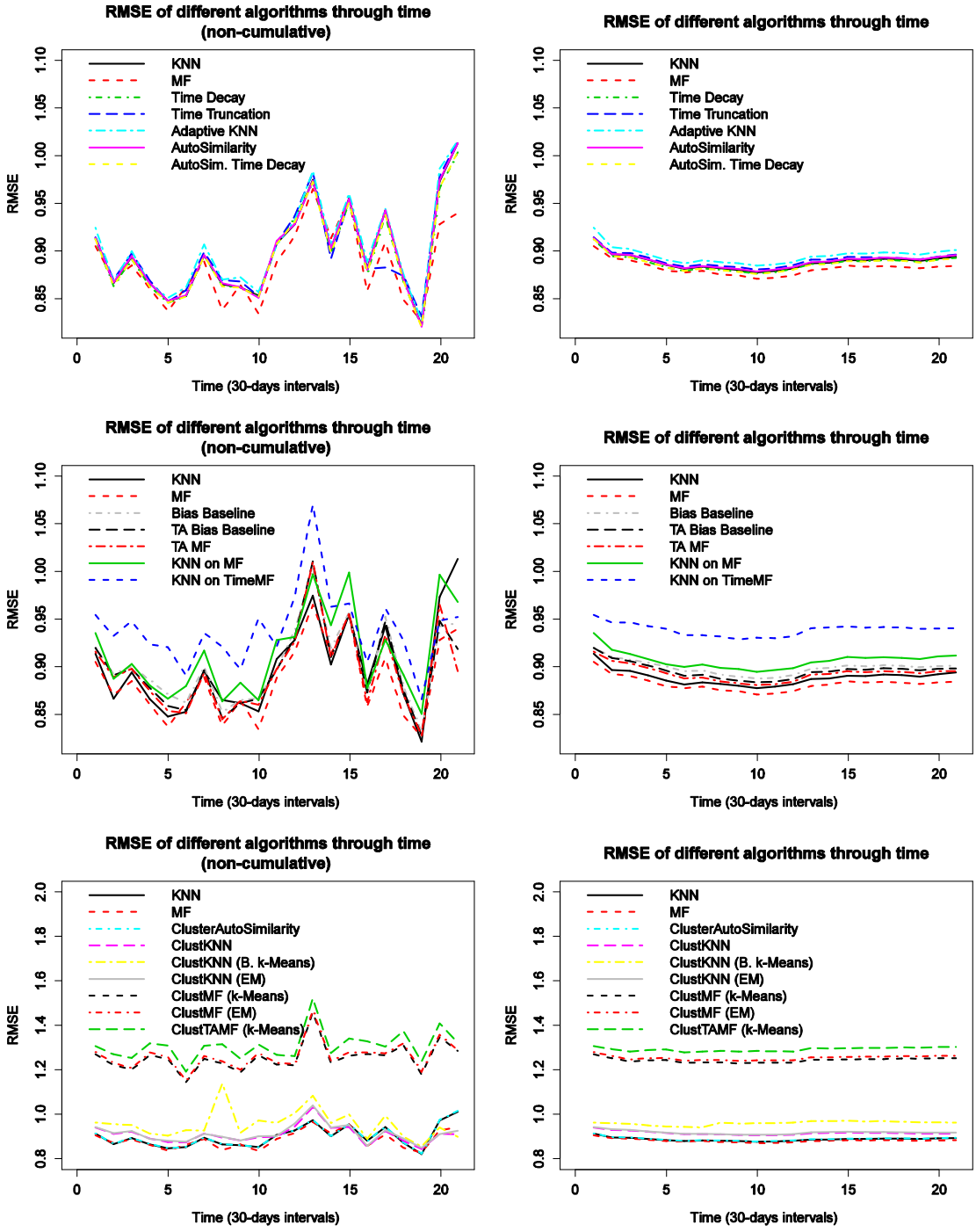


Figure 10. General overview of algorithms RMSE performance through time (cumulative data)

Respecting MF and bias baseline models (second row in figure), we note that all time-aware variants tested perform worst than basic MF, and even worst that item-based kNN (Time Aware MF -TA MF in plot legend— presents s.s.d. w.r.t MF on 3 data splits on FTI, and on 1 data split on TI1). Actually, this result was not unexpected, given that these variants work by heavily fitting into data. We must remember that most of their formulations were made in the context of the NetFlix prize in which, as already discussed in section 4.2.2, last ratings of users were used as test data, meaning that the algorithm is able to take advantage (learn) from some ratings made on prediction time, which in our setting is not allowed. Thus, as MF is less fitted into data, it is able to better predict users' future ratings. We consider this an important finding, which motivated a poster submission to RecSys conference to be held this year. We also note that kNN presents s.s.d. w.r.t. MF on 4 data splits on FTI and on 2 data splits on TI1.



a) Non-cumulative results. b) Cumulative results.

Figure 11. Detailed view of RMSE performance through time

With respect to clustering-based implementations (third row in figure), it is possible to see that all tested implementations (which correspond to item-based variants) except Cluster AutoSimilarity perform worst than kNN and MF algorithms, being the worst those that cluster items using as attributes the MF resulting factors (Cluster Time Aware MF – ClustTAMF in plot— and Cluster MF variants). Although Bisecting k-Means is supposed to provide better results, our experiments showed consistently a better behavior of the k-Means algorithm (s.s.d. on 5 data splits on the FTI, and on 4 data splits on TI1). It may be

due to the fact that Bisecting k-Means tends to provide more similar-sized clusters, and thus, each surrogate becomes more a mixture of different preferences (from different users), whilst k-Means tends to hold on each cluster only the most similar users to each other, thus surrogates can be the result of less different preferences (although there may be some very small clusters). However, this particular aspect should be studied more deeply. As in the case of AutoSimilarity, the ClusterSimilarity algorithm is not able to take advantage of the additional information provided for making recommendations (s.s.d. w.r.t. kNN only on 1 data split on TI1). As may be seen, in this case, clustering recommenders that use time-aware information are not able to improve accuracy.

4.4.2 Top-N Recommendation (Ranked Recommendations)

We count with many metrics for comparing algorithms output when we consider the *top-N* recommendation task. Given that 1) Precision and Recall are the most used IR measures; 2) our consideration that people, particularly on RS, only see a reduced part of a list of choices; and 3) we found that relative performances of algorithms were similar on Precision and Recall, we selected for analysis P@5 metric. Additionally, as AUC reflects the probability that relevant items are positioned over irrelevant items, which complement the general view of performance on the top-N recommendation task, thus we also include it for analysis.

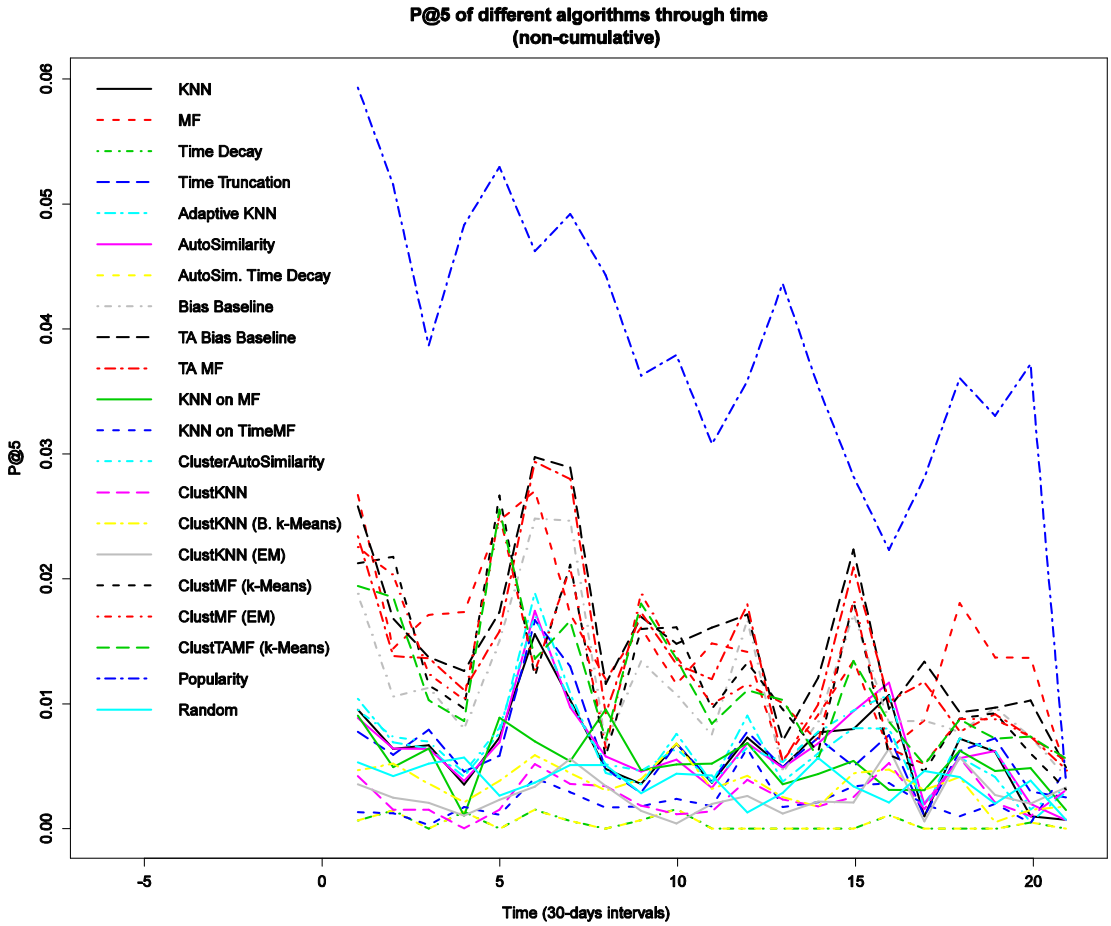


Figure 12. General overview of P@5 performance through time (non-cumulative data)

Figure 12 and Figure 13 show the general P@5 performance of all algorithms on non-cumulative and cumulative data respectively. In this case, we may see that all algorithms perform worst than the non-personalized, popularity based recommender. Moreover, there are algorithms that perform worst than a Random recommender, in particular Time Decay variants (s.s.d. between Random and Time Decay on 3 data splits on TI1 and on all data splits on FTI). There seems to be a clear tendency on users to more heavily rate popular items. It is interesting to note that the second best performing algorithm is Time Aware Bias Baseline, particularly on cumulative data (s.s.d between Time Aware Bias Baseline and MF on 3 data splits on FTI and on none data split on TI1). The MF algorithm also stands, and interestingly it seems that the Time-Aware MF algorithm is able to outperform MF on average values on last test intervals. These results may imply that, the Time Aware models are able to better detect long-term preferences of users than their specific rating behavior (considering that the MF model had a better performance than its Time Aware counterpart and Time Aware Bias Baseline on RMSE). We note that only 1 data split presents s.s.d. on the FTI when comparing MF and Time Aware MF.

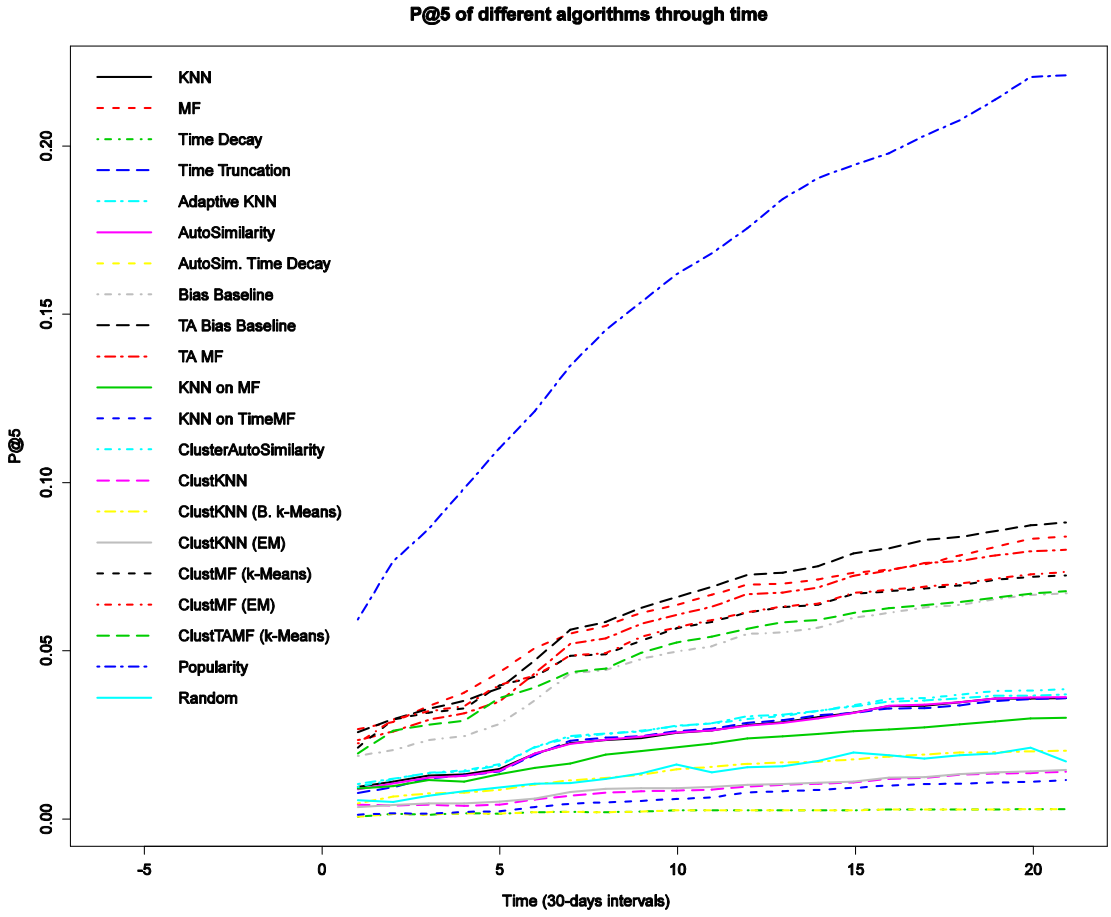
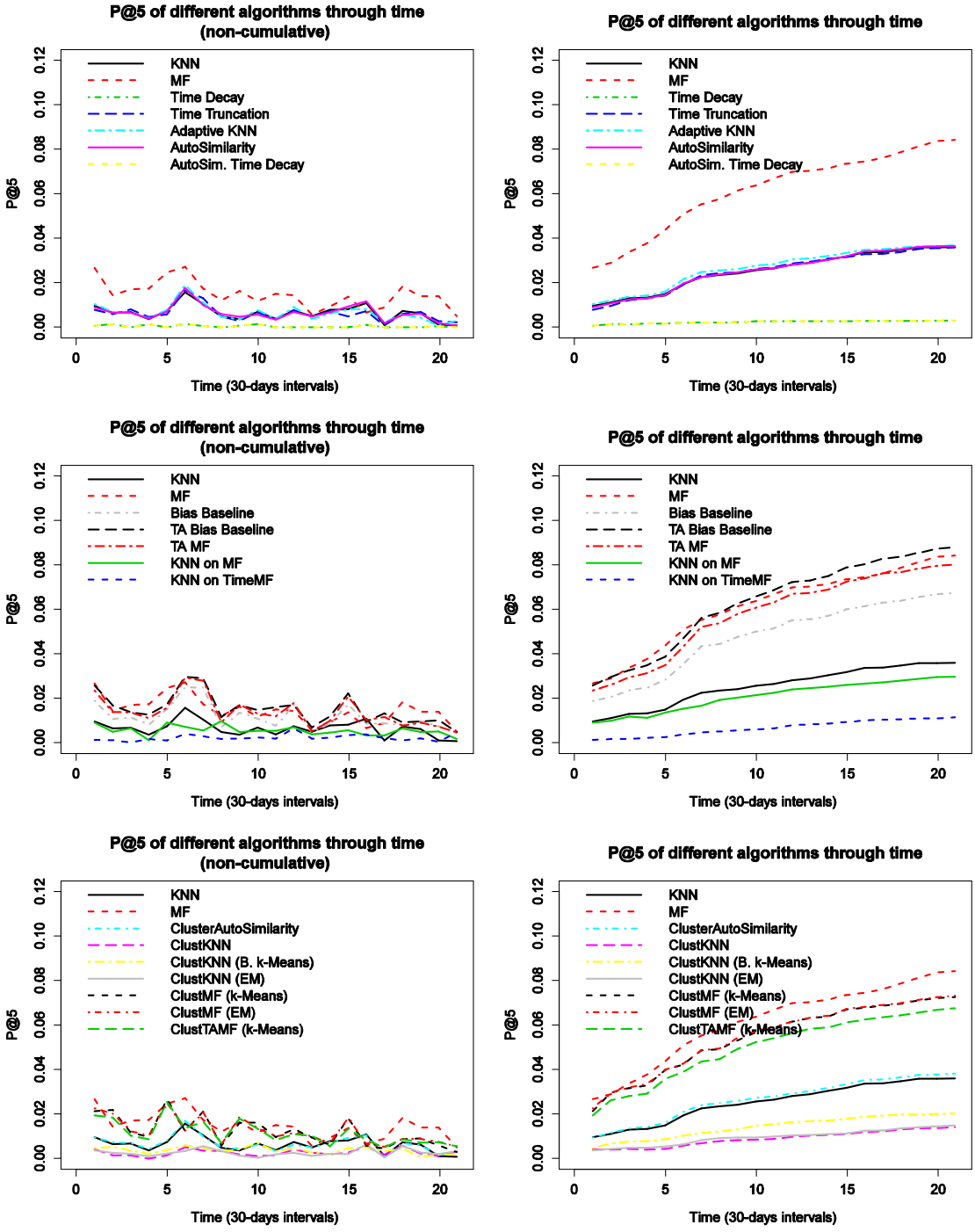


Figure 13. General overview of P@5 performance through time (cumulative data)

Figure 14 shows the detailed P@5 performance on the different groups of algorithms. From this figure we may appreciate that kNN variants are bad choices for the top-N recommendation task, and in particular Time Decay variants have a poor performance (s.s.d. between kNN and Time Decay on all data splits on FTI and 4 data splits on TI1). We also

note that there are not big differences between kNN and MF based kNN (s.s.d. on none data split on TI1 and on 2 data splits on FTI). On the other hand, Cluster kNN variants show poor performance on this metric, differently from Cluster MF variants, which have results similar to MF (s.s.d between MF and Cluster MF (k-Means) on 1 data split on both TI1 and FTI).



a) Non-cumulative results. b) Cumulative results.

Figure 14. Detailed view of P@5 performance through time

Figure 15 through Figure 17 show algorithms' AUC performance. It is interesting to note that, again most algorithms have worst performance than the popularity based recom-

mender, although in this case Cluster MF and Cluster Time MF variants rival best performance results (s.s.d. between Popularity and Cluster Time MF algorithms on 3 data splits on FTI and on 1 data split in TI1), a result consistent with the observed on P@5. On the other hand, although Time Decay variants again show a poor performance, they perform better than Random under this metric (s.s.d. between Random and Time Decay algorithms on all data splits on both TI1 and FTI). It may show that, although Time Decay algorithms have a bad performance for detecting the top preferences of users as showed by P@5 (considering that we impose a heavy restriction on the amount of preferences contemplated, just the first 5 of them), these algorithms in general have a better chance when ordering a long list of items for each user. Anyhow, we consider this result not as meaningful as P@5, given that it is expectable that users focus only on the very first items of a recommendation list.

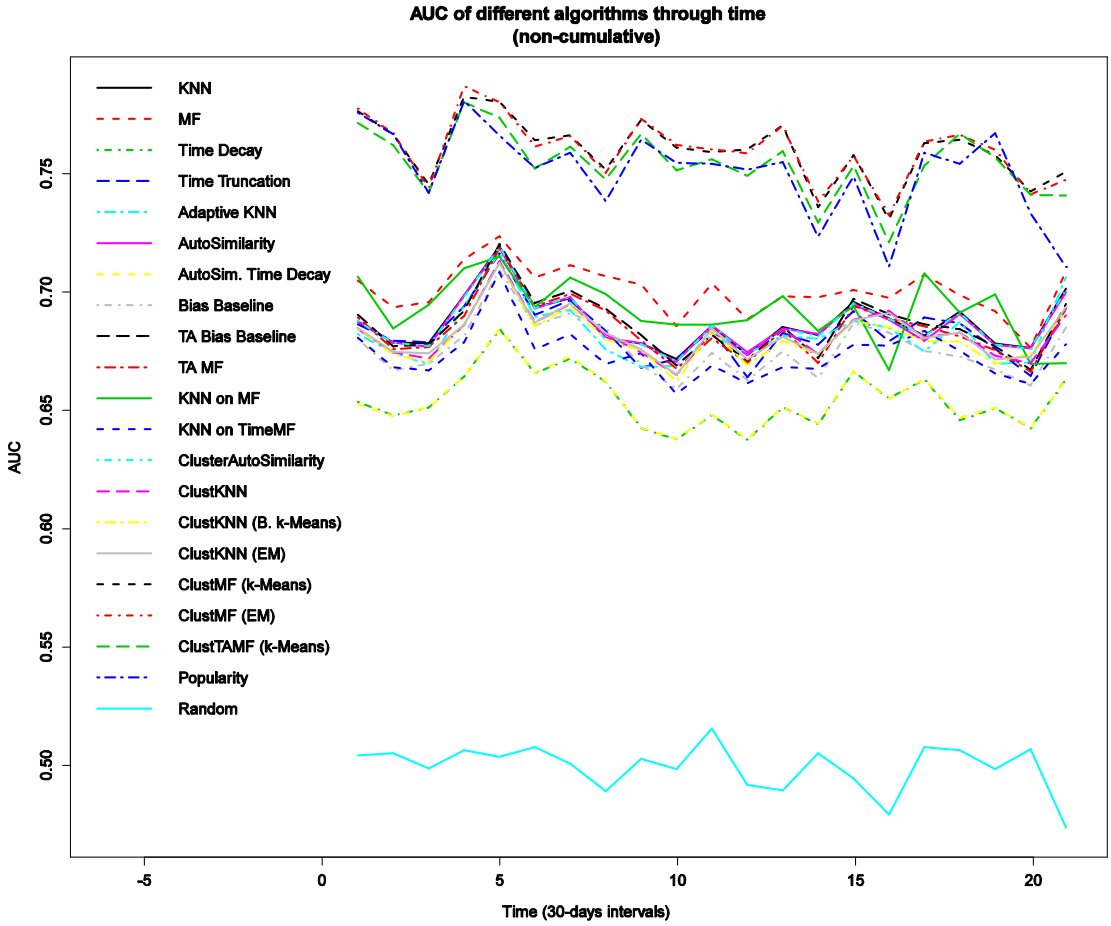


Figure 15. General overview of AUC performance through time (non-cumulative data)

The detailed view reveals that, in case of kNN variants, Time Decay algorithms perform the worst, which is consistent with P@5 observed behavior (s.s.d. on all data splits both on FTI and TI1).

In the case of MF and Bias models, it is possible to see a better performance of Time-Aware Bias model w.r.t. Bias model (s.s.d. on 5 data splits both on FTI and TI1). But,

Time-Aware MF model is outperformed by MF model (s.s.d. on 5 data splits on FTI and on 4 data splits on TI1). The kNN on MF algorithm is able to rival MF (s.s.d. between Mf and kNN on MF on 2 data splist on FTI and on none data split on TI1).

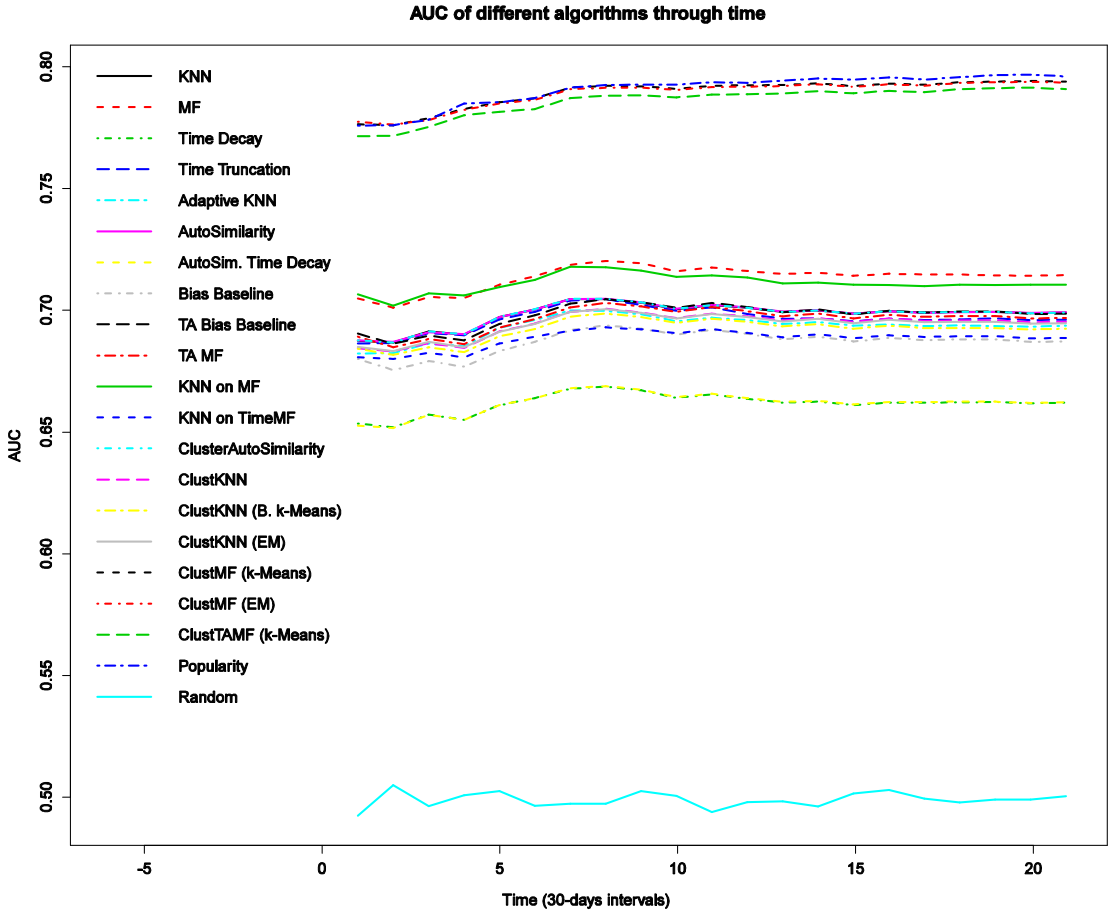
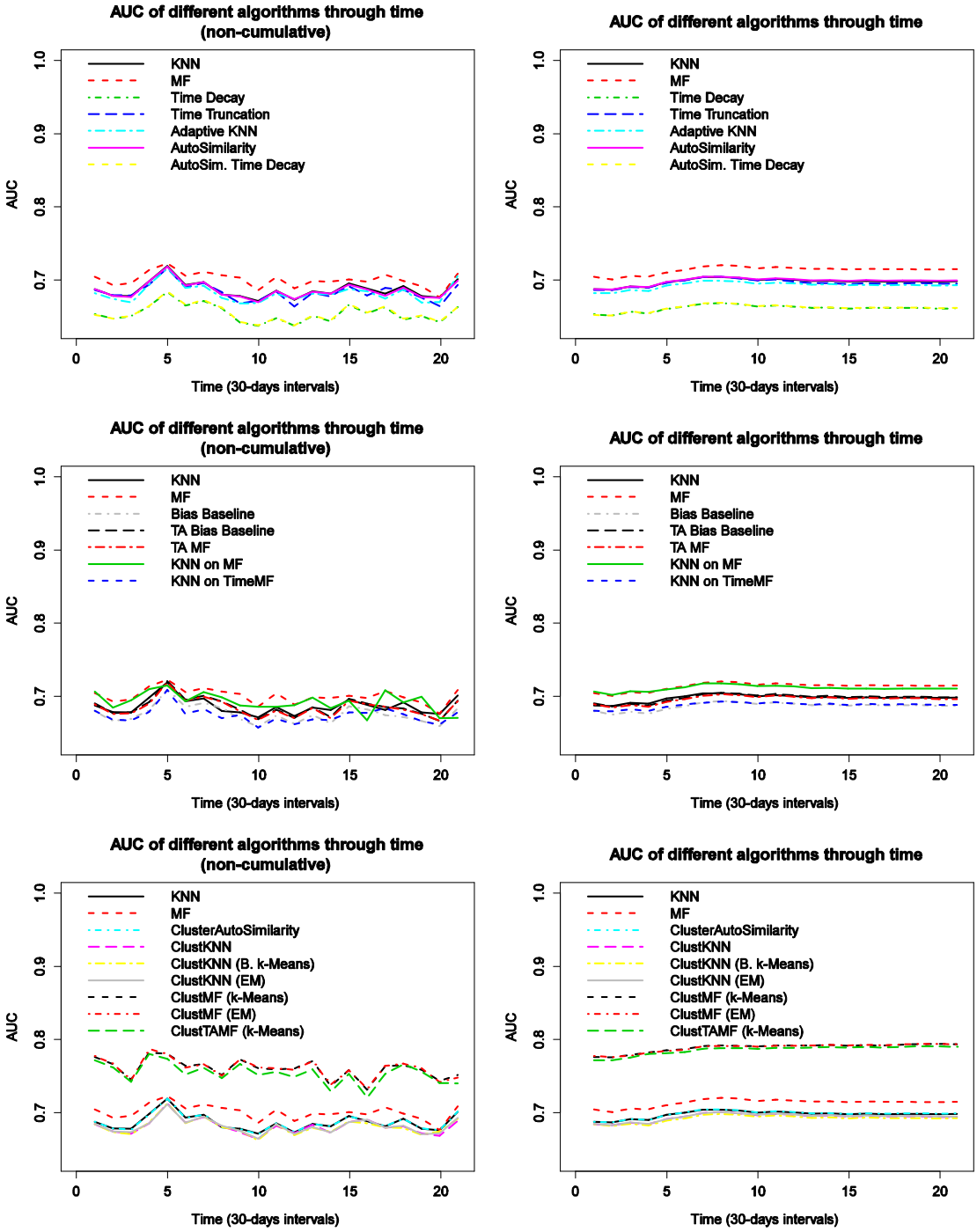


Figure 16. General overview of AUC performance through time (non-cumulative data)

Regarding Clustering-based algorithms, it is interesting to note that variants which cluster the results of MF are able to outperform results of the MF model on cumulative data, with statistical significance (s.s.d. between Cluster Time MF and MF algorithms on all data splits both on TI1 and FTI).



a) Non-cumulative results. b) Cumulative results.

Figure 17. Detailed view of AUC performance through time

4.4.3 Novelty and Diversity

We selected Self-Information (SelfInf) and Content-Based Intra List Similarity (ILSCB) as novelty and diversity metrics. The former have been used previously to formalize the generic novelty of items (lower values indicate decreased novelty), whilst the latter allows getting an idea of how similar are the items in a recommendation list, in terms of their declared content (lower values indicate increased diversity). In particular, in this implementation we used the genre(s) of the movies (as provided in the ML 1M dataset) as declared

content. They are measured on the first 5 recommended items for congruency with previously shown metrics.

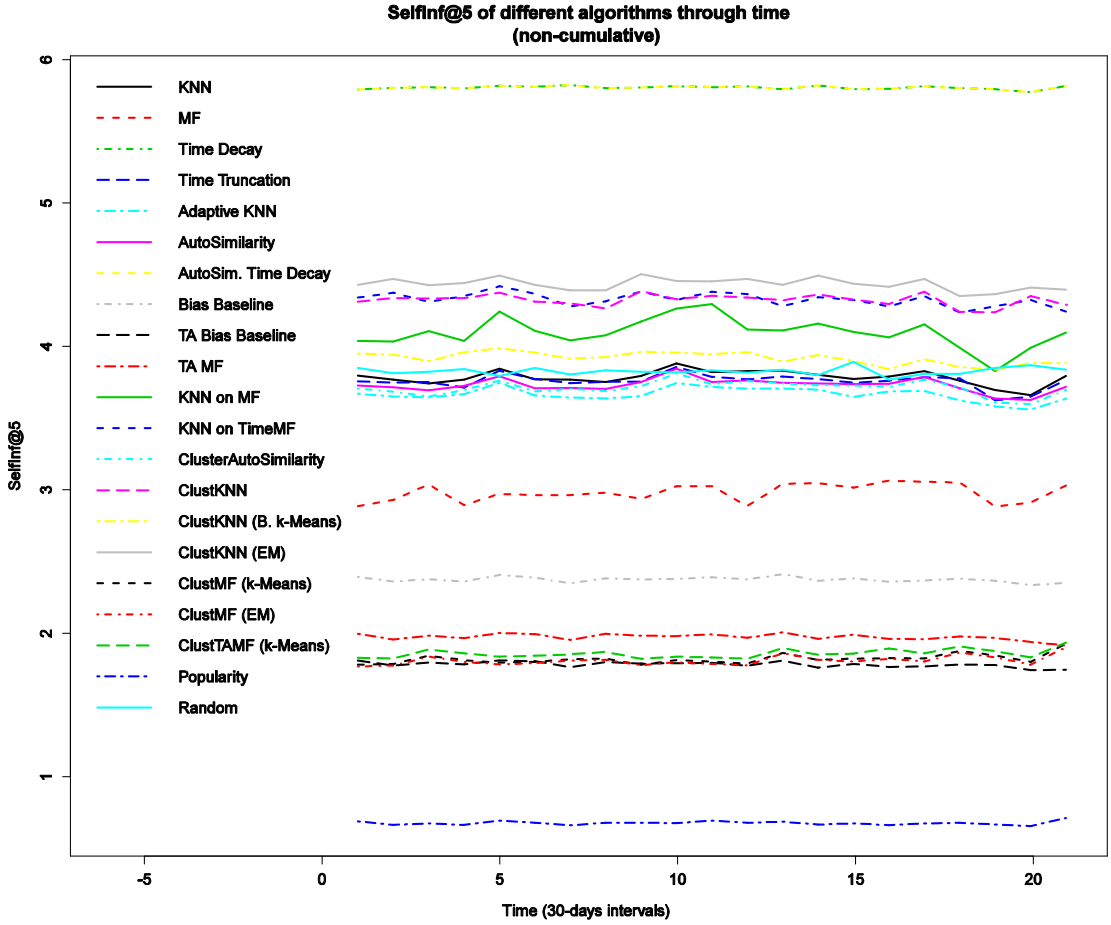


Figure 18. General overview of Self-Information@5 performance through time (non-cumulative data)

Figure 18 through Figure 20 show the results on Self-Information@5 metric. We note that in this case, clearly the Time Decay variants outperform all other algorithms, which indicates their ability to recommend more novel items than other algorithms (s.s.d. w.r.t. kNN on all data splits on both FTI and TI1). On the other hand, MF variants, particularly Time-Aware MF, and the Cluster MF algorithm have a bad performance (s.s.d. between MF and Time-Aware MF models on all data splits on FTI, and on 4 data splits on TI1). Moreover, MF algorithm is outperformed by Random recommender (s.s.d. between MF and Random on all data splits on both TI1 and FTI). Popularity based recommendations have the worst novelty as expected (Self-Information is defined in terms of popularity of items).

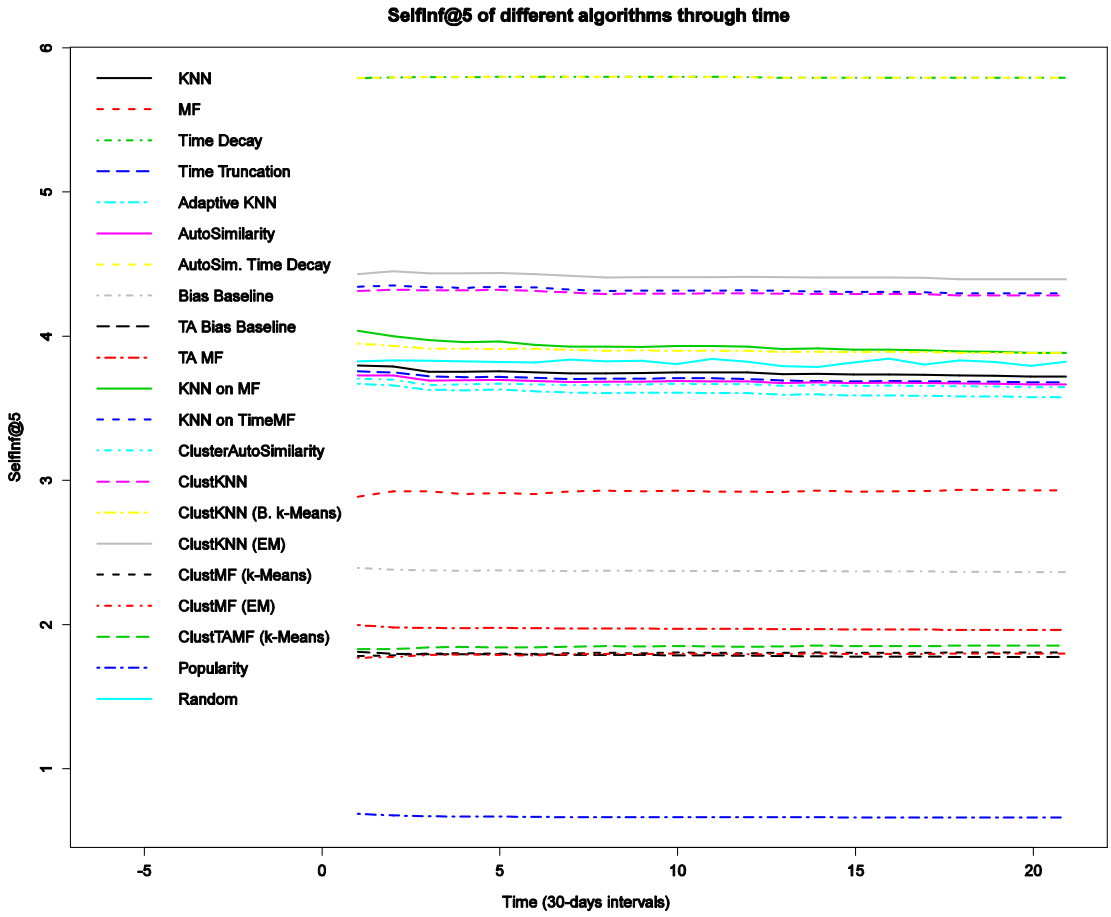
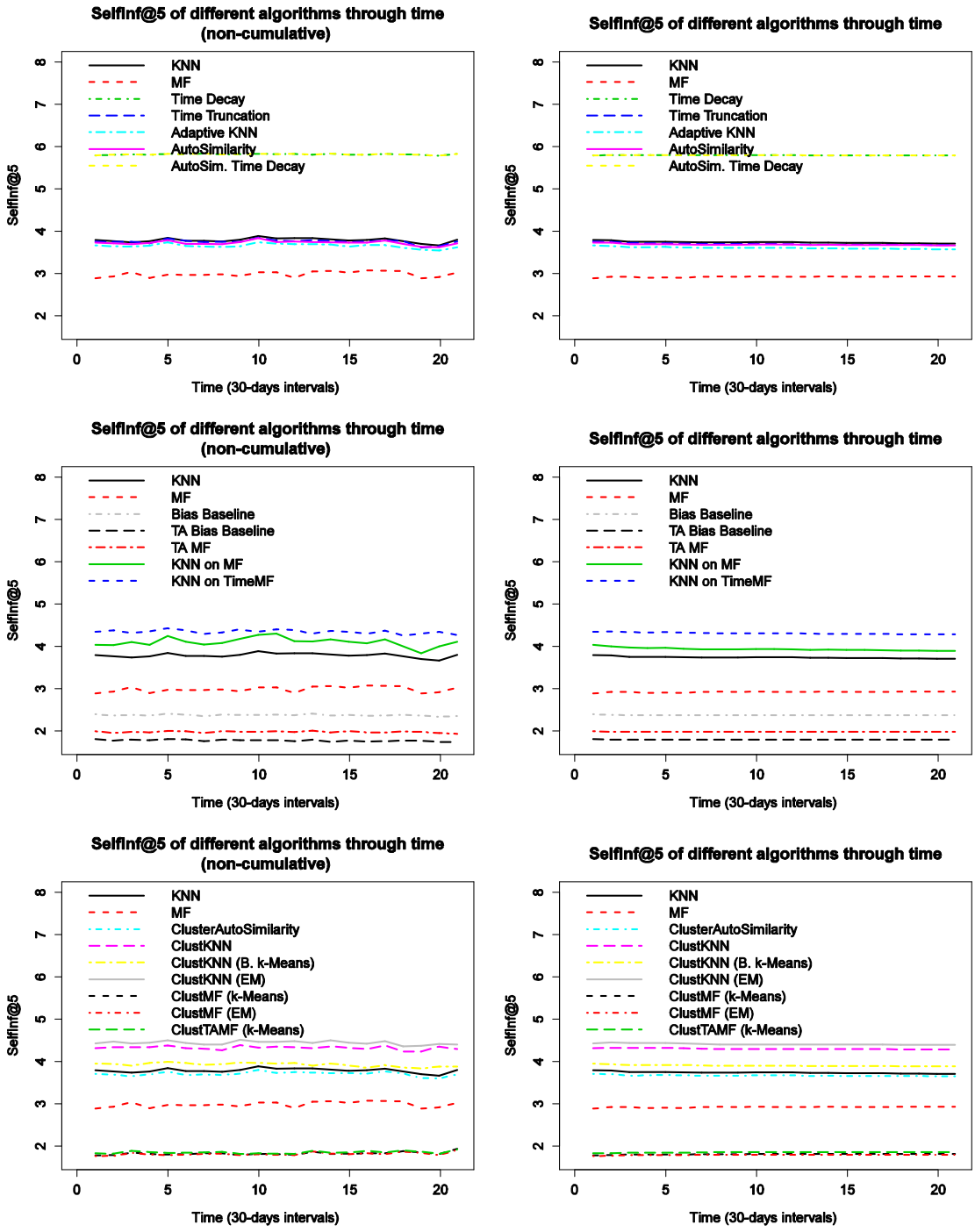


Figure 19. General overview of Self-Information@5 performance through time (cumulative data)

The detailed view allows seeing more clearly that kNN variants outperform the MF model (there are s.s.d. between kNN and MF algorithms on 4 data splits on TI1 and on all data splits on FTI). Without considering Time Decay variants, other Time Aware kNN variants show worse performance than the basic kNN algorithm (there are s.s.d. between kNN and Adaptive-kNN on all splits on both TI1 and FTI; between kNN and AutoSimilarity, there are s.s.d. on 4 data splits on TI1 and on 3 data splits on FTI).

In the case of MF variants, kNN on MF, and particularly kNN on Time Aware MF shows better novelty than MF and kNN baselines (s.s.d. between kNN on Time Aware MF and MF, and between kNN on Time Aware MF and kNN on all data splits on both TI1 and FTI).

With respect to clustering algorithms, Cluster kNN variants are able to outperform kNN (e.g. there are s.s.d. between Cluster kNN using EM clustering algorithm and kNN on all data splits on both TI1 and FTI). On the other hand, algorithms that cluster MF factors show worse performance (there are s.s.d. between MF and Cluster Time Aware MF on 4 data splits on TI1 and on all data splits on FTI).



a) Non-cumulative results. b) Cumulative results.

Figure 20. Detailed view of Self-Information@5 performance through time

Figure 21 through Figure 23 show the Content-Based Intra List Similarity@5 results. In this case and opposite to novelty results, Time Decay algorithm shows the worst performance (here higher values mean more similar items in the list, i.e. less diverse w.r.t. their declared content), even worse than Popularity based recommendations (s.s.d. on 3 data splits on both TI1 and FTI).

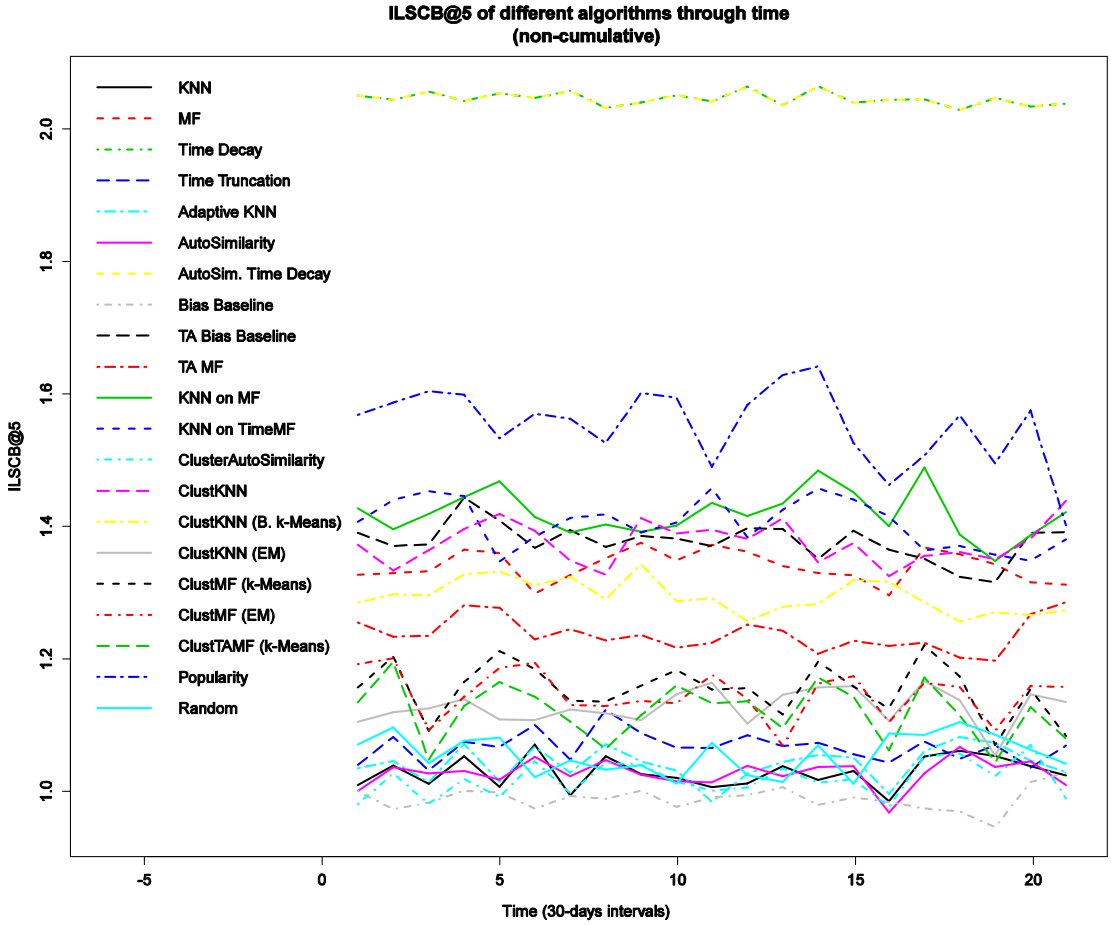


Figure 21. General overview of ILSCB@5 performance through time (non-cumulative data)

The detailed view shows that kNN variants outperform the MF model (there are s.s.d. between MF and kNN on 4 data splits on both TI1 and FTI). It is interesting that AutoSimilarity performs relatively better than kNN, although not statistically significant (s.s.d. on 1 data split on both TI1 and FTI when comparing kNN and AutoSimilarity).

In the case of MF and Bias models, on average Time Aware MF outperforms MF but not statistically significant (s.s.d. on 2 data splits on both TI1 and FTI). Other Time Aware variants have worse performance than their basic counterparts (s.s.d. between Time Aware Bias Baseline and Bias Baseline on all data splits on both TI1 and FTI). kNN has a relatively worse behavior when compared with Bias model on average values, although there are s.s.d. when comparing kNN and Bias Baseline only on 1 data splits on TI1 and on 2 data splits on FTI.

With respect to Clustering algorithms, although it seems that ClusterAutoSimilarity performs somewhat better than kNN, there are s.s.d. only on 1 data split on both TI1 and FTI. Other cluster based algorithms perform worse than kNN (e.g. there are s.s.d. between Cluster Time Aware MF and kNN algorithms on 4 data splits both on TI1 and FTI).

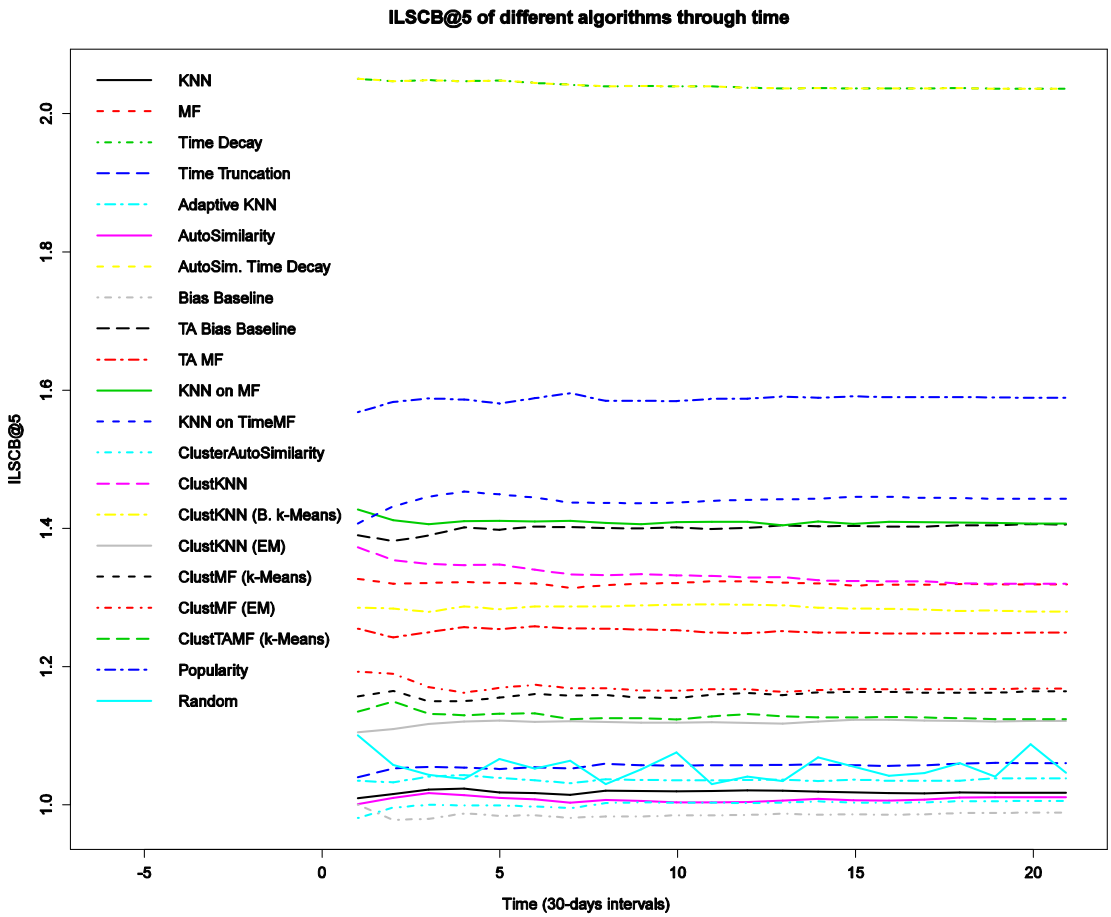
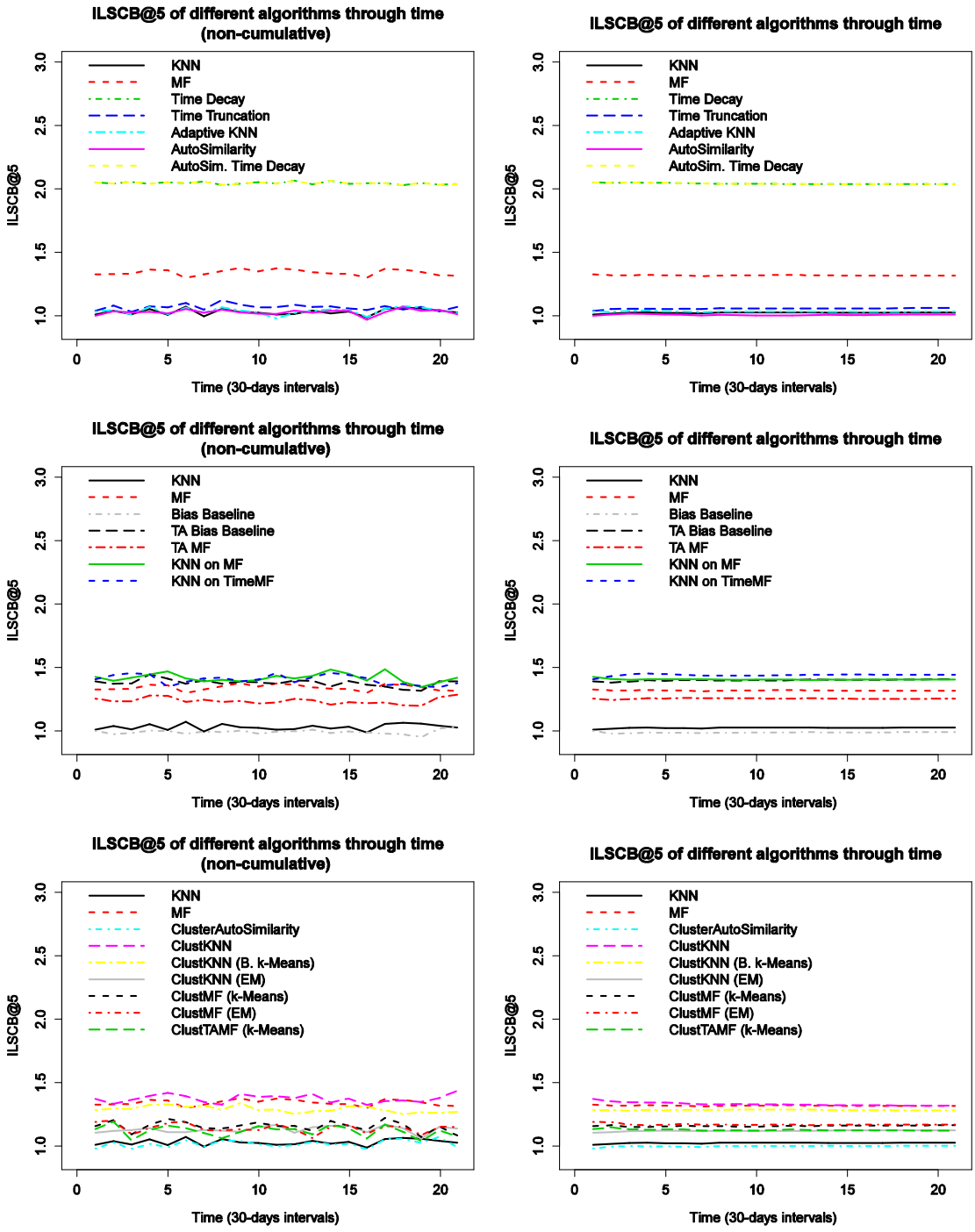


Figure 22. General overview of ILSCB@5 performance through time (cumulative data)



a) Non-cumulative results. b) Cumulative results.

Figure 23. Detailed view of ILSCB@5 performance through time

4.4.4 Other metrics

Among other possible evaluation dimensions, we centered on coverage, as it allows getting an idea of how much of the catalog can be recommended by a RS. In particular, we selected to assess Interest Coverage, as it allows to see how many items on which the user have manifested interest the RS is able to generate predictions.

Figure 24 through Figure 26 show results on the selected metric. We may see that majority of algorithms perform reasonably well, with the exception of the one that post-process MF

results via KNN. This is due to the selected neighbor formation scheme for these variants (similarity threshold instead a fixed number of neighbors), which was used for delivering acceptable accuracy results.

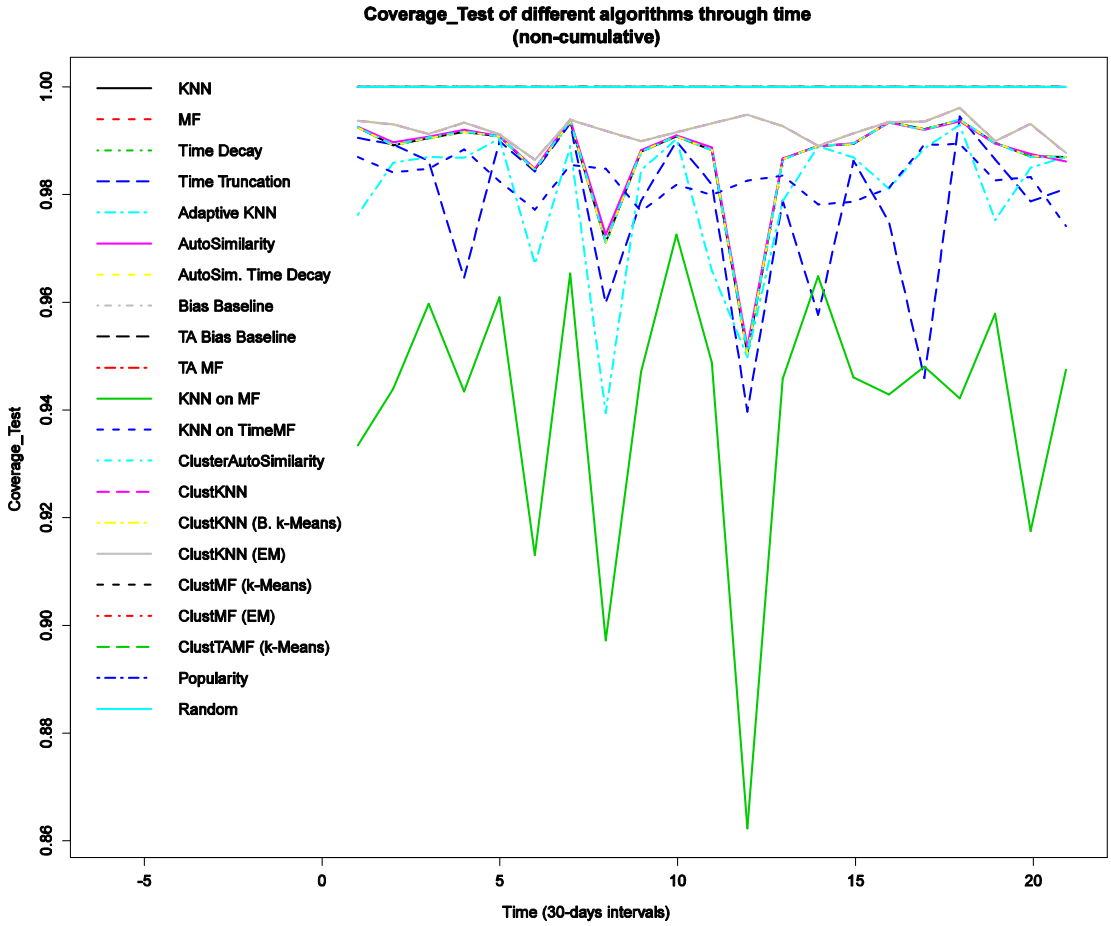


Figure 24. General overview of Interest Coverage performance through time (non-cumulative data)

The detailed view do not show considerable differences on Interest Coverage among the algorithms (except the above mentioned) which implies that most algorithms are to make predictions on almost all interesting items in the catalog. A slight tendency to decrease coverage through time can be seen (particularly con cumulative data), but with low effect (coverage on the last time interval is still over 0.98 for most algorithms).

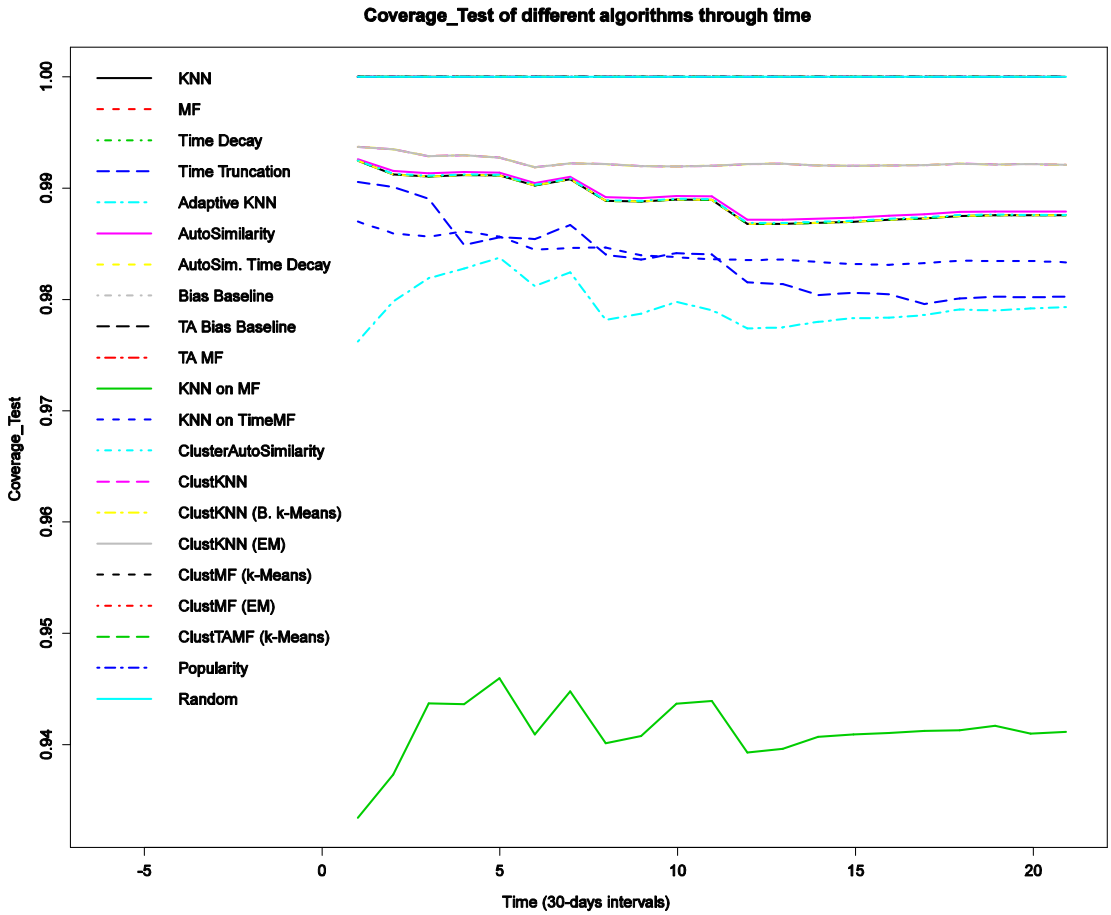
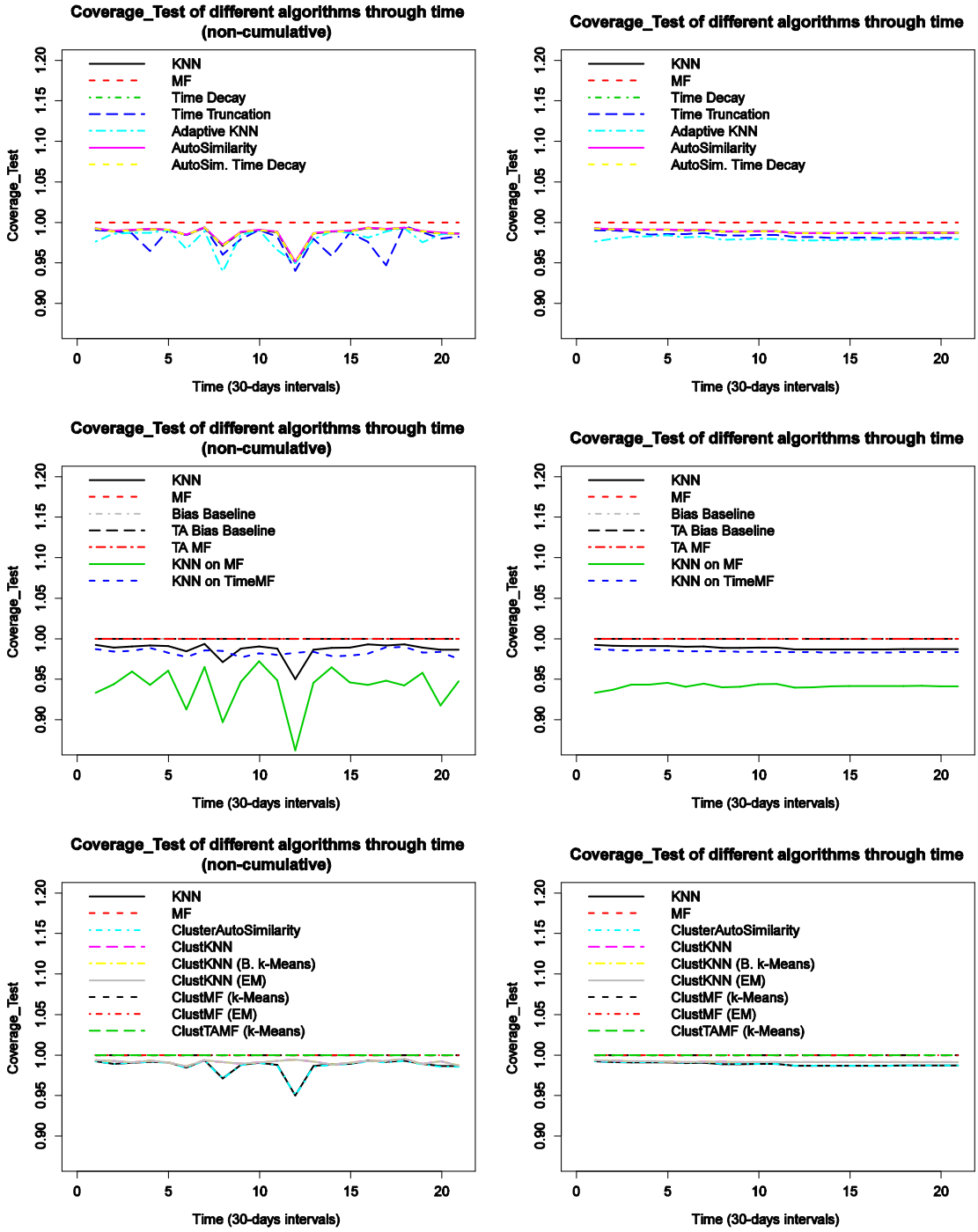


Figure 25. General overview of Interest Coverage performance through time (non-cumulative data)



a) Non-cumulative results. b) Cumulative results.

Figure 26. Detailed view of Interest Coverage performance through time

4.4.5 Concluding Remarks

The results shown provide an important insight about the influence that the incorporation of time information into recommendation models –and also the different base approaches (kNN, MF or Clustering)— have on different evaluation dimensions. As a first key finding, we must remark that time evolution differently affects results, depending on the evaluation dimension, and the recommendation task. Whilst accuracy on rating prediction appears not being penalized by not counting with recent information for generating predictions, in the

case of *top-N* recommendation it seems that time distance between training and recommendation dates does have an impact on results. On the other hand, the usage of models that increasingly fit into data seems to have a downgrading effect on statistical accuracy, which does not seem to occur on decision support accuracy.

It is interesting to note that a simple time-incorporation scheme as Time Decay was practically the only extension that showed improvements upon the base algorithm on rating prediction accuracy, although the Bias Baseline model also showed improvements when enhanced with temporal information. We were unsuccessful on improving the powerful MF model with the additional time information. It seems that in this case, simple is better. These results differ when the task is *top-N* recommendation, particularly assessed on P@5. All of these don't allow us to state that time information effectively helps to improve recommendation performance in terms of results' accuracy on all recommendation tasks. Moreover, the most suitable technique will depend on the particular task at hand. Anyhow, we can state that the MF model do bring cross-task strong results. We also note that, during experimentation, appropriate parameter tuning was crucial for obtaining good accuracy results (particularly for MF variants). This point motivated us to look for and use an auxiliary optimization library.

Regarding the less common evaluation dimensions of novelty and diversity again is not possible to establish what the best performing algorithm on both dimensions is. It is notable the case of the TimeDecay algorithm, which is at the same time the best on Novelty and the worst on Diversity. Moreover, in this case kNN variants bring in general better results than MF variants, oppositely to the observed on accuracy dimensions.

With respect to coverage, the different tested approaches seem to have a minor impact on coverage, having all of them a desirable performance with appropriate parameters selection.

Finally, we must remark in any case that these results are not conclusive, as we used only one particular dataset. Moreover, the different data splits used for testing shows variability on the results, as showed by the statistical test performed, so further research using additional datasets is needed so as to reach to more deciding conclusions. Additionally, we cannot discard that some of these results are biased due to the particular data selection we performed and a comparison using other possible (e.g. less restrictive) data selection scheme is desirable. Likewise, the different approach of generating different time-evolving training sets should also be studied more deeply, as it is more related to real-world tasks, although some practical problems should be adequately faced (e.g. how to make recommendations for novel –without training data—users).

Chapter 5. Conclusions and Future Work

5.1 Conclusions

Throughout this Master Thesis we have developed an exploratory study of different state-of-the-art extensions that incorporate time information into recommendation algorithms, and their results on several evaluation dimensions, under a common experimental setting, on a publicly available dataset. In doing so, we have also compared the performance of different recommendation approaches, such as kNN or Matrix Factorization, bringing this way insight about their impact beyond accuracy.

The main goal of this work was to study state-of-the-art techniques in data mining and machine learning fields that, applied on recommender systems, allow to deal with temporal information. The analyzed methods cover a considerable range of the published approaches on the subject. Moreover, we also tested novel ways of using these approaches.

The main contribution of this work is the assessment of many different techniques over diverse evaluation dimensions and recommendations tasks, under a common and systematic evaluation framework. This attempts to characterize the main benefits and disadvantages of each of them, thus bringing new information about what the techniques more suitable for a particular task and evaluation dimension are.

The specific research goals defined were:

a) To carry out a state-of-the-art revision of techniques used in recommender systems, especially those able to handle temporal information.
b) To apply some of the state-of-the-art data mining techniques for recommendation.
c) To analyze possible benefits derived from the application of clustering, as a traditional data mining technique, on the elaboration of recommendations.
d) To study the impact of techniques on different metrics related with the outcome of recommender systems.

Regarding a), an extensive review of published work on the subject was carried out, including not only time aware techniques, but also the basic techniques upon which time aware extensions are built. Additionally, a review of evaluation metrics available for assessing the different evaluation dimensions was carried out, which allowed us to select the most appropriate metric for each dimension, according to our judge.

A considerable proportion of the studied techniques and metrics were implemented from scratch (so as to have full control of implementation details which could affect recommenders output and metrics computation), and a rigorous evaluation protocol were developed, in order to allow the usage of the different chosen techniques under a common experimental setting, which included two common recommendation tasks, so as to properly fulfill b).

Considering c) we implemented several recommendation algorithms which made use of different clustering techniques, in order to establish benefits derived from their usage on recommendation tasks.

Finally, we assessed the results of recommendations obtained with the different implemented recommendation algorithms, using a unique dataset under a common evaluation protocol, on five different evaluation dimensions (statistical accuracy, decision support accuracy, novelty, diversity and coverage), including six different metrics (RMSE, Precision, AUC, Self-Information, Intra List Similarity and Interest Coverage) as devised on d), obtaining dissimilar results for each technique on these diverse metrics.

At the light of the observed results, differently to what we expected, not all time-aware algorithms were able to outperform their time-unaware counterparts, in particular with respect to accuracy on recommendation prediction (statistical accuracy), which is somewhat unexpected given that in general the main motivation for the elaboration of such extensions is accuracy increase. Over-fitting to training data induced by the usage of additional (time) information appears as a possible explanation for this result. These results appear somewhat opposite to observed behavior on accuracy on *top-N* recommendation task (decision support accuracy). Moreover, these results differ from results in literature, difference that we mainly impute to the different evaluation protocol that we have used. We remark this as a key finding that need to be further analyzed, which motivated us to submit a Poster to the RecSys conference to be held this year. We should note that, at this point, we reckon that the scheme of computing rating predictions in order to build on them a recommendation list is not the best approach, as can be noted when using a non-personalized, popularity based recommendation algorithm, which outperforms all the other tested algorithms.

Clustering methods do not appear as strong methods with respect to accuracy; however, their performance can be considered to be on the average, depending upon the particular data on which clusters are built. Thus, these methods should not be discarded a-priori, furthermore considering that, once trained, they may provide fast recommendation generation, given the dimensional reduction they induce.

With respect to the global evaluation of incorporating time information into recommendation models, at the light of results on the different evaluation dimensions assessed, it was not possible to establish a clear contribution from these models. Our conclusion is that the most suitable recommendation algorithm will depend on the particular recommendation task at hand and evaluation dimension of interest. We also note the importance of proper parameter tuning, which can heavily affect algorithms' results. In any case, we remark that we use a single dataset for testing, thus our results cannot be considered conclusive. Moreover, the different data splits used for testing showed dissimilar results. Regarding this point, replication of this kind of studies on different datasets is devised as a way for reaching towards more deciding conclusions.

From a personal perspective, I would like to remark two important outcomes from the fulfillment of this work. First, a deep understanding of several decision points and tradeoffs of recommendation algorithm development, due to the extensive effort of implement-

ing from scratch those algorithms and related metric computation code. And second, the observation of the need for rigorous and systematic evaluation on scientific disciplines, and in particular on RS field, to which we hope to have contributed with this work.

5.2 Future Work

To the judge of the authors, this work has brought more questions than the answers given (as expectable from an exploratory study). Additionally to the replication of this study on other datasets, among many interesting research lines that can be developed, we identify the following as top priority:

- Development of different recommendation generation schemes for *top-N* prediction task:

As noted in the Conclusions section, generating rating predictions is not the most suitable approach for this task. Other schemes such as relative ordering approaches should be tested, and furthermore, the impact of incorporating time information into them should be assessed.

- Extension of time-aware algorithms making them suitable for other domains, and rating data.

The review of state-of-the-art showed that most work on time-aware algorithms has been performed on movie recommendation domain. However, other domains can also be favored by incorporating time information, e.g. music or educational material recommendation to name a few, which may require completely different approaches to take advantage of temporal information. In fact, almost all proposals have been performed for explicit rating domains, and thus, implicit rating domains remains as an open field of research.

- Novelty and diversity impact.

As the presented results show, performance on these evaluation dimensions is very dissimilar. How to take advantage of temporal information in order to improve results on them appears as an interesting research opportunity, moreover given the increasing attention these dimensions are getting.

- Delivery of "better" recommendations

Given the multiple evaluation dimensions RS currently being evaluated, and the different impacts that a particular technique have on them, how to produce best recommendations is no longer just an accuracy problem. Consequently, novel ways of combining recommendation techniques, and how to decide which particular combination is better than other, arise as a research problem of increasing importance.

Bibliography

Adomavicius, G., Tuzhilin, A. (2001a), *Extending Recommender Systems: A Multidimensional Approach.* In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-01), Workshop on Intelligent Techniques for Web Personalization (ITWP2001). pp. 4-6.

Adomavicius, G., Tuzhilin, A. (2001b), *Multidimensional Recommender Systems: A Data Warehousing Approach.* Proceedings of the Second International Workshop on Electronic Commerce. Springer-Verlag, London, UK, pp. 180-192.

Adomavicius, G., Sankaranarayanan, R., Sen, S., Tuzhilin, A. (2005), *Incorporating Contextual Information in Recommender Systems using a Multidimensional Approach.* ACM Trans.Inf.Syst. **23**(1), 103-145.

Adomavicius, G., Tuzhilin, A. (2005), *Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions.* IEEE Trans.on Knowl.and Data Eng. **17**(6), 734-749.

Adomavicius, G., Tuzhilin, A. (2008), *Context-Aware Recommender Systems.* Proceedings of the 2008 ACM conference on Recommender systems. Lausanne, Switzerland. ACM, New York, NY, USA, pp. 335-336.

Adomavicius, G., Tuzhilin, A. (2011), *Context-Aware Recommender Systems.* In Ricci, F.; Rokach, L.; Shapira, B.and Kantor, P. B. (eds.), Recommender Systems Handbook. Springer US. . ISBN 978-0-387-85820-3.

Amatriain, X., Jaimes, A., Oliver, N., Pujol, J. M. (2011), *Data Mining Methods for Recommender Systems.* In Ricci, F.; Rokach, L.; Shapira, B.and Kantor, P. B. (eds.), Recommender Systems Handbook. Springer US. . ISBN 978-0-387-85820-3.

Ansari, A., Essegaier, S., Kohli, R. (2000), *Internet Recommendations Systems.* J. Marketing Research. **37**(3), 363-375.

Baeza-Yates, R. A., Ribeiro-Neto, B. (1999), *Modern Information Retrieval.* Addison-Wesley Longman Publishing Co., Inc, Boston, MA, USA. . ISBN 020139829X.

Balabanovic, M., Shoham, Y. (1997), *Fab: Content-Based, Collaborative Recommendation.* Commun ACM. **40**(3), 66-72.

Baltrunas, L., Amatriain, X. (2009), *Towards Time-Dependant Recommendation Based on Implicit Feedback.*

Bauer, D. F. (1972), *Constructing Confidence Sets using Rank Statistics.* Journal of the American Statistical Association. **67**(339), 687-690.

Belkin, N. J., Croft, W. B. (1992), *Information Filtering and Information Retrieval: Two Sides of the Same Coin?* Commun ACM. **35**(12), 29-38.

Bell, R. M., Koren, Y. (2007), *Scalable Collaborative Filtering with Jointly Derived Neighborhood Interpolation Weights*. ICDM '07: Proceedings of the 2007 Seventh IEEE International Conference on Data Mining. IEEE Computer Society, Washington, DC, USA, pp. 43-52.

Bell, R. M., Koren, Y., Volinsky, C. (2008), *The Bellkor 2008 Solution to the Netflix Prize*.

Bell, R. M., Bennett, J., Koren, Y., Volinsky, C. (2009), *The Million Dollar Programming Prize*. IEEE Spectr. **46**(5), 28-33.

Bellogín, A., Cantador, I., Castells, P. (2010), *A Study of Heterogeneity in Recommendations for a Social Music Service*. Proceedings of the 1st International Workshop on Information Heterogeneity and Fusion in Recommender Systems. Barcelona, Spain. ACM, New York, NY, USA, pp. 1-8.

Bennet, J., Lanning, S. (2007), *The Netflix Prize*. KDD Cup and Workshop.

Billsus, D., Pazzani, M. J. (1998), *Learning Collaborative Information Filters*. ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning. Morgan Kaufmann Publishers Inc, San Francisco, CA, USA, pp. 46-54.

Breese, J. S., Heckerman, D., Kadie, C. (1998), *Empirical Analysis of Predictive Algorithms for Collaborative Filtering*. Morgan Kaufmann, pp. 43-52.

Brenner, A., Pradel, B., Usunier, N., Gallinari, P. (2010), *Predicting most Rated Items in Weekly Recommendation with Temporal Regression*. Proceedings of the Workshop on Context-Aware Movie Recommendation. Barcelona, Spain. ACM, New York, NY, USA, pp. 24-27.

Burke, R. (2002), *Hybrid Recommender Systems: Survey and Experiments*. User Modeling and User-Adapted Interaction. **12**(4), 331-370.

Cao, H., Chen, E., Yang, J., Xiong, H. (2009), *Enhancing Recommender Systems Under Volatile Userinterest Drifts*. Proceeding of the 18th ACM conference on Information and knowledge management. Hong Kong, China. ACM, New York, NY, USA, pp. 1257-1266.

Celma, O. (2008), *Music Recommendation and Discovery in the Long Tail*. PhD Thesis, Universitat Pompeu Fabra.

Cohen, W. W., Schapire, R. E., Singer, Y. (1999), *Learning to Order Things*. J.Artif.Int.Res. **10**(1), 243-270.

Davies, D. L., Bouldin, D. W. (1979), *A Cluster Separation Measure*. Pattern Analysis and Machine Intelligence, IEEE Transactions on. **PAMI-1**(2), 224-227.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., Harshman, R. (1990), *Indexing by Latent Semantic Analysis*. Journal of the American Society for Information Science. **41**(6), 391.

Dempster, A. P., Laird, N. M., Rubin, D. B. (1977), *Maximum Likelihood from Incomplete Data Via the EM Algorithm*. Journal of the Royal Statistical Society, Series B. **39**(1), 1-38.

Deshpande, M., Karypis, G. (2004), *Item-Based Top-N Recommendation Algorithms.* ACM Trans.Inf.Syst. **22**(1), 143-177.

Ding, Y., Li, X. (2005), *Time Weight Collaborative Filtering.* CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management. Bremen, Germany. ACM, New York, NY, USA, pp. 485-492.

Ding, Y., Li, X., Orlowska, M. E. (2006), *Recency-Based Collaborative Filtering.* ADC '06: Proceedings of the 17th Australasian Database Conference. Hobart, Australia. Australian Computer Society, Inc, Darlinghurst, Australia, Australia, pp. 99-107.

Fouss, F., Saerens, M. *Evaluating Performance of Recommender Systems: An Experimental Comparison.* . ISBN 978-0-7695-3496-1.

Freund, Y., Iyer, R., Schapire, R. E., Singer, Y. (2003), *An Efficient Boosting Algorithm for Combining Preferences.* J.Mach.Learn.Res. **4**933-969.

Fu, C., Silver, D. (2004), *Time-Sensitive Sampling for Spam Filtering.* In Tawfik, A.; and Goodwin, S. (eds.), Advances in Artificial Intelligence. Springer Berlin / Heidelberg.

Gantner, Z., Rendle, S., Schmidt-Thie Lars. (2010), *Factorization Models for Context-/time-Aware Movie Recommendations.* Proceedings of the Workshop on Context-Aware Movie Recommendation. Barcelona, Spain. ACM, New York, NY, USA, pp. 14-19.

Getoor, L., Sahami, M. (1999), *Using Probabilistic Relational Models for Collaborative Filtering.* Working Notes of the KDD Workshop on Web Usage Analysis and User Profiling.

Goldberg, D., Nichols, D., Oki, B. M., Terry, D. (1992), *Using Collaborative Filtering to Weave an Information Tapestry.* Commun ACM. **35**(12), 61-70.

Goldberg, K., Roeder, T., Gupta, D., Perkins, C. (2001), *Eigentaste: A Constant Time Collaborative Filtering Algorithm.* Inf.Retr. **4**(2), 133-151.

Hanley, J. A., McNeil, B. J. (1982), *The Meaning and use of the Area Under a Receiver Operating Characteristic (ROC) Curve.* Radiology. **143**(1), 29-36.

Haveliwala, T. H. (2002), *Topic-Sensitive PageRank.* Proceedings of the 11th international conference on World Wide Web. Honolulu, Hawaii, USA. ACM, New York, NY, USA, pp. 517-526.

Herlocker, J., Konstan, J. A., Riedl, J. (2002), *An Empirical Analysis of Design Choices in Neighborhood-Based Collaborative Filtering Algorithms.* Inf.Retr. **5**(4), 287-310.

Herlocker, J. L., Konstan, J. A., Borchers, A., Riedl, J. (1999), *An Algorithmic Framework for Performing Collaborative Filtering.* SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval. Berkeley, California, United States. ACM, New York, NY, USA, pp. 230-237.

Herlocker, J. L., Konstan, J. A., Terveen, L. G., Riedl, J. T. (2004), *Evaluating Collaborative Filtering Recommender Systems.* ACM Trans.Inf.Syst. **22**(1), 5-53.

Herlocker, J. L. (2000), *Understanding and Improving Automated Collaborative Filtering Systems.* PhD Thesis, University of Minnesota. ISBN 0-599-89612-4.

Hofmann, T. (1999), *Probabilistic Latent Semantic Indexing.* SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval. Berkeley, California, United States. ACM, New York, NY, USA, pp. 50-57.

Hofmann, T. (2001), *Unsupervised Learning by Probabilistic Latent Semantic Analysis.* Mach.Learning. **42**(1), 177-196.

Hofmann, T. (2004), *Latent Semantic Models for Collaborative Filtering.* ACM Trans.Inf.Syst. **22**(1), 89-115.

Jarvelin, K., Kekalainen, J. (2002), *Cumulated Gain-Based Evaluation of IR Techniques.* ACM Trans.Inf.Syst. **20**(4), 422-446.

Jin, R., Si, L., Zhai, C. (2003), *Preference-Based Graphic Models for Collaborative Filtering.* Morgan Kaufmann, San Francisco, CA, pp. 329-336.

Kim, K., Ahn, H. (2008), *A Recommender System using GA K-Means Clustering in an Online Shopping Market.* Expert Syst.Appl. **34**(2), 1200-1209.

Koren, Y., Bell, R. M. (2011), *Advances in Collaborative Filtering.* In Ricci, F.; Rokach, L.; Shapira, B.and Kantor, P. B. (eds.), Recommender Systems Handbook. Springer US. . ISBN 978-0-387-85819-7.

Koren, Y. (2008), *Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model.* KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. Las Vegas, Nevada, USA. ACM, New York, NY, USA, pp. 426-434.

Koren, Y. (2009a), *Collaborative Filtering with Temporal Dynamics.* KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. Paris, France. ACM, New York, NY, USA, pp. 447-456.

Koren, Y. (2009b), *The BellKor Solution to the Netflix Grand Prize.* Report from the Netflix Prize Winners.

Koren, Y., Bell, R., Volinsky, C. (2009), *Matrix Factorization Techniques for Recommender Systems.* Computer. **42**(8), 30-37.

Kumar, R., Raghavan, P., Rajagopalan, S., Tomkins, A. (2001), *Recommendation Systems: A Probabilistic Analysis.* J.Comput.Syst.Sci. **63**(1), 42-61.

Lathauwer, L. D., Moor, B. D., Vandewalle, J. (2000), *A Multilinear Singular Value Decomposition.* SIAM J.Matrix Anal.Appl. **21**(4), 1253-1278.

Lathia, N. (2010), *Evaluating Collaborative Filtering Over Time.* PhD Thesis, Dept. of Computer Science, university College London, UK.

Lathia, N., Hailes, S., Capra, L. (2007), *Private Distributed Collaborative Filtering using Estimated Concordance Measures*. RecSys '07: Proceedings of the 2007 ACM conference on Recommender systems. Minneapolis, MN, USA. ACM, New York, NY, USA, pp. 1-8.

Lathia, N., Hailes, S., Capra, L. (2008), *KNN CF: A Temporal Social Network*. RecSys '08: Proceedings of the 2008 ACM conference on Recommender systems. Lausanne, Switzerland. ACM, New York, NY, USA, pp. 227-234.

Lathia, N., Hailes, S., Capra, L. (2009a), *Temporal Collaborative Filtering with Adaptive Neighbourhoods*. SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval. Boston, MA, USA. ACM, New York, NY, USA, pp. 796-797.

Lathia, N., Hailes, S., Capra, L. (2009b), *Evaluating Collaborative Filtering Over Time*. Proceedings of the SIGIR 2009 Workshop on the Future of IR Evaluation. pp. 41-42.

Lathia, N., Hailes, S., Capra, L., Amatriain, X. (2010), *Temporal Diversity in Recommender Systems*. Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval. Geneva, Switzerland. ACM, New York, NY, USA, pp. 210-217.

Lee, T. Q., Park, Y., Park, Y. (2008), *A Time-Based Approach to Effective Recommender Systems using Implicit Feedback*. Expert Syst.Appl. **34**(4), 3055-3062.

Lee, T. Q., Park, Y., Park, Y. (2009), *An Empirical Study on Effectiveness of Temporal Information as Implicit Ratings*. Expert Syst.Appl. **36**(2), 1315-1321.

Linden, G., Smith, B., York, J. (2003), *Amazon.Com Recommendations: Item-to-Item Collaborative Filtering*. IEEE Internet Comput. **7**(1), 76-80.

Ling, C., Huang, J., Zhang, H. (2003), *AUC: A Better Measure than Accuracy in Comparing Learning Algorithms*. In Xiang, Y.; and Chaib-draa, B. (eds.), Advances in Artificial Intelligence. Springer Berlin / Heidelberg.

Lu, Z., Agarwal, D., Dhillon, I. S. (2009), *A Spatio-Temporal Approach to Collaborative Filtering*. RecSys '09: Proceedings of the third ACM conference on Recommender systems. New York, New York, USA. ACM, New York, NY, USA, pp. 13-20.

Ma, S., Li, X., Ding, Y., Orlowska, M. E. (2007), *A Recommender System with Interest-Drifting*. Proceedings of the 8th international conference on Web information systems engineering. Nancy, France. Springer-Verlag, Berlin, Heidelberg, pp. 633-642.

MacQueen, J. B. (1967), *Some Methods for Classification and Analysis of MultiVariate Observations*. L. M. Le Cam; and J. Neyman (eds.), Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability. University of California Press, pp. 281-297.

Melville, P., Mooney, R. J., Nagarajan, R. (2002), *Content-Boosted Collaborative Filtering for Improved Recommendations*. Eighteenth national conference on Artificial intelligence. Edmonton, Alberta, Canada. American Association for Artificial Intelligence, Menlo Park, CA, USA, pp. 187-192.

Min, S., Han, I. (2005), *Detection of the Customer Time-Variant Pattern for Improving Recommender Systems.* Expert Syst.Appl. **28**(2), 189-199.

Mooney, R. J., Bennett, P. N. (1998), *Book Recommending using Text Categorization with Extracted Information.* In Recommender Systems. Papers from 1998 Workshop. AAAI Press, pp. 49-54.

Mylonas, P., Vallet, D., Castells, P., Fernández, M., Avrithis, Y. (2008), *Personalized Information Retrieval Based on Context and Ontological Knowledge.* Knowl.Eng.Rev. **23**(1), 73-100.

Pan, R., Zhou, Y., Cao, B., Liu, N. N., Lukose, R., Scholz, M., Yang, Q. (2008), *One-Class Collaborative Filtering.* Proceedings of the 2008 Eighth IEEE International Conference on Data Mining. IEEE Computer Society, Washington, DC, USA, pp. 502-511.

Panniello, U., Tuzhilin, A., Gorgoglione, M., Palmisano, C., Pedone, A. (2009), *Experimental Comparison of Pre- Vs. Post-Filtering Approaches in Context-Aware Recommender Systems.* Proceedings of the third ACM conference on Recommender systems. New York, New York, USA. ACM, New York, NY, USA, pp. 265-268.

Paterek, A. (2007), *Improving Regularized Singular Value Decomposition for Collaborative Filtering.* ACM Press.

Pazzani, M., Billsus, D. (1997), *Learning and Revising User Profiles: The Identification of Interesting Web Sites.* Mach.Learn. **27**(3), 313-331.

Pazzani, M. J. (1999), *A Framework for Collaborative, Content-Based and Demographic Filtering.* Artif.Intell.Rev. **13**(5-6), 393-408.

Pennock, D. M., Horvitz, E., Lawrence, S., Giles, C. L. (2000), *Collaborative Filtering by Personality Diagnosis: A Hybrid Memory and Model-Based Approach.* Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence. Morgan Kaufmann Publishers Inc, San Francisco, CA, USA, pp. 473-480.

Rashid, A. M., Lam, S. K., LaPitz, A., Karypis, G., Riedl, J. (2007), *Towards a Scalable kNN CF Algorithm: Exploring Effective Applications of Clustering.* Proceedings of the 8th Knowledge discovery on the web international conference on Advances in web mining and web usage analysis. Philadelphia, PA, USA. Springer-Verlag, Berlin, Heidelberg, pp. 147-166.

Rendle, S., Balby Marinho, L., Nanopoulos, A., Schmidt-Thie Lars. (2009a), *Learning Optimal Ranking with Tensor Factorization for Tag Recommendation.* Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. Paris, France. ACM, New York, NY, USA, pp. 727-736.

Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thie Lars. (2009b), *BPR: Bayesian Personalized Ranking from Implicit Feedback.* Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence. Montreal, Quebec, Canada. AUAI Press, Arlington, Virginia, United States, pp. 452-461.

Rendle, S., Schmidt-Thie Lars. (2010), *Pairwise Interaction Tensor Factorization for Personalized Tag Recommendation.* Proceedings of the third ACM international conference on Web

search and data mining. New York, New York, USA. ACM, New York, NY, USA, pp. 81-90.

Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J. (1994), *GroupLens: An Open Architecture for Collaborative Filtering of Netnews.* Proceedings of the 1994 ACM conference on Computer supported cooperative work. Chapel Hill, North Carolina, United States. ACM, New York, NY, USA, pp. 175-186.

Ricci, F., Quang Nhat Nguyen. (2007), *Acquiring and Revising Preferences in a Critique-Based Mobile Recommender System.* Intelligent Systems, IEEE. **22**(3), 22-29.

Rojas, G., Domínguez, F., Salvatori, S. (2009), *Recommender Systems on the Web: A Model-Driven Approach.* EC-Web 2009: Proceedings of the 10th International Conference on E-Commerce and Web Technologies. Linz, Austria. Springer-Verlag, Berlin, Heidelberg, pp. 252-263.

Santos, O. C., Boticario, J. G. (2008), *Users' Experience with a Recommender System in an Open Source Standard-Based Learning Management System.* USAB '08: Proceedings of the 4th Symposium of the Workgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society on HCI and Usability for Education and Work. Graz, Austria. Springer-Verlag, Berlin, Heidelberg, pp. 185-204.

Sarwar, B., Karypis, G., Konstan, J., Reidl, J. (2001), *Item-Based Collaborative Filtering Recommendation Algorithms.* Proceedings of the 10th international conference on World Wide Web. Hong Kong, Hong Kong. ACM, New York, NY, USA, pp. 285-295.

Sarwar, B. M., Konstan, J. A., Borchers, A., Herlocker, J., Miller, B., Riedl, J. (1998), *Using Filtering Agents to Improve Prediction Quality in the GroupLens Research Collaborative Filtering System.* Proceedings of the 1998 ACM conference on Computer supported cooperative work. Seattle, Washington, United States. ACM, New York, NY, USA, pp. 345-354.

Sarwar, B. M., Karypis, G., Konstan, J. A., Riedl, J. T. (2000), *Application of Dimensionality Reduction in Recommender System - A Case Study.* In ACM WebKDD Workshop.

Scarselli, F., Tsoi, A. C., Hagenbuchner, M. (2004), *Computing Personalized Pageranks.* Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters. New York, NY, USA. ACM, New York, NY, USA, pp. 382-383.

Schafer, J. B., Konstan, J., Riedi, J. (1999), *Recommender Systems in e-Commerce.* Proceedings of the 1st ACM conference on Electronic commerce. Denver, Colorado, United States. ACM, New York, NY, USA, pp. 158-166.

Schein, A. I., Popescul, A., Ungar, L. H., Pennock, D. M. (2002), *Methods and Metrics for Cold-Start Recommendations.* Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval. Tampere, Finland. ACM, New York, NY, USA, pp. 253-260.

Shani, G., Gunawardana, A. (2011), *Evaluating Recommendation Systems.* In Ricci, F.; Rokach, L.; Shapira, B.and Kantor, P. B. (eds.), Recommender Systems Handbook. Springer US. . ISBN 978-0-387-85820-3.

Shardanand, U., Maes, P. (1995), *Social Information Filtering: Algorithms for Automating "Word of Mouth"*. Proceedings of the SIGCHI conference on Human factors in computing systems. Denver, Colorado, United States. ACM Press/Addison-Wesley Publishing Co, New York, NY, USA, pp. 210-217.

Soboroff, I. M., Nicholas, C. K. (1999), *Combining Content and Collaboration in Text Filtering.* Joachims, T. (ed.), Stockholm, Sweden.

Su, X., Khoshgoftaar, T. M. (2009), *A Survey of Collaborative Filtering Techniques.* Adv.in Artif.Intell. **2009**.

Sugiyama, K., Hatano, K., Yoshikawa, M. (2004), *Adaptive Web Search Based on User Profile Constructed without any Effort from Users.* WWW '04: Proceedings of the 13th international conference on World Wide Web. New York, NY, USA. ACM, New York, NY, USA, pp. 675-684.

Swets, J. A. (1969), *Effectiveness of Information Retrieval Methods.* American Documentation. **20**(1), 72-89.

Symeonidis, P., Nanopoulos, A., Manolopoulos, Y. (2008a), *Tag Recommendations Based on Tensor Dimensionality Reduction.* Proceedings of the 2008 ACM conference on Recommender systems. Lausanne, Switzerland. ACM, New York, NY, USA, pp. 43-50.

Symeonidis, P., Nanopoulos, A., Papadopoulos, A. N., Manolopoulos, Y. (2008b), *Collaborative Recommender Systems: Combining Effectiveness and Efficiency.* Expert Syst.Appl. **34**(4), 2995-3013.

Takács, G., Pilászy, I., Németh, B., Tikk, D. (2008), *Investigation of various Matrix Factorization Methods for Large Recommender Systems.* Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition - NETFLIX '08. pp. 1-8.

Tang, T. Y., Winoto, P., Chan, K. C. C. (2003), *On the Temporal Analysis for Improved Hybrid Recommendations.* WI '03: Proceedings of the 2003 IEEE/WIC International Conference on Web Intelligence. IEEE Computer Society, Washington, DC, USA, pp. 214.

Terveen, L., McMackin, J., Amento, B., Hill, W. (2002), *Specifying Preferences Based on User History.* CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems. Minneapolis, Minnesota, USA. ACM, New York, NY, USA, pp. 315-322.

Ungar, L. H., Foster, D. P. (1998), *Clustering Methods for Collaborative Filtering.* AAAI Press, .

Witten, I. H., Frank, E., Hall, M. V. (2005), *Data Mining: Practical Machine Learning Tools and Techniques, Third Edition (the Morgan Kaufmann Series in Data Management Systems).* 2nd edition. ed. Morgan Kaufmann, . ISBN 0123748569.

Xiang, L., Yang, Q. (2009), *Time-Dependent Models in Collaborative Filtering Based Recommender System.* WI-IAT '09: Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology. IEEE Computer Society, Washington, DC, USA, pp. 450-457.

Xiang, L., Yuan, Q., Zhao, S., Chen, L., Zhang, X., Yang, Q., Sun, J. (2010), *Temporal Recommendation on Graphs Via Long- and Short-Term Preference Fusion.* Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining. Washington, DC, USA. ACM, New York, NY, USA, pp. 723-732.

Yang, Y., Padmanabhan, B. (2001), *On Evaluating Online Personalization.* Proceedings of the Workshop on Information Technology and Systems.

Zhang, L., Li, C., Xu, Y., Shi, B. (2005), *An Efficient Solution to Factor Drifting Problem in the pLSA Model.* Proceedings of the The Fifth International Conference on Computer and Information Technology. IEEE Computer Society, Washington, DC, USA, pp. 175-181.

Zhang, M., Hurley, N. (2008), *Avoiding Monotony: Improving the Diversity of Recommendation Lists.* Proceedings of the 2008 ACM conference on Recommender systems. Lausanne, Switzerland. ACM, New York, NY, USA, pp. 123-130.

Zhang, M., Hurley, N. (2009), *Novel Item Recommendation by User Profile Partitioning.* Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 01. IEEE Computer Society, Washington, DC, USA, pp. 508-515.

Zhao, M., Fukushima, T. (2010), *Temporally-Controlled Item Recommendation Method and System Based on Rating Prediction.* 706/12;706/58. United States. G06F15/18; G06N5/02. 2010/08/26.

Zhou, T., Kuscsik, Z., Liu, J., Medo, M., Wakeling, J. R., Zhang, Y. (2010), *Solving the Apparent Diversity-Accuracy Dilemma of Recommender Systems.* Proceedings of the National Academy of Sciences. **107**(10), 4511-4515.

Ziegler, C., McNee, S. M., Konstan, J. A., Lausen, G. (2005), *Improving Recommendation Lists through Topic Diversification.* Proceedings of the 14th international conference on World Wide Web. Chiba, Japan. ACM, New York, NY, USA, pp. 22-32.

Zimdars, A., Chickering, D. M., Meek, C. (2001), *Using Temporal Data for Making Recommendations.* Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence. Morgan Kaufmann Publishers Inc, San Francisco, CA, USA, pp. 580-588.

Annex 1. Numerical results

This annex presents numerical results of the several metric obtained on Test Interval 1 (TI1) and Full Test Interval (FTI). (see section 4.2.2). These results are the averages of each metric over 5 data splits. Other results are available by request to the author (pedro.campos@uam.es).

Algorithm	RMSE		P@5	
	TI1	FTI	TI1	FTI
KNN (Item Based)	0.9136583	0.8943395	0.009524444	0.036
MF	0.9050957	0.8841213	0.02671865	0.084
Time Decay	0.912255	0.8929205	0.0006654526	0.0028
Time Truncation	0.9142166	0.8955481	0.007751293	0.0357
Adaptive KNN	0.9246044	0.9008993	0.01040853	0.0369
AutoSimilarity	0.9135654	0.8946313	0.008855239	0.0364
AutoSimilarity Time Decay	0.9124045	0.8928052	0.0006654526	0.0028
Bias Baseline	0.920211	0.9011059	0.01877876	0.0671
TA Bias Baseline	0.919899	0.8978512	0.02579219	0.0881
Time Aware MF	0.9157953	0.8952391	0.02339875	0.0801
KNN on MF	0.935196	0.9113028	0.009061267	0.0301
KNN on TimeMF	0.954415	0.9410557	0.001318814	0.0116
ClusterAutoSimilarity	0.9139536	0.8945347	0.009504944	0.0383
ClustKNN (k-Means)	0.9407408	0.914537	0.004199791	0.0139
ClustKNN (Bisecting k-Means)	0.9618602	0.9628034	0.004675019	0.0203
ClustKNN (EM)	0.9416885	0.9156121	0.003535675	0.0144
ClustMF (k-Means)	1.269809	1.251704	0.02123358	0.0722
ClustMF (EM)	1.278982	1.260849	0.02253544	0.0734
ClustTimeMF (k-Means)	1.305815	1.302852	0.01942959	0.0676
Popularity	2.63967	2.690642	0.05931003	0.2209
Random	2.64009	2.691833	0.005527406	0.0171

Algorithm	AUC		Self-Information@5	
	TI1	FTI	TI1	FTI
KNN (Item Based)	0.6875297	0.6990058	3.797389	3.721897
MF	0.7048149	0.7143651	2.886243	2.93225
Time Decay	0.6535042	0.6621876	5.789974	5.79098
Time Truncation	0.6864019	0.6960275	3.756424	3.678098
Adaptive KNN	0.6822814	0.693321	3.669079	3.582289
AutoSimilarity	0.6876028	0.6990833	3.727576	3.665136
AutoSimilarity Time Decay	0.6526178	0.6621954	5.789974	5.79098
Bias Baseline	0.6800378	0.6871704	2.391977	2.360986
Time Aware Bias Baseline	0.6904698	0.698419	1.809306	1.771239
Time Aware MF	0.6892061	0.696455	1.996751	1.960706
KNN on MF	0.7065517	0.710401	4.037931	3.88936
KNN on TimeMF	0.6807459	0.6885066	4.341832	4.297828
ClusterAutoSimilarity	0.6875831	0.6989588	3.705999	3.640499
ClustKNN (k-Means)	0.6843489	0.6951586	4.312874	4.281835
ClustKNN (Bisecting k-Means)	0.6842975	0.6924516	3.949744	3.882886
ClustKNN (EM)	0.6849637	0.694853	4.429649	4.393718
ClustMF (k-Means)	0.7763728	0.7935729	1.78232	1.807992
ClustMF (EM)	0.7773855	0.7934578	1.769774	1.796565
ClustTimeMF (k-Means)	0.7714272	0.7906982	1.830442	1.853962
Popularity	0.7757836	0.7960166	0.6881637	0.65901
Random	0.4924091	0.5000232	3.826033	3.824432

Algorithm	ILS_CB@5		Interest Coverage	
	TI1	FTI	TI1	FTI
KNN (Item Based)	1.009498	1.018482	0.9924694	0.987531
MF	1.327015	1.319061	1	1
Time Decay	2.050362	2.036783	0.9924694	0.987531
Time Truncation	1.039887	1.059741	0.9905477	0.9802097
Adaptive KNN	1.034752	1.039145	0.976237	0.9793449
AutoSimilarity	1.000824	1.011263	0.992606	0.987826
AutoSimilarity Time Decay	2.050362	2.036783	0.9924694	0.987531
Bias Baseline	0.9999232	0.9881048	1	1
Time Aware Bias Baseline	1.390166	1.40541	1	1
Time Aware MF	1.254781	1.249305	1	1
KNN on MF	1.427013	1.406609	0.9334269	0.9411179
KNN on TimeMF	1.40673	1.442351	0.9870192	0.9833644
ClusterAutoSimilarity	0.9809189	1.005442	0.9924694	0.987611
ClustKNN (k-Means)	1.372318	1.319268	0.9937091	0.9921372
ClustKNN (Bisecting k-Means)	1.284892	1.279476	0.9937091	0.9921372
ClustKNN (EM)	1.104902	1.12111	0.9937091	0.9921372
ClustMF (k-Means)	1.156812	1.164824	1	1
ClustMF (EM)	1.192315	1.168148	1	1
ClustTimeMF (k-Means)	1.134733	1.123588	1	1
Popularity	1.568129	1.589977	1	1
Random	1.100741	1.046557	1	1

Annex 2. Statistical test results

This annex presents the *p*-values obtained of the application of the Wilcoxon Signed Rank Test [Bauer, 1972] with 95% confidence, on the results of RMSE metric of particular recommendation algorithms, using the *Wilcox.test* routine implemented in R language. In order to obtain such value, after obtaining recommendations with the corresponding recommender, we computed RMSE for each user. That is, we obtained recommendations for a particular user and computed and saved each metric result for that user (and repeated this process for each user). Those per-user values were the input to the Wilcoxon test. With this approach, we computed differences on each data split, and moreover, on each test set (see section 4.2.2).

As we seek to establish what algorithm are the best performing ones, we use a variant of the test that evaluates if one algorithm consistently obtain higher metric values than the other; thus, statistical significance differences presented imply that the first mentioned algorithm statistically have higher metric values than the second one. A *p*-value < 0.05 implies statistical significance (at 95% confidence).

We present results on Test Interval 1 (TI1), corresponding to test data on the first 30 days after training period, and on Full Test Interval (FTI), corresponding to test data on the 624 days after training period (these scheme was similarly applied for computing other metrics' difference statistical significance). Other results are available by request to the author (pedro.campos@uam.es).

RMSE on TI1

Split	kNN vs. TimeDecay	TimeDecay vs. TSAdj.TimeDecay	kNN vs. TSAutoSimilarity	kNN vs MF	TA_MF vs. MF
1	0.125	0.584	0.179	0.269	0.253
2	0.023	0.314	0.472	0.238	0.446
3	0.021	0.864	0.985	0.062	0.097
4	0.168	0.371	0.609	0.001	0.015
5	0.101	0.901	0.462	0.042	0.107

RMSE on FTI

Split	kNN vs. TimeDecay	TimeDecay vs. AutoSimilarity TimeDecay	kNN vs. TSAutoSimilarity	kNN vs MF	TA_MF vs. MF
1	0.026	0.807	0.119	0.018	0.023
2	0.001	0.705	0.665	0.03	0.106
3	0.005	0.956	0.910	0.078	0.327
4	1.59e-05	0.995	0.693	0.007	0.001
5	0.000	0.727	0.620	0.042	0.003

Annex 3. Brief Review of Related Data Mining Techniques

Data Mining and Knowledge Discovery

Witten and Frank [Witten et al., 2005] defines *Data Mining* (DM) as the process of pattern discovery in data. The pattern discovered should be significant, in the sense that they allow obtaining some advantage, e.g. allowing an e-commerce site to increment sales by identifying which items are likely to be bought by customers in order to put them visible. From the previous definition it follows that DM consists in extracting useful and understandable knowledge, previously unknown. This process should be automatic or semi-automatic (assisted) to be effective, and the discovered patterns usage should help to make safer decisions, thus reporting some benefit [Hernández Orallo et al., 2004]. In addition, in [Seifert, 2004] the author states that DM involves the usage of sophisticated data analysis tools to discover new knowledge, from large datasets stored in a variety of formats. These tools include statistical models, mathematical algorithms and machine learning methods[29].

Some common DM tasks are i) identify associations, which are patterns of the form *A is associated to B*, e.g. purchasing beers and diapers[30]; ii) identify sequences (or path analysis), corresponding to patterns of the form *A leads to B*, e.g. get a promotion and buy a new car; iii) classification of patterns, that is to identify coincidences among a new pattern and some predefined patterns, e.g. identify the kind of job of a person given his/her buying behavior knowing beforehand the jobs and buying behavior of a set of persons; iv) clustering of patterns, which aim to identify groups of related patterns without knowing relations among them previously, e.g. group people according to his/her buying behavior; and v) forecasting, whose purpose is to discover patterns which allow to predict some future fact, e.g. how much a person will expend next month given its past buying behavior.

DM is part of a larger process commonly known as *Knowledge Discovery in Databases* (KDD). Fayyad et al. [Fayyad et al., 1996] defines KDD as the non-trivial process of identifying valid, new, potentially useful and understandable patterns from data. This is a complex process, and includes not only the collection of models and patterns (DM goal) but the evaluation and possible interpretation of them, as shown in Figure 27. We stress the importance of the data pre-processing stage, as its output is the input for the DM stage, and the quality of the data will allow (or not) to obtain quality knowledge [Witten et al., 2005; Hernández Orallo et al., 2004].

[29] Machine learning can be conceived as algorithms that improve their performance automatically by means of experience, as neural networks or decision trees.

[30] A common DM example says that retailers found by DM that men who buy diapers also buy beer, though it is not necessarily realistic (http://www.dba-oracle.com/oracle_tips_beer_diapers_data_warehouse.htm).

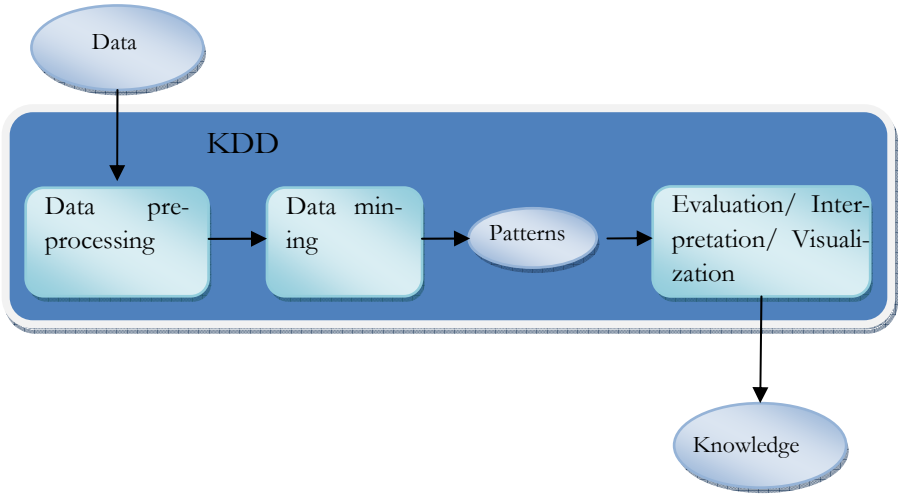


Figure 27. Relation between KDD process and DM (adapted from [Hernández Orallo et al., 2004])

Remainder of this section is mainly devoted to give the reader an overview of the DM work related with RS and temporal data mining. Majority of its content correspond to resumes of specific works which make a broader revision on the RS context [Amatriain et al., 2011], and on temporal data mining [Mitsa, 2010; Laxman and Sastry, 2006]. We point the interested reader to these works to get further details about particular methods and applications.

Data Mining and Recommender Systems

Considering the mentioned in the above section, it is interesting to note that most RS bear in their core an algorithm that can be understood as a particular instance of a DM technique [Amatriain et al., 2011]. More specifically, in this context of recommender applications, the term data mining is used to describe the collection of analysis techniques used to infer recommendation [Schafer, 2009].

Most classical DM techniques have been applied on RS [Amatriain et al., 2011; Schafer, 2009; Adomavicius and Tuzhilin, 2005; Su and Khoshgoftaar, 2009]. For instance, the kNN algorithm (see section 3.1) can be seen as a particular case of an *instance-classification* algorithm [Cover and Hart, 1967], in which utility (ratings) are considered as class labels [Amatriain et al., 2011]. Decision trees [Rokach and Maimon, 2008] have been used in conjunction with association rules [Cho et al., 2002; Nikovski and Kulev, 2006] and Bayesian networks [Breese et al., 1998]. An ontology-based decision tree has also been proposed [Bouza et al., 2008]. Bayesian classifiers have also been used, including Naïve Bayes Classifiers [Ghani and Fano, 2002; Miyahara and Pazzani, 2000; Pronk et al., 2007], Bayesian Networks [Breese et al., 1998] and Hierarchical Bayesian Networks [Yu et al., 2004]. Artificial Neural Networks (ANN) [Zurada, 1992] and Support Vector Machine (SVM) [Cristianini and Shawe-Taylor, 2000] classifiers have been tested as well [Pazzani and Billsus, 1997; Berka et al., 2002; Xia et al., 2006; Xu and Araki, 2006]. Moreover, ANN have been used to combine (or hybridize) the input from several recommendation modules or data sources [Hsu et al., 2007].

Clustering [Hartigan, 1975] is another DM technique that has been applied on RS. The classical k-means algorithm has been used to cluster users [Xue et al., 2005; Sarwar et al., 2002] and items [OConnor and Herlocker, 1999]. Probabilistic models also have been clustered for recommendation [Ungar and Foster, 1998; Li and Kim, 2003]. The main goal behind the usage of clustering is to improve efficiency as computations (after clusters formation) are considerable reduced, at the cost of accuracy, so there must be a consideration of tradeoffs between scalability and prediction performance [Su and Khoshgoftaar, 2009], even though many papers shows a minimum lost in accuracy whereas obtaining great scalability gaining [Rashid et al., 2007; Kim and Ahn, 2008; Braak et al., 2009; Georgiou and Tsapatsoulis, 2010].

According to [Schafer, 2009], one of the best known examples of DM in RS is the discovery of association rules [Witten et al., 2005], which can be used to make recommendation lists essentially by finding association rules between a set of co-viewed items [Sarwar et al., 2000]. It is important to note that the fact that two items are found to be related means co-occurrence but not causality [Amatriain et al., 2011]. Mobasher et al. [Mobasher et al., 2001] present a web personalization system based on association rule mining which outperforms a kNN-based RS both in terms of precision and coverage, based on the idea of multiple minimum supports for rules [Liu et al., 1999]. Lin et al. presented what can be considered as an evolution of this idea, reported as adaptive support rule mining [Lin et al., 2002]. More recently association rules have been proposed to tackle cold-start recommendations through cross-level association rule mining [Leung et al., 2008].

Temporal Data Mining

According to Mitsa [Mitsa, 2010], Temporal DM deals with the harvesting of useful information from temporal data. Laxman and Sastry [Laxman and Sastry, 2006] moreover states that TDM is concerned with data mining of large sequential data sets, meaning by "sequential data" data that are ordered with respect to some index (often, but not necessarily, time), thus being the ordering among the records the important and differentiating aspect. For instance, gene sequences fall into this definition of sequential data. They note that time series constitute a popular class of sequential data, whose analysis has more than fifty years by means of statistical modeling and spectral analysis [Box et al., 1994], whereas TDM has a more recent origin and somewhat different constraints and goals, being one of the main differences the size and nature of data sets (often prohibitively large for conventional time series modeling to handle efficiently). A second major difference described by authors is the kind of information to estimate from the data. Whereas classical time series analysis is intended to forecast or control specific variables, very often in DM applications it is not even known which variables in the data are expected to exhibit any correlations or causal relationships. In fact, exact model parameters (e.g. coefficients of an ARMA model) may be of little interest in DM context compared to unveiling of useful (and often unexpected) trends or patterns in the data. In the context of the present work, we center on data indexed by time, and often we will refer to it simply as time series (TS).

TDM tasks can be broadly classified into [Mitsa, 2010; Laxman and Sastry, 2006]: i) classification, ii) clustering, iii) prediction, and iv) pattern discovery. A central step for many of these tasks is the possibility of computing distance between TS. Mitsa [Mitsa, 2010] distinguishes the following similarity metrics for TS: 1) *Distance-based similarity*, which includes the classical Euclidean distance, Absolute Difference, etc.; 2) *Dynamic Time Warping* [Kruskal and Liberman, 1983; Niels, 2004], a non-linear distance measure that is able to "distort" (warp) the time axis, compressing it at some places and expanding it at others, thus allowing to compare misaligned, yet similar TS (see Figure 28); 3) *Longest Common Subsequence* (LCSS) [Das et al., 1997], a measure tolerant to gaps and able to handle noisy data and outliers better than DTW, and also more scalable [Mitsa, 2010; Vlachos, 2005].

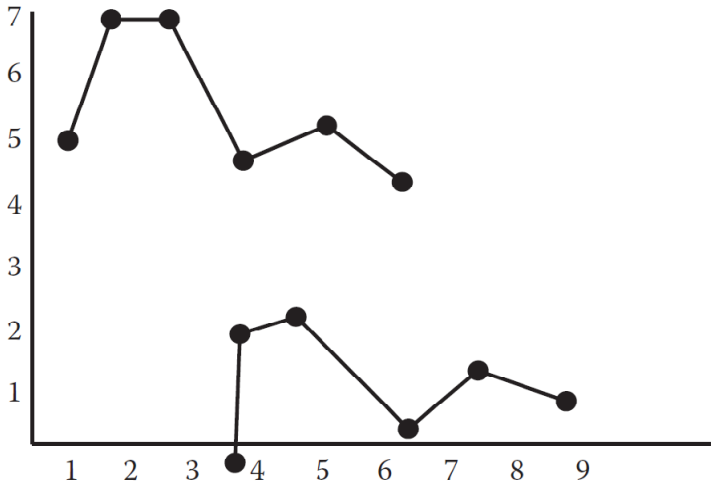


Figure 28. Example of misaligned, yet similar TS (X axis represents time) (from [Mitsa, 2010]).

Additionally, it is important to note that many non-temporal DM techniques for tasks like classification or clustering of TS can be used after the application of one or more temporal representation scheme(s) to the TS data. The goals of such schemes, used mainly to facilitate comparison among TS through similarity analysis, are to reduce dimensionality (in the similarity search problem) and to avoid false dismissals (i.e. if two TS are to be found similar in the original space, they should also be found similar in the transformation space) [Mitsa, 2010]. Some of these transformation schemes are [Mitsa, 2010]: i) Discrete Fourier Transform (DFT) [Agrawal et al., 1993a], which represents a time series $X = \{x_o, x_1, \cdots, x_{n-1}\}$ in the frequency domain by means of the DFT coefficients $F_k = \sum_{i=0}^{n-1} x_i e^{-j2\pi ik/n}$, where $k = 0,1, \cdots, n-1$, with the advantage of the existence of a fast algorithm for its computation known as the Fast Fourier Transform; ii) Discrete Wavelet Transform (DWT) [Chan and Fu, 1999], which utilizes basis functions known as wavelets (functions that allow localization of a TS in both frequency and space), allowing the analysis of a TS at different scales or *resolutions*. Similarly to DFT, wavelets preserve the Euclidean distance after proper normalization[Chan et al., 2003]; iii) Piecewise Aggregate Approximation (PAA) [Keogh et al., 2001], a transformation significantly simpler than DFT and wavelet transform, but whose dimensionality reduction ability rivals with the ability of the before-mentioned techniques; iv) Singular Value Decomposition of TS [Korn et al., 1997], a method based on Singular Value Decomposition which allows data compression;

v) Shape Definition Language (SDL) [Agrawal et al., 1995], a representation that uses a limited vocabulary that describes the gradient in the TS; vi) Landmark-Based Representation [Perng et al., 2000], in which only perceptually important points of a TS are used to represent it; vii) Symbolic Aggregate Approximation (SAX) [Lin et al., 2003], a further discretization of a PAA-represented TS using a string alphabet. Also important are summarization methods, which use global characteristics of a TS to represent it, allowing significant dimensionality reduction. Commonly a feature vector consisting in several summarization methods' output is used to represent (and compare) a TS [Mitsa, 2010]. Some of these methods are [Mitsa, 2010]: i) Basic statistics-based summarization, like Mean, Median, Mode or Variance; ii) Fractal Dimension-based Summarization [Leduc et al., 1994], a method introduced in ecology research of landscapes and related to a estimation of *self-similarity*; iii) Run-Length-based Signature [Uppaluri et al., 2002], based on computations over sequences of consecutive values such as Short Run-Length Emphasis, Long Run-Length Emphasis or Run-Length Non-uniformity; iv) Histogram-based Signature and Statistical Measures, the former being a simple representation of each TS value and value frequency pair, and the latter being statistical computations defined on the histogram such as *skewness*, *kurtosis* or *entropy*; v) Local Trend-based Summarization [Batyrshin et al., 2005], in which linear regressions on TS values are calculated in a moving window scheme, thus obtaining a kind of moving average regression.

As stated by Mitsa, "classification is the task of assigning a new sample to a set of previously known classes, while clustering is the task of grouping samples into clusters of similar samples. This is the reason that classification is known as *supervised learning* while clustering is known as *unsupervised learning*"[Mitsa, 2010]. Most non-temporal classification and clustering algorithms can be applied on a TS representation made with one (or more) of the above-mentioned representation schemes. Nonetheless, specific methods have been developed for TS classification and clustering, as [Mitsa, 2010]: *1-NN TS Classification with Dynamic Time Warping* used as similarity metric [Xi et al., 2006], which is argued as one of the best classification methods for TS. Other classifier proposals include the discretization of TS using an entropy measure [Chen et al., 2007], or variants and improvements for 1-NN TS Classification and DTW computation on classification (see for example [Xi et al., 2006; Ratanamahatana and Keogh, 2004; Wei and Keogh, 2006]). Involving clustering, proposals include a wavelet-based partitional clustering method which allows *anytime clustering*[31] [Lin et al., 2004] and clustering based on Hidden Markov Models (HMM) [Alon et al., 2003]. Other proposals are intended to cluster particular kind of TS, as Auto-Regresive Integrated Moving Average (ARIMA) modeled TS [Kalpakis et al., 2001] or TS data streams [Aggarwal et al., 2003; Guha et al., 2003; Rodrigues et al., 2008].

The prediction task, whose goal has to do with forecasting (typically) future values of the TS based on its past samples [Laxman and Sastry, 2006], can be further split on two related sub-tasks [Mitsa, 2010]: i) *TS prediction* a.k.a. *TS forecasting*, in which the problem is to predict the value of a variable at a multiple of a time interval, and ii) *event prediction*, whose goal is to predict the occurrence (or the number or duration of ocurrences) of an event, given the

[31] An anytime algorithm improves its quality with execution time, allowing to trade execution time for quality of results.

existence of certain conditions. The most simple models for these can be built using *regression*, which examines whether a variable (the dependent variable) can be predicted using one or more variables (the independent variable(s)). Several kinds of regression models can be built, ranging from *simple linear regression* (one independent variable, and the dependent variable being described by a linear curve) to *multiple non-linear regression* (several dependent variables, and the dependent variable described by a non-linear curve; this way, by means of e.g. a simple linear regression, the total duration of *hypoglycemic episodes* can be predicted from the *daily medication dosage* of patients on treatment of diabetes [Mitsa, 2010]). Other simple models that can be used in TS forecasting are [Mitsa, 2010]: *moving averages*, whose idea is to forecast the average of past patterns using a sliding window, *exponential smoothing*, in which is possible to weight differently recent and past observations via the use of a smoothing constant, *random walk*, whose main idea is that the difference between successive observations is random. A more elaborated model is based on *autoregression*, in which the TS forecasting are done via a regression where the independent variables are *lagged values* (i.e. previous values) of the dependent variable. In order to determine the best lagged values to include, the *autocorrelation* formula can be used [Mitsa, 2010]:

$$Autocorrelation_m = \frac{\sum_{i=1}^{N-m}(Y_i - Y_{mean})(Y_{i+m} - Y_{mean})}{\sum_{i=1}^{N}(Y_i - Y_{mean})^2} \quad \textbf{(78)}$$

where $Y_1, Y_2, \cdots, Y_N$ is a series of observations, Y_{mean} is the mean of the observations and m is the lag being computed. The highest correlated lags are then used to compute the autoregression [Mitsa, 2010]:

$$Y(t) = c + B_1 \times Y(t-1) + B_2 \times Y(t-2) + \cdots \quad \textbf{(79)}$$

where the coefficients B_i and c are calculated via regression. Combining autoregression and moving average ideas lead to *AutoRegressive Moving Average* (ARMA) models [Mitsa, 2010; Box et al., 1994]:

$$Y(t) = constant + \sum_{i=1}^{M} a_i Y(t-i) + \sum_{i=1}^{N} b_i E(t-i) + \varepsilon(t) \quad \textbf{(80)}$$

where the first summation is the autoregressive part consisting of weighted past observations, the second summation is the moving average part consisting of weighted past observation errors, and $\varepsilon(t)$ is an error term that is assumed to be a random variable sampled from a normal distribution. ARMA models are designed for stationary TS (i.e. the mean and the variance of the TS do not change over time). *AutoRegressive Integrated Moving Average* (ARIMA) models are ARMA models intended to deal with stationary TS after a transformation via differentiation or logarithms which converts the TS into a stationary one [Mitsa, 2010]. Some other ideas has also been tested, as the usage of ANN [Palit and Popovic, 2005], Genetic Algorithms [Ferreira et al., 2005], or Clustering [Geva, 1999; Sfetsos and Siriopoulos, 2004] for TS forecasting.

Temporal pattern discovery deals with the discovery of temporal patters of interest in TS, or temporal sequences, where the interest is determined by the domain and the application [Mitsa, 2010]. A pattern in this context is a local structure in the data, existing many ways of defining what constitutes a pattern [Laxman and Sastry, 2006]. Two main related sub-task can be identified [Mitsa, 2010; Laxman and Sastry, 2006]: *sequence mining* (or *frequent sequential pattern discovery*), whose purpose is to find frequent *sequences* (in this context a (time) ordered list of *itemsets*, with an itemset consisting of all *items* that appear together in a transaction or session) that exceed a user-specified *support* threshold (where support is the percentage of tuples in a database that contain the sequence) and *frequent episode discovery*, where an *episode* is defined as a sequence of events appearing in a specific order, where an event correspond to time-stamped data. A third related sub-task identified by Mitsa [Mitsa, 2010] is *temporal association rule discovery*, where a *temporal association rule* can be defined as pair $(Rule, T)$ where $Rule$ is an association rule and T is a temporal feature, such as a period or calendar [Mitsa, 2010; Chen et al., 2007]. The firsts algorithms for sequence mining are based on the *Apriori* algorithm [Agrawal et al., 1993b], which iteratively finds frequent itemsets using an efficient candidate generation scheme taking advantage of the fact that any subset of a frequent itemset is also a frequent itemset. Based on this idea, Agrawal and Srikant [Agrawal and Srikant, 1995] extend the frequent itemset idea to the case of items with temporal order. Since its publication, variants and improvements of this algorithm have been proposed, e.g. parallelizing its computation [Shintani and Kitsuregawa, 1996]. Other proposals include Generalized Sequential Patterns (GSP) [Srikant and Agrawal, 1996], which takes advantage of user-specified taxonomies of items, allows the incorporation of time constraints and further has very good scalability; and Sequential PAttern Discovery using Equivalence classes (SPADE) [Zaki, 2001], which is able to discover all sequences in only three database scans. In the case of frequent episode discovery, the proposal of Mannila et al. [Mannila et al., 1997] presents a framework widely used for this task consisting of defining episodes as partially ordered sets of events within specific time windows. Finally, regarding temporal association rule discovery, proposals range from the usage of extension of the Apriori algorithm [Ale and Rossi, 2000], to the use of genetic programming and specialized hardware [Hetland and Saetrom, 2002].

Annex Bibliography

Adomavicius, G., Tuzhilin, A. (2005), *Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions.* IEEE Trans.on Knowl.and Data Eng. **17**(6), 734-749.

Aggarwal, C. C., Han, J., Wang, J., Yu, P. S. (2003), *A Framework for Clustering Evolving Data Streams.* Proceedings of the 29th international conference on Very large data bases - Volume 29. Berlin, Germany. VLDB Endowment, pp. 81-92.

Agrawal, R., Srikant, R. (1995), *Mining Sequential Patterns.* Data Engineering, 1995. Proceedings of the Eleventh International Conference on. pp. 3-14.

Agrawal, R., Faloutsos, C., Swami, A. N. (1993a), *Efficient Similarity Search in Sequence Databases*. Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms. Springer-Verlag, London, UK, pp. 69-84.

Agrawal, R., Imielinski, T., Swami, A. (1993b), *Mining Association Rules between Sets of Items in Large Databases*. Proceedings of the 1993 ACM SIGMOD international conference on Management of data. Washington, D.C., United States. ACM, New York, NY, USA, pp. 207-216.

Agrawal, R., Psaila, G., Wimmers, E. L., Zait, M. (1995), *Querying Shapes of Histories*. Proceedings of the 21th International Conference on Very Large Data Bases. Morgan Kaufmann Publishers Inc, San Francisco, CA, USA, pp. 502-514.

Ale, J. M., Rossi, G. H. (2000), *An Approach to Discovering Temporal Association Rules*. Proceedings of the 2000 ACM symposium on Applied computing - Volume 1. Como, Italy. ACM, New York, NY, USA, pp. 294-300.

Alon, J., Sclaroff, S., Kollios, G., Pavlovic, V. (2003), *Discovering Clusters in Motion Time-Series Data*. Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on. pp. I-375-I-381 vol.1.

Amatriain, X., Jaimes, A., Oliver, N., Pujol, J. M. (2011), *Data Mining Methods for Recommender Systems*. In Ricci, F.; Rokach, L.; Shapira, B.and Kantor, P. B. (eds.), Recommender Systems Handbook. Springer US, . ISBN 978-0-387-85820-3.

Batyrshin, I., Herrera-Avelar, R., Sheremetov, L., Panova, A. (2005), *Association Networks in Time Series Data Mining*. Fuzzy Information Processing Society, 2005. NAFIPS 2005. Annual Meeting of the North American. pp. 754-759.

Berka, T., Behrendt, W., Gams, E., Reich, S. (2002), *Recommending Internet-Domains using Trails and Neural Networks*. Proceedings of the Second International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems. Springer-Verlag, London, UK, UK, pp. 368-371.

Bouza, A., Reif, G., Bernstein, A., Gall, H. (2008), *SemTree: Ontology-Based Decision Tree Algorithm for Recommender Systems*. International Semantic Web Conference (Posters & Demos).

Box, G., Jenkins, G. M., Reinsel, G. (1994), *Time Series Analysis: Forecasting & Control*. 3rd; 3rd ed. Prentice Hall, , February. ISBN 978-0130607744.

Braak, P. t., Abdullah, N., Xu, Y. (2009), *Improving the Performance of Collaborative Filtering Recommender Systems through User Profile Clustering*. Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology - Volume 03. IEEE Computer Society, Washington, DC, USA, pp. 147-150.

Breese, J. S., Heckerman, D., Kadie, C. (1998), *Empirical Analysis of Predictive Algorithms for Collaborative Filtering*. Morgan Kaufmann, pp. 43-52.

Chan, K., Fu, A. W. (1999), *Efficient Time Series Matching by Wavelets*. Data Engineering, 1999. Proceedings., 15th International Conference on. pp. 126-133.

Chan, F. K., Wai-chee Fu, A., Yu, C. (2003), *Haar Wavelets for Efficient Similarity Search of Time-Series: With and without Time Warping.* IEEE Trans.on Knowl.and Data Eng. **15**(3), 686-705.

Chen, X., Ye, D., Hu, X. (2007), *Entropy-Based Symbolic Representation for Time Series Classification.* Proceedings of the Fourth International Conference on Fuzzy Systems and Knowledge Discovery - Volume 02. IEEE Computer Society, Washington, DC, USA, pp. 754-760.

Cho, Y. H., Kim, J. K., Kim, S. H. (2002), *A Personalized Recommender System Based on Web Usage Mining and Decision Tree Induction.* Expert Syst.Appl. **23**(3), 329-342.

Cover, T., Hart, P. (1967), *Nearest Neighbor Pattern Classification.* Information Theory, IEEE Transactions on. **13**(1), 21-27.

Cristianini, N., Shawe-Taylor, J. (2000), *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods.* Cambridge University Press, , March 2000. ISBN 978-0521780193.

Das, G., Gunopulos, D., Mannila, H. (1997), *Finding Similar Time Series.* Proceedings of the First European Symposium on Principles of Data Mining and Knowledge Discovery. Springer-Verlag, London, UK, pp. 88-100.

Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P. (1996), *Advances in Knowledge Discovery and Data Mining.* In Fayyad, U. M.; Piatetsky-Shapiro, G.; Smyth, P.and Uthurusamy, R. (eds.), American Association for Artificial Intelligence, Menlo Park, CA, USA. . ISBN 0-262-56097-6.

Ferreira, T. A. E., Vasconcelos, G. C., Adeodato, P. J. L. (2005), *A New Evolutionary Method for Time Series Forecasting.* Proceedings of the 2005 conference on Genetic and evolutionary computation. Washington DC, USA. ACM, New York, NY, USA, pp. 2221-2222.

Georgiou, O., Tsapatsoulis, N. (2010), *Improving the Scalability of Recommender Systems by Clustering using Genetic Algorithms.* Proceedings of the 20th international conference on Artificial neural networks: Part I. Thessaloniki, Greece. Springer-Verlag, Berlin, Heidelberg, pp. 442-449.

Geva, A. B. (1999), *Non-Stationary Time-Series Prediction using Fuzzy Clustering.* Fuzzy Information Processing Society, 1999. NAFIPS. 18th International Conference of the North American. pp. 413-417.

Ghani, R., Fano, A. (2002), *Building Recommender Systems using a Knowledge Base of Product Semantics.*

Guha, S., Meyerson, A., Mishra, N., Motwani, R., O'Callaghan, L. (2003), *Clustering Data Streams: Theory and Practice.* IEEE Trans.on Knowl.and Data Eng. **15**(3), 515-528.

Hartigan, J. A. (1975), *Clustering Algorithms (Probability & Mathematical Statistics).* John Wiley & Sons Inc, . ISBN 047135645X.

Hernández Orallo, J., Ramírez Quintana, M. J., Ferri Ramírez, C. (2004), *Introducción a La Minería De Datos*. Pearson Educación, S. A., Madrid.

Hetland, M. L., Saetrom, P. (2002), *Temporal Rule Discovery using Genetic Programming and Specialized Hardware*. . Ahmad Lotfi; Jon Garibaldiand Robert John (eds.), Proceedings of the 4th International Conference on Recent Advances in Soft Computing. The Nottingham Trent University, Nottingham, United Kingdom, pp. 182-188.

Hsu, S. H., Wen, M., Lin, H., Lee, C., Lee, C. (2007), *AIMED: A Personalized TV Recommendation System*. Proceedings of the 5th European conference on Interactive TV: a shared experience. Amsterdam, The Netherlands. Springer-Verlag, Berlin, Heidelberg, pp. 166-174.

Kalpakis, K., Gada, D., Puttagunta, V. (2001), *Distance Measures for Effective Clustering of ARIMA Time-Series*. Proceedings of the 2001 IEEE International Conference on Data Mining. IEEE Computer Society, Washington, DC, USA, pp. 273-280.

Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S. (2001), *Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases*. Knowledge and Information Systems. **3**(3), 263-286.

Kim, K., Ahn, H. (2008), *A Recommender System using GA K-Means Clustering in an Online Shopping Market*. Expert Syst.Appl. **34**(2), 1200-1209.

Korn, F., Jagadish, H. V., Faloutsos, C. (1997), *Efficiently Supporting Ad Hoc Queries in Large Datasets of Time Sequences*. Proceedings of the 1997 ACM SIGMOD international conference on Management of data. Tucson, Arizona, United States. ACM, New York, NY, USA, pp. 289-300.

Kruskal, J. B., Liberman, M. (1983), *The Symmetric Time-Warping Problem: From Continuous to Discrete*. In Sankoff, D.; and Kruskal, J. B. (eds.), Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison. Addison-Wesley, Reading, Massachusetts.

Laxman, S., Sastry, P. (2006), *A Survey of Temporal Data Mining*. Sadhana. **31**(2), 173-198.

Leduc, A., Prairie, Y. T., Bergeron, Y. (1994), *Fractal Dimension Estimates of a Fragmented Landscape: Sources of Variability*. Landscape Ecol. **9**(4), 279-286.

Leung, C. W., Chan, S. C., Chung, F. (2008), *An Empirical Study of a Cross-Level Association Rule Mining Approach to Cold-Start Recommendations*. Know.-Based Syst. **21**(7), 515-529.

Li, Q., Kim, B. M. (2003), *Clustering Approach for Hybrid Recommender System*. Proceedings of the 2003 IEEE/WIC International Conference on Web Intelligence. IEEE Computer Society, Washington, DC, USA, pp. 33.

Lin, J., Keogh, E., Lonardi, S., Chiu, B. (2003), *A Symbolic Representation of Time Series, with Implications for Streaming Algorithms*. Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery. San Diego, California. ACM, New York, NY, USA, pp. 2-11.

Lin, J., Vlachos, M., Keogh, E. J., Gunopulos, D. (2004), *Iterative Incremental Clustering of Time Series.* . Bertino, E.; Christodoulakis, S.; Plexousakis, D.; Christophides, V.; Koubarakis, M.; Böhm, K.and Ferrari, E. (eds.), Advances in Database Technology - EDBT 2004, 9th International Conference on Extending Database Technology. Heraklion, Crete, Greece. Springer, pp. 106-122.

Lin, W., Alvarez, S. A., Ruiz, C. (2002), *Efficient Adaptive-Support Association Rule Mining for Recommender Systems.* Data Min.Knowl.Discov. **6**(1), 83-105.

Liu, B., Hsu, W., Ma, Y. (1999), *Mining Association Rules with Multiple Minimum Supports.* Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining. San Diego, California, United States. ACM, New York, NY, USA, pp. 337-341.

Mannila, H., Toivonen, H., Inkeri Verkamo, A. (1997), *Discovery of Frequent Episodes in Event Sequences.* Data Min.Knowl.Discov. **1**(3), 259-289.

Mitsa, T. (2010), *Temporal Data Mining.* Chapman & Hall/CRC, . ISBN 978-1-4200-8976-9.

Miyahara, K., Pazzani, M. J. (2000), *Collaborative Filtering with the Simple Bayesian Classifier.* Proceedings of the 6th Pacific Rim international conference on Artificial intelligence. Melbourne, Australia. Springer-Verlag, Berlin, Heidelberg, pp. 679-689.

Mobasher, B., Dai, H., Luo, T., Nakagawa, M. (2001), *Effective Personalization Based on Association Rule Discovery from Web Usage Data.* Proceedings of the 3rd international workshop on Web information and data management. Atlanta, Georgia, USA. ACM, New York, NY, USA, pp. 9-15.

Niels, R. (2004), *Dynamic Time Warping: An Intuituve Way of Handwriting Recognition?* Nijmegenm, The Netherlands: Radboud University Nijmegen.

Nikovski, D., Kulev, V. (2006), *Induction of Compact Decision Trees for Personalized Recommendation.* Proceedings of the 2006 ACM symposium on Applied computing. Dijon, France. ACM, New York, NY, USA, pp. 575-581.

OConnor, M., Herlocker, J. (1999), *Clustering Items for Collaborative filtering.*

Palit, A. K., Popovic, D. (2005), *Computational Intelligence in Time Series Forecasting: Theory and Engineering Applications (Advances in Industrial Control).* Springer-Verlag New York, Inc, Secaucus, NJ, USA. . ISBN 1852339489.

Pazzani, M., Billsus, D. (1997), *Learning and Revising User Profiles: The Identification of Interesting Web Sites.* Mach.Learn. **27**(3), 313-331.

Perng, C. S., Wang, H., Zhang, S. R., Parker, D. S. (2000), *Landmarks: A New Model for Similarity-Based Pattern Querying in Time Series Databases.* Data Engineering, 2000. Proceedings. 16th International Conference on. pp. 33-42.

Pronk, V., Verhaegh, W., Proidl, A., Tiemann, M. (2007), *Incorporating User Control into Recommender Systems Based on Naive Bayesian Classification.* Proceedings of the 2007 ACM con-

ference on Recommender systems. Minneapolis, MN, USA. ACM, New York, NY, USA, pp. 73-80.

Rashid, A. M., Lam, S. K., LaPitz, A., Karypis, G., Riedl, J. (2007), *Towards a Scalable kNN CF Algorithm: Exploring Effective Applications of Clustering.* Proceedings of the 8th Knowledge discovery on the web international conference on Advances in web mining and web usage analysis. Philadelphia, PA, USA. Springer-Verlag, Berlin, Heidelberg, pp. 147-166.

Ratanamahatana, C., Keogh, E. J. (2004), *Making Time-Series Classification More Accurate using Learned Constraints.* . Berry, M. W.; Dayal, U.; Kamath, C.and Skillicorn, D. B. (eds.), Proceedings of the Fourth SIAM International Conference on Data Mining. Lake Buena Vista, Florida, USA.

Rodrigues, P. P., Gama, J., Pedroso, J. (2008), *Hierarchical Clustering of Time-Series Data Streams.* IEEE Trans.on Knowl.and Data Eng. **20**(5), 615-627.

Rokach, L., Maimon, O. (2008), *Data Mining with Decision Trees: Theory and Applications.* . Bunke, H.; and Wang, P. S. P. (eds.), Series in Machine Perception and Artificial Intelligence. World Scientific Publishing, .

Sarwar, B., Karypis, G., Konstan, J., Riedl, J. (2000), *Analysis of Recommendation Algorithms for e-Commerce.* Proceedings of the 2nd ACM conference on Electronic commerce. Minneapolis, Minnesota, United States. ACM, New York, NY, USA, pp. 158-167.

Sarwar, B. M., Karypis, G., Konstan, J. A., Riedl, J. (2002), *Recommender Systems for Large-Scale E-Commerce: Scalable Neighborhood Formation using Clustering.*

Schafer, J. B. (2009), *The Application of Data-Mining to Recommender Systems.* In Wang, J. (ed.), *Encyclopedia of Data Warehousing and Mining.* 2nd. edition ed. Information Science Publishing, . ISBN 9781605660103.

Seifert, J. W. (2004), *Data Mining: An Overview.*

Sfetsos, A., Siriopoulos, C. (2004), *Time Series Forecasting with a Hybrid Clustering Scheme and Pattern Recognition.* Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on. **34**(3), 399-405.

Shintani, T., Kitsuregawa, M. (1996), *Hash Based Parallel Algorithms for Mining Association Rules.* Parallel and Distributed Information Systems, 1996., Fourth International Conference on. pp. 19-30.

Srikant, R., Agrawal, R. (1996), *Mining Sequential Patterns: Generalizations and Performance Improvements.* Proceedings of the 5th International Conference on Extending Database Technology: Advances in Database Technology. Springer-Verlag, London, UK, pp. 3-17.

Su, X., Khoshgoftaar, T. M. (2009), *A Survey of Collaborative Filtering Techniques.* Adv.in Artif.Intell. **2009**2-2.

Ungar, L. H., Foster, D. P. (1998), *Clustering Methods for Collaborative Filtering.* AAAI Press, .

Uppaluri, R., Mitsa, T., Hoffman, E. A., McLennan, G., Sonka, M. (2002), *Method and Apparatus for Analyzing CT Images to Determine the Presence of Pulmonary Tissue Pathology*. 382/128, 382/131. United States. G06K 9/00. oct 15, 1998.

Vlachos, M. (2005), *A Practical Time Series Tutorial with Matlab*. Porto, Portugal: .

Wei, L., Keogh, E. (2006), *Semi-Supervised Time Series Classification*. Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining. Philadelphia, PA, USA. ACM, New York, NY, USA, pp. 748-753.

Witten, I. H., Frank, E., Hall, M. V. (2005), *Data Mining: Practical Machine Learning Tools and Techniques, Third Edition (the Morgan Kaufmann Series in Data Management Systems)*. 2nd edition. ed. Morgan Kaufmann, . ISBN 0123748569.

Xi, X., Keogh, E., Shelton, C., Wei, L., Ratanamahatana, C. A. (2006), *Fast Time Series Classification using Numerosity Reduction*. Proceedings of the 23rd international conference on Machine learning. Pittsburgh, Pennsylvania. ACM, New York, NY, USA, pp. 1033-1040.

Xia, Z., Dong, Y., Xing, G. (2006), *Support Vector Machines for Collaborative Filtering*. Proceedings of the 44th annual Southeast regional conference. Melbourne, Florida. ACM, New York, NY, USA, pp. 169-174.

Xu, J. A., Araki, K. (2006), *A SVM-Based Personal Recommendation System for TV Programs*. Multi-Media Modelling Conference Proceedings, 2006 12th International. pp. 4 pp.

Xue, G., Lin, C., Yang, Q., Xi, W., Zeng, H., Yu, Y., Chen, Z. (2005), *Scalable Collaborative Filtering using Cluster-Based Smoothing*. Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval. Salvador, Brazil. ACM, New York, NY, USA, pp. 114-121.

Yu, K., Tresp, V., Yu, S. (2004), *A Nonparametric Hierarchical Bayesian Framework for Information Filtering*. Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval. Sheffield, United Kingdom. ACM, New York, NY, USA, pp. 353-360.

Zaki, M. J. (2001), *SPADE: An Efficient Algorithm for Mining Frequent Sequences*. Mach.Learn. **42**(1-2), 31-60.

Zurada, J. (1992), *Introduction to Artificial Neural Systems*. West Publishing Co, St. Paul, MN, USA. . ISBN 0-314-93391-3.