A Framework for Time-aware Recommendations

Kostas Stefanidis[1], Irene Ntoutsi[2], Kjetil Nørvåg[1], and Hans-Peter Kriegel[2]

[1] Department of Computer and Information Science, Norwegian University of Science and Technology, Trondheim, Norway
{kstef,kjetil.norvag}@idi.ntnu.no
[2] Institute for Informatics, Ludwig Maximilian University, Munich, Germany
{ntoutsi, kriegel}@dbs.ifi.lmu.de

Abstract. Recently, recommendation systems have received significant attention. However, most existing approaches focus on recommending items of potential interest to users, without taking into consideration how temporal information influences the recommendations. In this paper, we argue that time-aware recommendations need to be pushed in the foreground. We introduce an extensive model for time-aware recommendations from two perspectives. From a *fresh-based* perspective, we propose using a suite of aging schemes towards making recommendations mostly depend on fresh and novel user preferences. From a *context-based* perspective, we focus on providing different suggestions under different temporal specifications. The proposed strategies are experimentally evaluated using real movies ratings.

1 Introduction

Recommendation systems provide users with suggestions about products, movies, videos, pictures and many other items. Many systems, such as Amazon, NetFlix and MovieLens, have become very popular nowadays. One popular category of recommendation systems are the collaborative filtering systems (e.g., [11, 8]) that try to predict the utility of items for a particular user based on the items previously rated by similar users. That is, users similar to a target user are first identified, and then, items are recommended based on the preferences of these users. Users are considered as similar if they buy common items as in case of Amazon or if they have similar evaluations as in case of MovieLens.

The two typical types of entities that are dealt in recommendation systems, i.e., users and items, are represented as sets of ratings, preferences or features. Assume, for example, a restaurant recommendation application (e.g., ZAGAT.com). Users initially rate a subset of restaurants that they have already visited. Ratings are expressed in the form of preference scores. Then, a recommendation engine estimates preference scores for the items, i.e., restaurants in this case, that are not rated by a user and offers him/her appropriate recommendations. Once the unknown scores are computed, the k items with the highest scores are recommended to the user.

Although there is a substantial amount of research performed in the area of recommendation systems, most of the approaches produce recommendations by

ignoring the temporal information that is inherent in the ratings, since ratings are given at a specific point in time. Due to the fact that a huge amount of user preferences data is accumulated over time, it is reasonable to exploit the temporal information associated with these data in order to obtain more accurate and up to date recommendations. In this work, our goal is to use the time information of the user ratings towards improving the predictions in collaborative recommendation systems. We consider two different types of time effects based upon the recency/freshness and the temporal context of the ratings and consequently, we propose two different time-aware recommendation models.

The *fresh-based recommendations* assume that the most recent user preferences better reflect the current trends and thus, they contribute more in the computation of the recommendations. To account for the recency of the ratings we distinguish between the *damped window model* that gradually decreases the importance of ratings over time and the *sliding window model* that counts only for the most recent data and ignores any previous historical information. For example, consider a movie recommendation system that gives higher priority to new releases compared to other old seasoned movies (damped window model) or focuses only on new releases (sliding window model).

From a different perspective, *context-based recommendations* offer different suggestions for different time specifications. The main motivation here, is that user preferences may change over time but have temporal repetition, i.e., recur over time. As an example consider a tourist guide system that should provide different suggestions during summer than during winter. Or, a restaurant recommendation system that might distinguish between weekdays (typically business lunches) and weekends (typically family lunches).

It is the purpose of this paper to provide a framework for studying various approaches that handle different temporal aspects of recommendations. To deal with the sparsity of the explicitly defined user preferences, we introduce the notion of support in recommendations to model how confident the recommendations of an item for a user is. We also consider different cases for selecting the appropriate set of users for producing the recommendations of a user.

The rest of the paper is organized as follows. A general, time-invariant recommendations model is presented in Sect. 2. Time is introduced in Sect. 3, where we distinguish between the aging factor (Sect. 3.1) and the temporal context factor (Sect. 3.2). The computation of recommendations under different temporal semantics is discussed in Sect. 4, while in Sect. 5, we present our experiments using a real dataset of movie ratings. Related work is presented in Sect. 6. Finally, conclusions and outlook are pointed out in Sect. 7.

2 A Time-free Recommendation Model

Assume a set of items $\mathcal{I}$ and a set of users $\mathcal{U}$ interacting with a recommendation application $\mathcal{A}$. Each user $u \in \mathcal{U}$ may express a preference for an item $i \in \mathcal{I}$, which is denoted by $preference(u, i)$ and lies in the range $[0.0, 1.0]$. We use $\mathcal{Z}_i$ to denote the set of users in $\mathcal{U}$ that have expressed a preference for item i. The cardinality of the items set $\mathcal{I}$ is usually high and typically users rate only a few

of these items, that is, $|\mathcal{Z}_i| << |\mathcal{U}|$ for a specific item i. For the items unrated by the users, a *relevance score* is estimated by invoking a recommendation strategy.

In this section, we first present a model for time-free recommendations (Sect. 2.1) and then define the top-k recommendations (Sect. 2.2). The time–free recommendations model is the generally used recommendations model where the notion of time is completely ignored.

2.1 Defining Time-free Recommendations

There are different ways to estimate the relevance of an item for a user. In general, the recommendation methods are organized into three main categories: (i) *content-based*, that recommend items similar to those the user has preferred in the past (e.g., [17, 13]), (ii) *collaborative filtering*, that recommend items that similar users have liked in the past (e.g., [11, 8]) and (iii) *hybrid*, that combine content-based and collaborative ones (e.g., [5]).

Our work falls into the *collaborative filtering* category. The key concept of collaborative filtering is to use, for a given user $u \in \mathcal{U}$, the preferences of other users in $\mathcal{U}$ in order to produce relevance scores for the items unrated by u. But, *which is the appropriate set of users, hereafter called peers, for computing the recommendations of u?* Due to the inherent fuzziness associated with this question, there exists no single definition for locating the peers of u. In our model, we assimilate three different aspects of peers: (i) *close friends*, (ii) *domain experts* and (iii) *similar users*.

The *close friends* of a user u are explicitly selected by u. Computing recommendations using only close friends is based on the assumption that these users would have similar tastes for most things, because of the closeness of relationship.

Close Friends: Let $\mathcal{U}$ be a set of users. The *close friends* $\mathcal{C}_u$, $\mathcal{C}_u \subseteq \mathcal{U}$, of a user $u \in \mathcal{U}$ are explicitly defined by u.

From a different perspective, *domain experts* can be used for producing recommendations for specific queries, since they are considered to be knowledgeable on a specific topic or domain. Several methods deal with the problem of finding experts (e.g., [6]); the focus of this paper though is on how to exploit experts preferences to recommend interesting items to other users and not on how to identify these experts. So, we consider that the set of experts for a given query are predefined, e.g., experts in tablet pcs.

Domain Experts: Let $\mathcal{U}$ be a set of users and Q be a query. The *domain experts* $\mathcal{D}_Q$, $\mathcal{D}_Q \subseteq \mathcal{U}$, are the users considered as experts for the domain of Q.

We denote this set as $\mathcal{D}_Q$, so, not dependent on the user, since typically experts are associated with specific queries, subjects or domains rather than with certain users.

Alternatively, a user can opt to employ the preferences of the users that exhibit the most similar behavior to him/her in order to produce relevance scores for the items unrated by him/her, even if other friendship or expert relationships exist. *Similar users* are located via a *similarity function* $simU(u, u')$ that evaluates the proximity between u and u'. Several methods can be applied for

selecting the similar users of a user u. A direct method is to locate those users u' with similarity $simU(u, u')$ greater than or equal to a threshold value.

SIMILAR USERS: Let $\mathcal{U}$ be a set of users. The *similar users* $\mathcal{S}_u$, $\mathcal{S}_u \subseteq \mathcal{U}$, of a user $u \in \mathcal{U}$ is a set of users, such that, $\forall u' \in \mathcal{S}_u$, $simU(u, u') \geq \delta$ and $\forall u'' \in \mathcal{U} \backslash \mathcal{S}_u$, $simU(u, u'') < \delta$, where δ is a threshold similarity value.

Clearly, one could argue for other ways of selecting $\mathcal{S}_u$, e.g., by taking the m most similar users to u. Our main motivation here is that we opt for selecting only highly connected users even if the resulting set of users $\mathcal{S}_u$ is small.

We define now the general notion of peers for a user by taking into account the three different cases.

Definition 1 (Peers). *Let $\mathcal{U}$ be a set of users, u be a user in $\mathcal{U}$ and Q be a query posed by u. The peers $\mathcal{P}_{u,Q}$, $\mathcal{P}_{u,Q} \subseteq \mathcal{U}$, of u for Q are either: (i) the close friends $\mathcal{C}_u$ of u, (ii) the domain experts $\mathcal{D}_Q$ for Q, or (iii) the similar users $\mathcal{S}_u$ of u.*

Based on the peers of a user for a query, we define formally the relevance of an item for a user as follows:

Definition 2 (Time-free Relevance). *Let u be a user in $\mathcal{U}$, Q be a query posed by u and $\mathcal{P}_{u,Q}$ be the peers of u for Q. If u has not expressed any preference for an item i, the time-free relevance of i for u under Q is:*

$$relevance^f(u, i, Q) = \frac{\sum_{u' \in (\mathcal{P}_{u,Q} \cap \mathcal{Z}_i)} contribution(u, u') \times preference(u', i)}{\sum_{u' \in (\mathcal{P}_{u,Q} \cap \mathcal{Z}_i)} contribution(u, u')}$$

$$where\ contribution(u, u') = \begin{cases} 1, & if\ \mathcal{P}_{u,Q}\ is\ \mathcal{C}_u\ or\ \mathcal{D}_Q \\ simU(u, u'), & if\ \mathcal{P}_{u,Q}\ is\ \mathcal{S}_u \end{cases}$$

The relevance score of user u for an item i depends on the peers of u that have given a rating for i, i.e., those in $\mathcal{P}_{u,Q} \cap \mathcal{Z}_i$. The $contribution(u, u')$ reflects the importance of each $preference(u', i)$ for u; this importance depends on how "reliable" u' is for u. When close friends or domain experts are used, contribution is set to 1, since we are certain about the importance of the preferences of the selected users. For the similar users case, the contribution of each user u' depends on the similarity between u and u'.

As already mentioned, due to the abundance of items in a recommendation application, users, even the expert ones, rate only a small portion of them. So, the following question usually arises: *How confident are the relevance scores associated with the recommended items?* To deal with this problem, we introduce the notion of *support* for each candidate item i for user u, which defines the fraction of peers of u that have expressed preferences for i.

Definition 3 (Time-free Support). *Let u be a user in $\mathcal{U}$, Q be a query posed by u and $\mathcal{P}_{u,Q}$ be the peers of u for Q. The time-free support of i for u under Q is:*

$$support^f(u, i, Q) = |\mathcal{P}_{u,Q} \bigcap \mathcal{Z}_i| / |\mathcal{P}_{u,Q}|$$

Intuitively, the notion of support expresses how reliable is our estimation for i.

To estimate the worthiness of an item recommendation for a user, we propose to combine the *relevance* and *support* scores in terms of a *value* function.

Definition 4 (Time-free Value). *Let $\mathcal{U}$ be a set of users and $\mathcal{I}$ be a set of items. For $\sigma \in [0,1]$, the time-free value of an item $i \in \mathcal{I}$ for a user $u \in \mathcal{U}$ under a query Q, such that, $\nexists preference(u,i)$, is:*

$$value^f(u,i,Q) = \sigma \times relevance^f(u,i,Q) + (1-\sigma) \times support^f(u,i,Q)$$

We take a generic approach for computing the *time-free value* of an item for a user. More sophisticated functions can be designed. However, this general form of weighted summation is simple and easy to implement. Moreover, when $\sigma = 1$, *value* maps to *relevance*, which is the typically used recommendation score.

2.2 Top-k Time-free Recommendations

Given a query Q submitted by a user u and a restriction k on the number of the recommended items, the goal is to provide u with k suggestions for items that are highly relevant to u and exhibit high support.

Definition 5 (Top-k Time-free Recommendations). *Let $\mathcal{U}$ be a set of users and $\mathcal{I}$ be a set of items. Given a query Q posed by a user $u \in \mathcal{U}$, recommend to u a list of k items $\mathcal{I}_u =< i_1, \ldots, i_k >$, $\mathcal{I}_u \subseteq \mathcal{I}$, such that:*

(i) $\forall i_j \in \mathcal{I}_u$, $\nexists preference(u, i_j)$,
(ii) $value^f(u, i_j, Q) \geq value^f(u, i_{j+1}, Q)$, $1 \leq j \leq k-1$, $\forall i_j \in \mathcal{I}_u$, *and*
(iii) $value^f(u, i_j, Q) \geq value^f(u, x_y, Q)$, $\forall i_j \in \mathcal{I}_u$, $x_y \in \mathcal{I} \backslash \mathcal{I}_u$.

The first condition ensures that the suggested items do not include already evaluated items by the user (for example, do not recommend a movie that the user has already watched). The second condition ensures the descending ordering of the items with respect to their value, while the third condition defines that every item in the result set has value greater than or equal to the value of any of the non–suggested items.

3 Time-aware Recommendations

The general time-free recommendation model assumes that all preferences are equally active and potentially they can be used for producing recommendations. This way though the temporal aspects of the user ratings are completely ignored. However, the information needs of a user evolve over time, especially if we consider a long period of time, either smoothly (i.e., drift) or more drastically (i.e., shift). This fact makes the recent user preferences to reflect better the current trends than the older preferences do. From a different point of view, user interests differ from time to time which means that users may have different needs under different temporal circumstances. For example, during the weekdays one might be interested in reading IT news whereas during the weekends he/she might be interested in reading about cooking, gardening or doing other hobbies.

To handle such different cases, we propose a framework for time-aware recommendations that incorporates the notion of time in the recommendations process with the goal of improving their accuracy. We distinguish between two types of time-aware recommendations, namely the fresh-based and the context-based ones. The *fresh-based recommendations* pay more attention to more recent user ratings thus trying to deal with the problems of drift or shift in the user

information needs over time. The *context-based recommendations* take into account the temporal context under which the ratings were given (e.g., weekdays, weekends).

In our time-aware recommendation model, the rating of a user u for an item i, i.e., $preference(u, i)$, is associated with a timestamp $t_{u,i}$, which is the time that i was rated by u. So, this timestamp declares the freshness or age of the rating. Below, we first define the fresh-based recommendation model (Sect. 3.1) and then, the temporal context-based recommendation model (Sect. 3.2). We also present a variant of the top-k recommendations problem by defining the top-k time-aware recommendations (Sect. 3.3).

3.1 Fresh-based Recommendations

Generally speaking, the popularity of the items of a recommendation application changes with time; typically, items lose popularity while time goes on. For example, a movie, a picture or a song may lose popularity because they are too seasoned. Motivated by the intuition that the importance of preferences for items increases with the popularity of the items themselves, fresh-based recommendation approaches care for suggesting items taking mainly into account recent and novel user preferences.

Driven by the work in stream mining [10], we use different types of aging mechanisms to define the way that the historical information (in form of ratings) is incorporated in the recommendation process. Aging in streams is typically implemented through the notion of windows, which define which part of the stream is active at each time point and thus could be used for the computations. In this work, we use the *damped window model* that gradually decreases the importance of historical data comparing to more recent data and the *sliding window model* that remembers only the preferences defined within a specific, recent time period. We present these cases in more detail below. Note that the static case (Sect. 2), corresponds to the *landmark window model* where the whole history from a given landmark is considered.

Damped window model. In the damped window model, although all user preferences are active, i.e., they can contribute to produce recommendations, their contribution depends upon their arrival time. In particular, the preference of a user u for an item i is weighted appropriately with the use of a temporal decay function. Typically, in temporal applications, the exponential fading function is employed, so the weight of $preference(u, i)$ decreases exponentially with time via the function: $2^{-\lambda(t-t_{u,i})}$, where $t_{u,i}$ is the time that the preference was defined and t is the current time. Thus, $t - t_{u,i}$ is actually the age of the preference. The parameter λ, $\lambda > 0$, is the decay rate that defines how fast the past history is forgotten. The higher λ, the lower the importance of historical preferences compared to more recent preferences.

Under this aging scheme, the so-called *damped relevance* of an item i for a user u with respect to a query Q in a given timepoint t is given by:

$$relevance^d(u, i, Q) = \frac{\sum_{u' \in (\mathcal{P}_{u,Q} \cap \mathcal{Z}_i)} 2^{-\lambda(t-t_{u',i})} \times contribution(u, u') \times preference(u', i)}{\sum_{u' \in (\mathcal{P}_{u,Q} \cap \mathcal{Z}_i)} contribution(u, u')}$$

So, all user item scores are weighted by the recency $2^{-\lambda(t-t_{u',i})}$.

Since all preferences are active, the *damped support* of i for u under Q is equal to the corresponding time-free support, that is:

$$support^d(u, i, Q) = support^f(u, i, Q)$$

Finally, the *damped value* of i for u under Q is computed as in the time-free case by aggregating the relevance and support scores ($\sigma \in [0, 1]$):

$$value^d(u, i, Q) = \sigma \times relevance^d(u, i, Q) + (1 - \sigma) \times support^d(u, i, Q)$$

Sliding window model. The sliding window model employs an alternative method which exploits only a subset of the available preferences, and in particular, the most recent ones. The size of this subset, referred to as window size, might be defined in terms of timepoints (e.g., use the preferences defined after Jan 2011) or records (e.g., use the 1000 most recent preferences). We adopt the first case. The preferences within the window are the active preferences that participate in the recommendations computation. Let t be the current time and W be the window size. Then, a preference of a user u for an item i, $preference(u, i)$, is active only if $t_{u,i} > t - W$.

In the sliding window model, the *sliding relevance* of an item i for a user u under a query Q is defined with regard to the active preferences of the peers of u for i. More specifically:

$$relevance^s(u, i, Q) = \frac{\sum_{u' \in (\mathcal{P}_{u,Q} \cap \mathcal{X}_i)} contribution(u, u') \times preference(u', i)}{\sum_{u' \in (\mathcal{P}_{u,Q} \cap \mathcal{X}_i)} contribution(u, u')}$$

where $\mathcal{X}_i$ is the set of users in $\mathcal{Z}_i$, such that, $\forall u' \in \mathcal{X}_i$, $t_{u',i} > t - W$.

The *sliding support* of i for u under Q is defined as the fraction of peers of u that have expressed preferences for i that are active at time t. That is:

$$support^s(u, i, Q) = |\mathcal{P}_{u,Q} \bigcap \mathcal{X}_i| / |\mathcal{P}_{u,Q}|$$

Finally, the *sliding value* of i for u under Q, for $\sigma \in [0, 1]$, is:

$$value^s(u, i, Q) = \sigma \times relevance^s(u, i, Q) + (1 - \sigma) \times support^s(u, i, Q)$$

3.2 Temporal Context-based Recommendations

In contrast to fresh-based recommendations, the context-based ones assume that although the preferences may change over time, they display some kind of temporal repetition. Or in other words, users may have different preferences under different temporal contexts. For instance, during the weekend a user may prefer to watch different movies from those in the weekdays. So, a movie recommendation system should provide movie suggestions for the weekends that may differ from the suggestions referring to weekdays.

As above, the rating of a user for an item, $preference(u, i)$, is associated with the rating time $t_{u,i}$. Time is modeled here as a multidimensional attribute. The dimensions of time have a hierarchical structure, that is, time values are organized at different levels of granularity (similar to [16, 18]). In particular, we consider three different levels over time: $time\_of\_day$, day_of_week and $time\_of\_week$ with domain values {"morning", "afternoon", "evening", "night"}, {"Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"} and {"Weekday", "Weekend"}, respectively. It is easy to derive such kind of information from the time

value $t_{u,i}$ that is associated with each user rating by using SQL or other programming languages. More elaborate information can be extracted by using the WordNet or other ontologies.

Let Θ be the current temporal context of a user u. We define the *context-based relevance* of an item i for u under a query Q expressed at Θ based on the preferences of the peers of u for i that are defined for the same context Θ. Formally:

$$relevance^c(u,i,Q) = \frac{\sum_{u' \in (\mathcal{P}_{u,Q} \cap \mathcal{Y}_i)} contribution(u,u') \times preference(u',i)}{\sum_{u' \in (\mathcal{P}_{u,Q} \cap \mathcal{Y}_i)} contribution(u,u')}$$

where $\mathcal{Y}_i$ is the set of users in $\mathcal{Z}_i$, such that, $\forall u' \in \mathcal{Y}_i$, $t_{u',i} \mapsto \Theta$, that is, the user rating has been expressed for a context equal to Θ. For example, if the temporal context of a user query is "Weekend", only the user preferences given for context "Weekend" would be considered.

The *context-based support* of i for u under Q is defined with respect to the number of peers of u that have expressed preferences for i under the same to the query context. That is:

$$support^c(u,i,Q) = |\mathcal{P}_{u,Q} \bigcap \mathcal{Y}_i| / |\mathcal{P}_{u,Q}|$$

Similar to the fresh-based recommendations, the *context-based value* of i for u under Q is calculated taking into account the context-based relevance and support. For $\sigma \in [0,1]$:

$$value^c(u,i,Q) = \sigma \times relevance^c(u,i,Q) + (1-\sigma) \times support^c(u,i,Q)$$

3.3 Top-k Time-aware Recommendations

Next, we define the time-aware variation of the top-k recommendations applicable to both fresh-based and context-based approaches.

Definition 6 (Top-k Time-aware Recommendations). *Let $\mathcal{U}$ be a set of users and $\mathcal{I}$ be a set of items. Given a query Q posed by a user $u \in \mathcal{U}$ at time t mapped to Θ, recommend to u a list of k items $\mathcal{I}_u =< i_1, \ldots, i_k >$, $\mathcal{I}_u \subseteq \mathcal{I}$, such that:*

(i) $\forall i_j \in \mathcal{I}_u$, $\nexists preference(u,i_j)$, *for the fresh-based recommendations, and* $\forall i_j \in \mathcal{I}_u$, $\nexists preference(u,i_j)$ *that is associated with context equal to Θ, for the context-based recommendations,*

(ii) $value^o(u,i_j,Q) \geq value^o(u,i_{j+1},Q)$, $1 \leq j \leq k-1$, $\forall i_j \in \mathcal{I}_u$, *and*

(iii) $value^o(u,i_j,Q) \geq value^o(u,x_y,Q)$, $\forall i_j \in \mathcal{I}_u$, $x_y \in \mathcal{I} \backslash \mathcal{I}_u$,

where o corresponds to the same d (for the damped window model), s (for the sliding window model), or c (for the context-based model).

The first condition ensures that the suggested items do not include already evaluated items by the user either in general or under a specific context, while the second and the third conditions resemble those of Definition 5.

4 Time-aware Recommendations Computation

Assume a user that submits a query presenting his information needs. Each query is enhanced with a contextual specification expressing some temporal information. This temporal information of the query may be postulated by the application or be explicitly provided by the user as part of his query. Typically, in the first case, the context implicitly associated with a query corresponds to the current context, that is, the time of the submission of the query. As a query example, for a restaurant recommendation application, consider a user looking for restaurants serving chinese cuisine during the weekend. As part of his/her query, the user should also provide the aging scheme that will be used.

Then, we locate the peers of the user (Sect. 4.1) and employ their preferences for estimating the time-aware recommendations (Sect. 4.2). Recommendations are presented to the user along with explanations on the reasons behind them (Sect. 4.3). In following, we overview the details of each step.

4.1 Selecting Peers

Our model assumes three different kinds of peers, namely close friends, domain experts and similar users. For each submitted query Q of a user u, u specifies the peers that will be used for producing his/her recommendations. This selection step of the peers is, in general, application dependent. For example, when a user is asking for advice for a personal computer, the domain experts may fit well to the user needs, while when asking for a suggestion about a movie, the user's close friends may provide good answers. In a similar manner, when using a trip advisor, the choice of users with similar tastes seems appropriate.

For the close friends case, the set of peers of u consists of the close friends of u, while for the domain experts case, the set of peers of u consists of the users that are considered to be experts for Q. We assume that this information is already known. For the similar users case, we need to calculate all similarity measures $simU(u, u')$ for all users $u' \in \mathcal{U}$. Those users u' with similarity $simU(u, u')$ greater than or equal to the threshold δ represent the similar users of u.

4.2 Computing Recommendations

Having established the methodology for finding the peers of a user, we focus next on how to generate valued recommendations for him/her. Given a user $u \in \mathcal{U}$ and his/her peers $\mathcal{P}_{u,Q}$, the procedure for estimating the value score of an item i for u requires the computation of the relevance and support of i. Note that we do not compute value scores for all items in $\mathcal{I}$, but only for the items $\mathcal{I}'$, $\mathcal{I}' \subseteq \mathcal{I}$, that satisfy the query selection conditions. To do this, we perform a pre-processing step to select the relevant to the query data by running a typical database query. For example, for a query about destinations in Greece posed to a travel recommendation system, we ignore all the rest destinations.

For computing the scores of the items in $\mathcal{I}'$, pairs of the form $(i, value^o(u, i, Q))$ are maintained in a set $\mathcal{V}_u$, where o corresponds to d, s or c for the damped window, sliding window and context-based approach, respectively. As a post-processing step, we rank all pairs in $\mathcal{V}_u$ on the basis of their value score. To

provide the top-k recommendations to u, we report the k items with the highest scores, i.e., the k first items in $\mathcal{V}_u$.

Next, we discuss separately the particulars of each time-aware recommendation approach. For the damped window approach, all the preferences of the peers of u are employed for computing recommendations. However, this is not the case for the other two approaches, where only a subset of the peers preferences are taken into consideration. More specifically, for the sliding window approach, only the most recent preferences are used, while for the context-based approach, the preferences that are defined for a temporal context equal to the query context. This can be seen as a preference pre-filtering step. It is worth noting that, since some preferences are ignored, some of the peers may not contribute finally to the recommendation list construction.

Moreover, for the context-based approach, the associated set of preferences for a specific query may be empty, that is, there may be no preferences for the query. In this case, we can use for the recommendation process these preferences whose context is more general than the query context. For example, for a query with context "Sat", we can use a preference defined for context "Weekend". The selection of the appropriate preferences can be made more efficient by deploying indexes on the context of the preferences. Such a data structure that exploits the hierarchical nature of context, termed profile tree, is introduced in [18].

As a final note, consider that the two approaches for computing time-aware recommendations can be applied together. For instance, we can apply the context-based approach first. Then, we can apply the damped window approach. This way, the importance of the preferences that are defined for the query context decreases with time.

4.3 Presenting Recommendations

After identifying the k items with the highest value scores for a user u, u is presented with these items. Recently, it has been shown that the success of recommendations relies on explaining the cause behind them [19]. This is our motivation for providing an explanation along with each suggested item, i.e., for explaining why this specific recommendation appears in the top-k list.

To do this, we present recommendations along with their explanations as text by using a simple template mechanism. Since explanations depend on the employed approach, different templates are associated with the two different approaches.

For the fresh-based approach, the reporting results have the following form:
ITEM @i IS RECOMMENDED BY THE SYSTEM
BECAUSE OF ITS HIGH VALUE SCORE, @$value^o$,
COMPUTED USING THE RECENT PREFERENCES OF YOUR @$peers$.
@$|\mathcal{P}_{u,Q} \bigcap \mathcal{T}_i|$ USERS OUT OF YOUR @$|\mathcal{P}_{u,Q}|$ PEERS HAVE RATED THIS ITEM.
In this case, o corresponds to d or s and $\mathcal{T}_i$ to $\mathcal{Z}_i$ or $\mathcal{X}_i$ for the damped window or the sliding window approach, respectively.

Similarly, for the context-based approach, the results are presented as follows:
ITEM @i IS RECOMMENDED BY THE SYSTEM
BECAUSE OF ITS HIGH VALUE SCORE, @$value^c$,

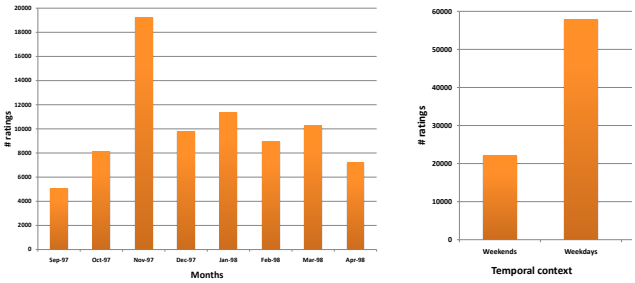
COMPUTED USING THE PREFERENCES, DEFINED FOR A TEMPORAL CONTEXT EQUAL TO THE QUERY ONE, OF YOUR @*peers*.
@$|\mathcal{P}_{u,Q} \bigcap \mathcal{Y}_i|$ USERS OUT OF YOUR @$|\mathcal{P}_{u,Q}|$ PEERS HAVE RATED THIS ITEM.

We use the symbol @ to mark parameter variables. Variables are replaced with specific values at instantiation time. For a movie recommendation system, an example of a reported result with its explanation is the following.

ITEM *Dracula* IS RECOMMENDED BY THE SYSTEM
BECAUSE OF ITS HIGH VALUE SCORE, 0.9,
COMPUTED USING THE PREFERENCES, DEFINED FOR A TEMPORAL CONTEXT EQUAL TO THE QUERY ONE, OF YOUR *close friends*.
27 USERS OUT OF YOUR 68 PEERS HAVE RATED THIS ITEM.

5 Experiments

In this section, we evaluate the effectiveness of our time-aware recommendation system using a real movie ratings dataset [1], which consists of 100,000 ratings given from September 1997 till April 1998 by 1,000 users for 1,700 items. The monthly split is shown in Fig. 1(a), while the split per weekends and weekdays is shown in Fig. 1(b).



(a) (b)

Fig. 1. (a) Ratings per month and (b) ratings per temporal context.

Since there is no information about actual friends and experts in this dataset, we employ as the peers of a given user his/her similar users. To this end, the notion of user similarity is important. We use here a simple variation; that is, we use distance instead of similarity. More specifically, we define the distance between two users as the Euclidean distance over the items rated by both. Let $u, u' \in \mathcal{U}$ be two users, $\mathcal{I}_u$ be the set of items for which $\exists preference(u, i)$, $\forall i \in \mathcal{I}_u$, and $\mathcal{I}_{u'}$ be the set of items for which $\exists preference(u', i)$, $\forall i \in \mathcal{I}_{u'}$. We denote by $\mathcal{I}_u \cap \mathcal{I}_{u'}$ the set of items for which both users have expressed preferences. Then, the distance between u, u' is:

$$distU(u, u') = \sqrt{\sum_{i \in \mathcal{I}_u \cap \mathcal{I}_{u'}} (preference(u, i) - preference(u', i))^2} / |\mathcal{I}_u \cap \mathcal{I}_{u'}|$$

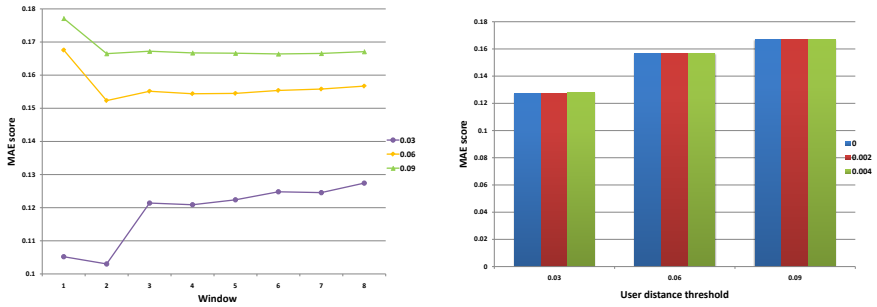
To evaluate the quality of the recommendations, we use a predictive accuracy metric that directly compares the predicted ratings with the actual ones [12]. A commonly used metric in the literature is the *Mean Absolute Error* (MAE), which is defined as the average absolute difference between predicted ratings and actual ratings: $MAE = \sum_{u,i} |preference(u, i) - value^o(u, i, Q)|/N$, where N is

the total number of ratings in the employed dataset and o corresponds to d, s or c. Clearly, the lower the MAE score, the better the predictions.

Next, we report on the results for the sliding window model, the damped window model and the context-based model compared to the time-free model.

Sliding window model. To illustrate the effectiveness of the sliding window model, we use windows of different sizes W. The window size $W = 1$ stands for the most recent month, i.e., April 1998, the window size $W = 2$ stands for both April 1998 and March 1998, and so forth. The window size $W = 8$ includes the whole dataset, from April 1998 till September 1997. We denote the resulting dataset as D_W, where $W = [1 - 8]$ is the window size. For each dataset D_W, we compute the recommendations for each user by considering the user ratings within the corresponding window W. We compare the predicted values with the actual values given by the user within the same window W and report the average results.

The results for different windows are presented in Fig. 2(a). In the same figure, we also show the effect of the user distance threshold. In general, recommendations present better quality for small windows (this is not the case for the smallest window size ($W = 1$) because of the small amount of ratings used for predictions). For example, for a user distance threshold equal to 0.03 and $W = 2$, the predictions are improved around 2.5% compared to $W = 8$ (i.e., compared to the time-free recommendations model). Or, for a threshold equal to 0.06 and $W = 3$, the predictions are improved on average 0.5% compared to $W = 8$. Moreover, the larger the window, the smaller the improvement. Regarding the effect of the user similarity thresholds, as expected, for larger user distance thresholds, the MAE scores increase for all window sizes, since more dissimilar users are considered for the suggestions computation.



(a) (b)

Fig. 2. MAE scores for (a) the sliding window and (b) the damped window model.

Damped window model. Next, we evaluate the effect of the decay rate λ in the recommendations accuracy. We use different values for λ; the higher the λ is, the less the historical data count. The value $\lambda = 0$ corresponds to the time-free model. We downgrade the original ratings based on the decay factor λ and the time difference between the end of the observation period (22/04/1998) and the ratings timestamp.

The results of this experiment are shown in Fig. 2(b). Practically, this aging model seems to not offer any (or offer a very small) improvement in this setting, i.e., for the employed dataset. As above, larger distance thresholds lead to larger MAE scores.

Context-based recommendations. In this set of experiments, we demonstrate the effect of temporal context on producing recommendations. We consider two different temporal contexts "Weekends" and "Weekdays". For the "Weekends" context, we base our predictions only on ratings defined for weekends ($D_{weekends}$), whereas for the "Weekdays" context, we consider ratings from Monday to Friday ($D_{weekdays}$). The predicted values are compared to the actual values given by the user within the same temporal context through the MAE metric.

Fig. 3 displays the results. Except for the two temporal contexts, "Weekends" and "Weekdays", we also present the scores for the time-free model, i.e., when the whole dataset is used. Generally speaking, the temporal context affects the recommendations accuracy. In particular, for both contexts, "Weekends" and "Weekdays", the quality of the recommendations is improved compared to the time-free approach that completely ignores the temporal information of the ratings. For example, for a user distance threshold equal to 0.03, the predictions for "Weekends" are improved on average 0.95% when using ratings for "Weekends" instead of using the whole rating set. Similarly, for a distance equal to 0.06, the predictions for "Weekdays" are improved around 0.5%. Also, larger distance thresholds values result in larger MAE scores, that is, the quality of the recommendations decreases with the user distance threshold.

In overall, time plays an important role towards improving the quality of the proposed recommendations. The sliding window and the context-based approaches increase the recommendations accuracy. However, a mere decay model seems to be not adequate. Our goal is to design a more elaborate aging scheme that considers not only the age of the ratings but also other parameters, such as the recency and popularity of the recommended items and the context under which the ratings were given. We expect that the time effect will be more evident for datasets that span a larger period of time. Further experimentation with other kinds of peers provided by the application dataset will also be interesting.

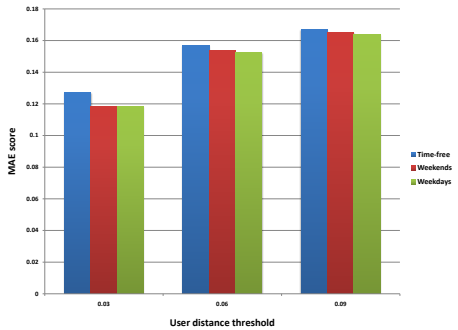


Fig. 3. MAE scores for context-based approach.

6 Related Work

The research literature on recommendations is extensive. Typically, recommendation approaches are distinguished between: *content-based*, that recommend items similar to those the user previously preferred (e.g., [17, 13]), *collaborative filtering*, that recommend items that users with similar preferences liked (e.g., [11, 8]) and *hybrid*, that combine content-based and collaborative ones (e.g., [5]). Several extensions have been proposed, such as employing multi-criteria ratings (e.g., [2]) and defining recommendations for groups (e.g., [4, 15, 14]).

Recently, there are also approaches focusing on enhancing recommendations with further contextual information (e.g., [3, 16]). In these approaches, context is defined as a set of dimensions, or attributes, such as location, companion and time, with hierarchical structure. While a traditional recommendation system considers only two dimensions that correspond to users and items, a context-aware recommendation system considers one additional dimension for each context attribute. In our approach, we focus on a particular case of this model, that is, the three-dimensional recommendations space among users, items and time, since our specific goal is to study how the time effects contribute to the improvement of predictions.

Moreover, there are some approaches which incorporate temporal information to improve recommendations effectiveness. [21] presents a graph-based recommendation system that mixes long-term and short-term user preferences to improve predictions accuracy, while [20] considers how time can be used into matrix factorization models by examining changes in user and society tastes and habits, and items popularity. [9] uses a strategy, similar to our damped window model, that decreases the importance of known ratings as time distance from recommendation time increases. However, the proposed algorithm uses clustering to discriminate between different kinds of items. [7] introduces the idea of micro-profiling, which splits the user preferences into several sets of preferences, each representing the user in a particular temporal context. The predictions are computed using these micro-profiles instead of a single user model. The main focus of this work is on the identification of a meaningful partition of the user preferences using implicit feedback. In our paper, the goal is to examine time from different perspectives. This way, we use a general model for time, considering time either as specific time instances or specific temporal conditions, in order to define a unified time-aware recommendation model.

7 Conclusions

In this paper, we studied different semantics to exploit the time information associated with user preferences to improve the accuracy of recommendations. We considered various types of time effects, and thus, proposed different time-aware recommendation models. Fresh-based recommendations care mainly for recent and novel preferences, while context-based recommendations are computed with respect to preferences with temporal context equal to the query context. Finally, we demonstrated our approach using a real dataset of movie ratings. There are many directions for future work. One is to extend our framework so as to support a novel mode of interaction between users and recommendation systems;

our goal is to exploit the whole rating history to produce valued recommendations and, at the same time, use the fresh ratings to assist users in database exploration.

References

1. Movielens data sets. Available online at: http://www.grouplens.org/node/12; visited on Nov.2011.
2. G. Adomavicius and Y. Kwon. New recommendation techniques for multicriteria rating systems. *IEEE Intelligent Systems*, 22(3):48–55, 2007.
3. G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst.*, 23(1):103–145, 2005.
4. S. Amer-Yahia, S. B. Roy, A. Chawla, G. Das, and C. Yu. Group recommendation: Semantics and efficiency. *PVLDB*, 2(1):754–765, 2009.
5. M. Balabanovic and Y. Shoham. Content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72, 1997.
6. K. Balog, T. Bogers, L. Azzopardi, M. de Rijke, and A. van den Bosch. Broad expertise retrieval in sparse data environments. In *SIGIR*, pages 551–558, 2007.
7. L. Baltrunas and X. Amatriain. Towards time-dependant recommendation based on implicit feedback. In *CARS*, pages 1–5, 2009.
8. J. S. Breese, D. Heckerman, and C. M. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI*, pages 43–52, 1998.
9. Y. Ding and X. Li. Time weight collaborative filtering. In *CIKM*, pages 485–492, 2005.
10. J. Gama. *Knowledge Discovery from Data Streams*. CRC Press, 2010.
11. J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. Grouplens: Applying collaborative filtering to usenet news. *Commun. ACM*, 40(3):77–87, 1997.
12. P. Melville and V. Sindhwani. Recommender systems. In *Encyclopedia of Machine Learning*, pages 829–838. 2010.
13. R. J. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *ACM DL*, pages 195–204, 2000.
14. I. Ntoutsi, K. Stefanidis, K. Nørvåg, and H.-P. Kriegel. Fast group recommendations by applying user clustering. In *ER*, 2012.
15. M. O'Connor, D. Cosley, J. A. Konstan, and J. Riedl. Polylens: A recommender system for groups of user. In *ECSCW*, pages 199–218, 2001.
16. C. Palmisano, A. Tuzhilin, and M. Gorgoglione. Using context to improve predictive modeling of customers in personalization applications. *IEEE Trans. Knowl. Data Eng.*, 20(11):1535–1549, 2008.
17. M. J. Pazzani and D. Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27(3):313–331, 1997.
18. K. Stefanidis, E. Pitoura, and P. Vassiliadis. Managing contextual preferences. *Inf. Syst.*, 36(8):1158–1180, 2011.
19. N. Tintarev and J. Masthoff. Designing and evaluating explanations for recommender systems. In *Recommender Systems Handbook*, pages 479–510. Springer, 2011.
20. L. Xiang and Q. Yang. Time-dependent models in collaborative filtering based recommender system. In *Web Intelligence*, pages 450–457, 2009.
21. L. Xiang, Q. Yuan, S. Zhao, L. Chen, X. Zhang, Q. Yang, and J. Sun. Temporal recommendation on graphs via long- and short-term preference fusion. In *KDD*, pages 723–732, 2010.