A Framework for Modeling, Computing and Presenting Time-aware Recommendations

Kostas Stefanidis[1,3], Eirini Ntoutsi[2], Mihalis Petropoulos[2], Kjetil Nørvåg[3], and Hans-Peter Kriegel[2]

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

Abstract. Lately, recommendation systems have received significant attention. Most existing approaches though, recommend items of potential interest to users by completely ignoring the temporal aspects of ratings. 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 different aging schemes for decreasing the effect of historical ratings and increasing the influence of fresh and novel ratings. From a *context-based* perspective, we focus on providing different suggestions under different temporal specifications. To facilitate user browsing, we propose an effective presentation layer for time-aware recommendations based on user preferences and summaries for the suggested items. Our experiments with real movies ratings show that time plays an important role in the recommendation process.

1 Introduction

Recommendation systems provide users with suggestions about products, movies, videos and a variety of other items. A popular category of recommendation systems is the collaborative filtering approaches (e.g., [21, 11]) 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 ratings of these users. Users are considered as similar if they buy common items as in case of Amazon or if they provide similar movie evaluations as in case of MovieLens.

Although there is a substantial amount of research in the area of recommendation systems [34], 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. 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, namely the fresh-based and the context-based recommendations model.

The *fresh-based recommendations model* assumes that the most recent user ratings better reflect his/her current trends and thus, they should 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 ratings and ignores any previous historical information. As an example, consider a movie recommendation system that gives higher priority to new releases compared to other old seasoned movies (damped window model) or focuses solely on new releases (sliding window model).

From a different perspective, the *context-based recommendations model* offers different suggestions under different time specifications. The main motivation here, is that although user preferences may change over time they display temporal repetition, i.e., recur over time. As an example consider a tourist guide system that provides different suggestions for winter (typically ski resorts) and summer (typically sea resorts). 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 time-aware recommendations that handles the different temporal aspects of recommendations through the fresh-based or the context-based model. Apart from the top-k recommendations extraction, we also focus on their effective presentation to the end user by adding structure in the results. Our goal is to minimize the browsing effort of the user and help him/her receive a broader view of the recommended items. Towards this direction, we exploit preferences defined by users upon items and extract a ranking of preferences that is used for ordering the suggested items. We further enrich this structure by summarizing the different levels of preferences with information for the items.

In a nutshell, this paper makes the following contributions:

- We propose a framework for time-aware recommendations that models the different types of time effects, that is, the age and the temporal context of ratings. Furthermore, we consider different cases for selecting the appropriate set of users for estimating the recommendations of a user and introduce the notion of support in recommendations to model how confident the recommendations of an item for a user is, in order to deal with the sparsity of the explicitly defined user ratings.
- We propose an effective presentation solution for the recommended items which builds upon user preferences for items. Our solution provides a ranked

overview of the suggested items enriched with summarized information and can facilitate user browsing.
– We implement a proof of concept prototype for time-aware recommendations in a movie recommendations application and we experiment with different types of aging and temporal contexts. Our experiments show that time is an important dimension and should be part of the recommendation process.

The rest of the paper is organized as follows. The basic, time-invariant recommendation model is presented in Sect. 2. The time dimension is introduced in Sect. 3, where we distinguish between the aging factor (Sect. 3.1) and the temporal context factor (Sect. 3.2). In Sect. 4, we focus on the effective presentation of time-aware recommendations based on user preferences. The computation of recommendations under different temporal semantics is discussed in Sect. 5. In Sect. 6, we present our experiments using a real dataset of movie ratings. Our prototype implementation is outlined in Sect. 7, while related work is presented in Sect. 8. Finally, conclusions and outlook are pointed out in Sect. 9.

2 The Basic Time-free Recommendation Model

Assume a set of items $\mathcal{I}$ with relational schema $R(A_1, \ldots, A_d)$, where each attribute A_j, $1 \leq j \leq d$, takes values from a domain $dom(A_j)$. Let $A = \{A_1, \ldots, A_d\}$ be the attribute set of R and $dom(A) = dom(A_1) \times \ldots \times dom(A_d)$ be its value domain. We use i to denote an item in $dom(A)$ of R. For instance, consider the movies shown in Fig. 1.

mID	*title*	*year*	*director*	*genre*	*language*	*duration*
1	Casablanca	1942	Curtiz	Drama	English	102
2	Vertigo	1958	Hitchcock	Horror	English	128
3	Psycho	1960	Hitchcock	Horror	English	109
4	Schindler's List	1993	Spielberg	Drama	English	195
5	The Farmer's Wife	1945	Hitchcock	Drama	English	129
6	Suspicion	1941	Hitchcock	Drama	English	99
7	Twilight Zone: The Movie	1983	Spielberg	Horror	English	101
8	Arachnophobia	1990	Spielberg	Horror	English	103
9	Lincoln	2012	Spielberg	Drama	English	150
10	The Walking Dead	1936	Curtiz	Horror	English	66

Fig. 1. Movies instance.

Assume also a set of users $\mathcal{U}$. Each user $u \in \mathcal{U}$ may give a rating for an item $i \in \mathcal{I}$, which is denoted by $rating(u, i)$ and lies in the range $[0.0, 1.0]$. For instance, consider the ratings shown in Fig 2. We use $\mathcal{Z}_i$ to denote the set of users in $\mathcal{U}$ that have expressed a rating 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.

uID	mID	$rating$	$timestamp$
1	3	0.9	1296367200
1	1	0.6	1297317600
2	2	0.7	1294639200
2	3	0.9	1298181600

Fig. 2. Ratings instance.

In this section, we first present the basic model for time-free recommendations (Sect. 2.1) and then define the top-k recommendations problem (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., [33, 28]), (ii) *collaborative filtering*, that recommend items that similar users have liked in the past (e.g., [21, 11]) and (iii) *hybrid*, that combine content-based and collaborative filtering approaches (e.g., [8]).

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 ratings 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 consider three different aspects of peers: (i) *close friends*, (ii) *area experts* and (iii) *similar users*.

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

An alternative solution might be the implicit extraction of the set of friends through some social network like Facebook or Google+.

From a different perspective, *area experts* can be used for producing recommendations for specific queries, since they are considered to be knowledgeable on a specific topic, domain or area. Several methods deal with the problem of finding experts (e.g., [9]); 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.

AREA EXPERTS: Let $\mathcal{U}$ be a set of users and Q be a query. The *area experts* $\mathcal{D}_Q$, $\mathcal{D}_Q \subseteq \mathcal{U}$, are the users considered as experts for the query 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 ratings 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 two users 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 $simU(u, u')$ above a given threshold.

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 k most similar users to u. Our main motivation here is that we opt for selecting only highly relevant users.

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

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 area 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 formally define the relevance of an item for a user as follows:

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

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

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

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 $rating(u', i)$ for u; this importance depends on how "reliable" u' is for u. When close friends or area experts are used, contribution is set to 1, since we are certain about the importance of the ratings of the selected users to the given user. For the similar users case, the contribution of each user u' depends on his/her similarity to u.

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

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

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

Intuitively, the notion of support expresses how reliable is our estimation of the relevance of item i for user u.

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, $\nexists rating(u, i)$, 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 linear combination of relevance and support 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) $\forall i_j \in \mathcal{I}_u$, $\nexists rating(u, i_j)$,

(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 basic time-free recommendation model presented above assumes that all ratings are active and potentially they could be exploited for 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). As such, the recent user ratings reflect better his/her current interests comparing to older possible obsolete ratings. From another point of view, user interests might change under different temporal circumstances and thus, users may have different needs depending on the temporal context. 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 other hobbies.

To handle such different cases, we propose a framework for time-aware recommendations that incorporates the notion of time in the recommendation process towards accuracy improvement. 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 problem 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 vs weekends).

In our time-aware recommendation model, the rating of a user u for an item i, $rating(u, i)$, is associated with a timestamp $t_{u,i}$, which is the time that i was rated by u (c.f., Fig 2) and thus, it denotes 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 in a recommendation application changes over time; typically, items, e.g., movies, pictures or songs, lose popularity as time goes by. Motivated by the intuition that the importance of item ratings

increases with the popularity of the items themselves, fresh-based recommendations suggest items by mainly exploiting recent and novel user ratings.

Driven by the work in data streams [18], 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 further processing. 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 ratings given 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* which considers the whole rating history from a given landmark.

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

Under this aging schema, 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:

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

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

Since all ratings 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 combining 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. In the sliding window model only a subset of the available ratings is exploited, 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 ratings given within the last month) or records (e.g., use the 1000

most recent ratings). We adopt the first case. The ratings within the window are the active ratings that participate in the recommendation computation. Let t be the current time and W be the window size. Then, a rating of a user u for an item i, $rating(u, i)$, 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 ratings of the peers of u for i. More specifically:

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

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 ratings for i that are active at time t. That is:

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

Finally, the *sliding value* of i for u under Q, for $\sigma \in [0, 1]$, is a linear combination of their relevance and support scores:

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

3.2 Temporal Context-based Recommendations

In contrast to fresh-based recommendations, the context-based ones assume that although the user 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, $rating(u, i)$, 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 [32, 35]). 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 ratings of the peers of u for i that are defined for the same context Θ. Formally:

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

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 ratings given for the 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 ratings for i under the same temporal context as the query context. That is:

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

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 recommendation problem (c.f., Sec 2.2) 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 a temporal context 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) $\forall i_j \in \mathcal{I}_u$, $\nexists rating(u, i_j)$, *for the fresh-based recommendations, and* $\forall i_j \in \mathcal{I}_u$, $\nexists rating(u, i_j)$ *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 either 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 Def. 5.

4 Presentation of Time-aware Recommendations based on User Preferences

Depending on the value of k and the recommendation application per se, the top-k recommendations for a user u might result in a lot of information for u. To facilitate the user selection, we propose to organize the results in a compact yet intuitive and representative way. To achieve this goal, we employ, apart from ratings, preferences expressed by users over items. These user preferences might

be either qualitative (e.g., the director is more important than the genre of the movie) or quantitative (e.g., the preference scores for the directors Q. Tarantino, F. F. Coppola are 0.9, 0.5, respectively).

In the following, we discuss in more details how preferences can be given (Sect. 4.1) and we present a formal model for the effective presentation of user top-k recommendations based on his/her preferences (Sect. 4.2).

4.1 User Preferences

In general, preferences can be expressed either in a qualitative or in a quantitative way. Following a qualitative preference model, users employ binary relations to directly define preferences between data items (e.g., [13, 20]). Following a quantitative preference model, users provide numeric scores via scoring functions to indicate their degree of interest (e.g., [6, 22, 35]). We use a qualitative preference model [13], since this model is more general than the quantitative one and also closer to the users intuition. Specifically:

Definition 7 (Preference Model). *Let $\mathcal{U}$ be a set of users and $\mathcal{I}$ be a set of items with relational schema $R(A_1, \ldots, A_d)$. For a user $u \in \mathcal{U}$, assume a set of values P_u of an attribute A_j, $1 \leq j \leq d$, such that, $P_u \subseteq dom(A_j)$. The user u specifies a binary preference relation $pref_u$ on P_u, $pref_u = \{(p_1 \succ p_2) | p_1, p_2 \in P_u\}$, where $p_1 \succ p_2$ denotes that u prefers p_1 over p_2.*

For example, a user might prefer A. Hitchcock over S. Spielberg, i.e., *(A. Hitchcock $\succ$ S. Spielberg)*, and S. Spielberg over Q. Tarantino, i.e., *(S. Spielberg $\succ$ Q. Tarantino)*.

Alternatively, instead of providing comparative relationships, users could provide explicit preference scores for the values in P_u. This would correspond to a quantitative approach, with higher preference scores indicating more important preferences. For example, a user might assign to A. Hitchcock, S. Spielberg and Q. Tarantino the preference scores 0.9, 0.7 and 0.6, respectively. The transition from the quantitative to the qualitative approach is straightforward. For the aforementioned example, the qualitative equivalent is: *(A. Hitchcock $\succ$ S. Spielberg)*, *(A. Hitchcock $\succ$ Q. Tarantino)*, and *(S. Spielberg $\succ$ Q. Tarantino)*.

Irrespectively of their qualitative or quantitative formulation, preferences may also be expressed at different levels of granularity. We distinguish between value-based and attribute-based preferences. *Value-based preferences* are expressed between individual values of item attributes. Typically, they are formulated over the items of a relation based on the values of their attributes. An example of a value-based preference for the item attribute director could be *(A. Hitchcock $\succ$ Q. Tarantino)*. *Attribute preferences* express preferences between the different attributes of R, i.e., they evaluate how important for the end user each attribute or feature of the item description is. For example, a user might consider the attribute director more important than the attribute genre, i.e., *(director $\succ$ genre)*. Attribute preferences might be also expressed either qualitatively (e.g., [19]) or quantitatively (e.g., [26]). In what follows, we will use the

term $pref_u$ to denote the whole set of value-based and attribute-based preferences of a user u.

Clearly, preferences may be collected using various ways. Specifically, preferences can be provided explicitly by the users, as above, or constructed automatically, for instance, based on the past behavior of the user or of similar users. Such methods for the automatic construction of preferences have been the focus of much current research (e.g., [27]) and are beyond the scope of this paper. For our study, we assume that the set of preferences is provided for each user.

4.2 Time-aware Recommendations Presentation

A user in a recommendation application might express both value-based and attribute-based preferences. To combine these different types, we use attribute-based preferences to set priorities among value-based preferences based on the attributes involved in the preferences, similarly to [19]. For example, assume a user with value-based preference *(A. Hitchcock* $\succ$ *S. Spielberg)* over the attribute director and *(horror* $\succ$ *drama)* over the attribute genre. Assume also that our user considers the director of a movie to be more important than its genre which is expressed through the attribute-based preference *(director* $\succ$ *genre)*.

Given the above set of preferences, the following combined preferences can be drawn: our user prefers the set of values or keywords {*A. Hitchcock, horror*} over the set of values {*A. Hitchcock, drama*}. The latter set is preferred over the set {*S. Spielberg, horror*}, which in turn is preferred over the set {*S. Spielberg, drama*}.

The combined preferences of a user can be directly exploited for ranking the top-k recommendations of the user. The idea is to first extract the combined preferences and then use them to rank and present the top-k recommended items to the user.

This ranking could be also enhanced by exploiting the different attributes of the item description and building summaries upon these descriptions. We start with the brick of this concept, which is, the keyword-based summary.

Definition 8 (Keyword-based summary). *Let M be a set of keywords and $\mathcal{I}', \mathcal{I}' \subseteq \mathcal{I}$, be a set of items. A keyword-based summary, key-sum, is a pair (M: $\mathcal{I}'_M$), $\mathcal{I}'_M \subseteq \mathcal{I}'$, such that, all items in $\mathcal{I}'_M$ contain all keywords in M.*

For example, the keyword-based summary *({A. Hitchcock, horror}: {Psycho, Vertigo})* consists of a set of two keywords *(A. Hitchcock* and *horror)* that is associated with a set of movies *(Psycho* and *Vertigo)* that contain (or are related with) the keywords. In this example, the keywords {*A. Hitchcock, horror*} derived from user preferences are extended by the titles of the movies which are directed by A. Hitchcock and belong to the horror genre type. Other extension options could be employed as well, e.g., information about the production year or the duration of the movies. Also, the extension might refer to more than one attributes, e.g., both title and production year of a movie could be considered. Such a summary offers more information to the end user regarding the recommended item and facilitates his/her selection.

So far, we focus on the summary of a single combined preference. Our goal is to construct a summary for the top-k recommendations based on user preferences. This summary consists of an ordered set of keyword-based summaries, such that, a keyword-based summary $key\text{-}sum_i = (M_i, \mathcal{I}'_{M_i})$ appears before a keyword-based summary $key\text{-}sum_j = (M_j, \mathcal{I}'_{M_j})$, if the keywords of M_i are preferred over the keywords of M_j with respect to the available value-based and attribute-based preferences. Considering our previous example and using only the titles of the movies for the extension, the corresponding ordering would be: the summary *(*{*A. Hitchcock, horror*}*:* {*Psycho, Vertigo*}*)* is preferred over the summary *(*{*A. Hitchcock, drama*}*:* {*The Farmer's Wife, Suspicion*}*)*, which is preferred over the summary *(*{*S. Spielberg, horror*}*:* {*Twilight Zone: The Movie, Arachnophobia*}*)*, which in turn is preferred over *(*{*S. Spielberg, drama*}*:* {*Lincoln, Schindler's List*}*)*.

Note also that there might be cases where the preferences may be equivalent, e.g., {*M. Curtiz, horror*} is equally preferred to {*S. Spielberg, horror*}, and consequently, the corresponding keyword-based summaries would be equivalent. To accommodate such cases, we propose to summarize the equivalent keyword-based summaries in the so called keyword-based class summaries.

Definition 9 (Keyword-based class summary). *A keyword-based class summary, class-sum, is a set of keyword-based summaries* {*key-sum*$_1$*, ..., key-sum*$_n$}*, such that, the keywords in* M_1*, ...,* M_n *are considered equally preferable with respect to a given set of value and attribute preferences.*

For example, given that the sets of keywords {*M. Curtiz, horror*} and {*S. Spielberg, horror*} are equally preferable with respect to a specific set of preferences, then their keyword-based summaries, e.g., {*(*{*M. Curtiz, horror*}*:* {*The Walking Dead*}*)* and *(*{*S. Spielberg, horror*}*:* {*Twilight Zone: The Movie, Arachnophobia*}*)*}, constitute a keyword-based class summary.

Based on the keyword-based class summaries, we define formally the time-aware recommendations summary as follows:

Definition 10 (Time-aware recommendations summary). *Let* $\mathcal{U}$ *be a set of users,* $\mathcal{I}$ *be a set of items,* Q *be a query posed by a user* $u \in \mathcal{U}$ *at time* t *mapped to the temporal context* Θ *and* $pref_u$ *be the set of value and attribute preferences of* u*. Let also* $\mathcal{I}_u$ *be the top-k time-aware recommendations for* u*. The time-aware recommendation summary for* u *is a list of keyword-based class summaries tar-sum =* <*class-sum*$_1$*, ..., class-sum*$_x$>*, such that:*

(i) all sets of keywords in class-sum$_i$ *are preferred over all sets of keywords in class-sum*$_{i+1}$*,* $1 \leq i \leq x-1$*, with respect to* $pref_u$*,*

(ii) all keywords of the value preferences of $pref_u$ *appear in an* M *set of tar-sum and*

(iii) only the keywords of the M *sets of tar-sum appear in the value preferences of* $pref_u$*.*

So, given that the set of keywords {*A. Hitchcock, horror*} is preferred over the sets {*M. Curtiz, horror*} and {*S. Spielberg, horror*}, with the last two being

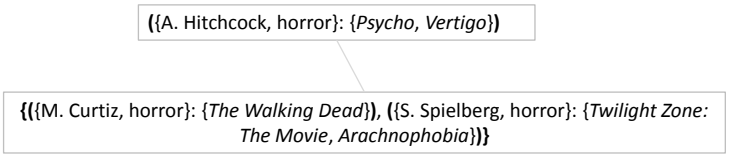


Fig. 3. A presentation example.

equally preferred, then the keyword-based summary for $\{A.\ Hitchcock,\ horror\}$ will be first displayed. The keyword-based class summary for $\{M.\ Curtiz,\ horror\}$ and $\{S.\ Spielberg,\ horror\}$ will follow. Schematically, this would look as in Fig. 3.

5 Time-aware Recommendations Computation

A high level representation of the main components of the architecture of our system is depicted in Fig. 4. 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 schema that will be used.

Then, we locate the peers of the user (Sect. 5.1) and employ their ratings for estimating the time-aware recommendations (Sect. 5.2). Finally, recommendations are summarized and presented to the user (Sect. 5.3). In following, we overview the details of each step.

5.1 Selecting Peers

Our model assumes three different kinds of peers, namely close friends, area 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 area 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 area 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

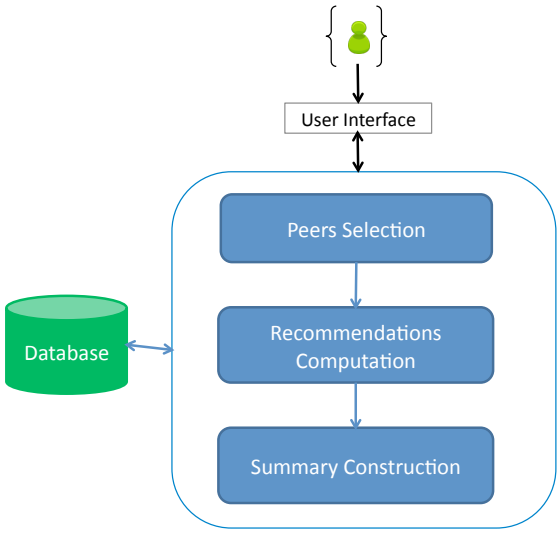


Fig. 4. System architecture.

$simU(u, u')$ 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 (Algorithm 1).

5.2 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.

Algorithm 2 presents the general procedure for computing the value 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 ratings 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 ratings are taken

Algorithm 1 Finding Similar Users Algorithm

Input: A set of users $\mathcal{U}$, a user $u \in \mathcal{U}$ and a threshold similarity value δ.
Output: The peers of u, $\mathcal{P}_{u,Q}$.

```
1: begin
2: P_{u,Q} = ∅;
3: for each user u' ∈ U\{u} do
4:    compute simU(u, u');
5:    if simU(u, u') ≥ δ then
6:       add u' to P_{u,Q};
7:    end if
8: end for
9: return P_{u,Q};
10: end
```

Algorithm 2 Value Computation Algorithm

Input: A user $u \in \mathcal{U}$, a query Q, the peers $\mathcal{P}_{u,Q}$ along with their ratings, the aging schema and the weight σ.
Output: A set $\mathcal{V}_u$ of pairs $(i, value^o(u, i, Q))$, $\forall i \in \mathcal{I}'$.

```
1: begin
2: V_u = ∅;
3: for each item i ∈ I' unrated by user u do
4:    compute relevance^o(u, i, Q);
5:    compute support^o(u, i, Q);
6:    compute value^o(u, i, Q);
7:    add (i, value^o(u, i, Q)) to V_u;
8: end for
9: return V_u;
10: end
```

into consideration. More specifically, for the sliding window approach, only the most recent ratings are used, while for the context-based approach, the ratings that are defined for a temporal context equal to the query context are employed. This can be seen as a rating pre-filtering step. It is worth noting that, since some ratings are ignored due to temporal specifications, some of the peers finally may not contribute at all to the recommendation list construction.

Moreover, for the context-based approach, the associated set of ratings for a specific query may be empty, that is, there may be no ratings for the query. In this case, we can use for the recommendation process these ratings whose context is more general than the query context. For example, for a query with context "Sat", we can use a rating given for context "Weekend". The selection of the appropriate ratings can be made more efficient by deploying indexes on the context of the ratings. Such a data structure that exploits the hierarchical nature of context, termed profile tree, is introduced in [35].

As a final note, the two approaches for computing time-aware recommendations can be combined. For instance, we can apply the context-based approach

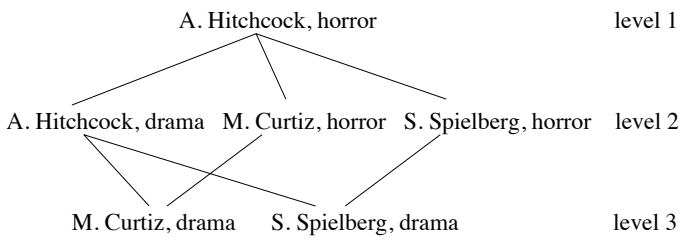


Fig. 5. A lattice example.

first. Then, we can apply the damped window approach. This way, the importance of the ratings that are defined for the query context decreases with time.

5.3 Presenting Recommendations

To facilitate users in item selection, we present the top-k time-aware recommended items for a user u in a compact and intuitive way, by employing his/her value-based and attribute-based preferences $pref_u$.

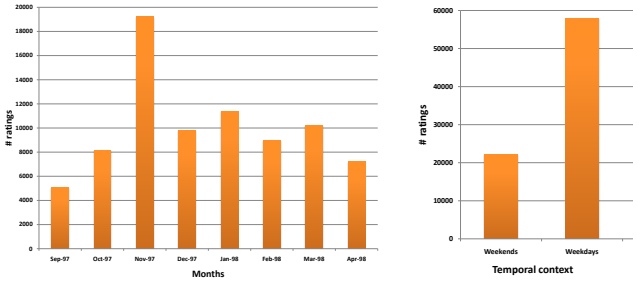
The problem we deal with here can be stated as follows: *Given a relation R describing the items in a recommendation application and the preferences $pref_u$ of u over R, how to produce ranked groups of items in R based on $pref_u$*. To this end, a lattice is built where the nodes correspond to the combinations of values appearing in $pref_u$. For example, for the list of preferences: (i) A. Hitchcock is preferred over M. Curtiz or S. Spielberg, (ii) horror movies are preferred over drama movies and (iii) the director of a movie is as important as its genre, the lattice of Fig. 5 is constructed. The top nodes are more important to the user comparing to the bottom nodes, whereas nodes lying in the same level of the lattice are of equal importance.

A query is formulated for each node in the lattice. Considering only the top-k items for recommendations, all queries in a specific level are associated with equally preferable items and each query is associated with the items that contain the keywords of the query. The queries of each level are successively executed starting from the queries of the top level and going down the lattice. For example, for the lattice of Fig. 5, items with keywords {*A. Hitchcock, horror*}, i.e., items in the result of the query of the first level, are preferred over the items with keywords {*S. Spielberg, horror*}, i.e., the items in the results of a query of the second level, and so on. Within the same level, items are ranked according to their recommendation value score.

Then, we construct a keyword-based summary for each query in the lattice. The set of queries in a level of the lattice corresponds to a keyword-based class summary, while the total set of ordered queries in the lattice represents the time-aware recommendation summary.

6 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. 6(a), while the split per weekends and weekdays is shown in Fig. 6(b).



(a) (b)

Fig. 6. (a) Ratings per month and (b) ratings per temporal context.

Since there is no information about actual friends and experts in the 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 $\exists rating(u, i)$, $\forall i \in \mathcal{I}_u$, and $\mathcal{I}_{u'}$ be the set of items for which $\exists rating(u', i)$, $\forall i \in \mathcal{I}_{u'}$. 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:

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

To evaluate the quality of the recommendations, we use a predictive accuracy metric that directly compares the predicted ratings with the actual ones [25]. 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: $MAE = \sum_{u,i} |rating(u, i) - value^o(u, i, Q)|/N$, 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.

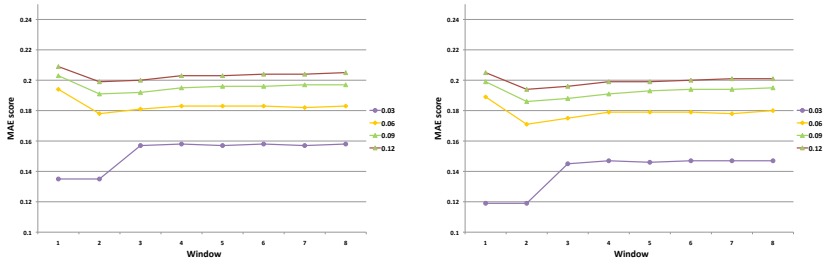
Sliding window model. To illustrate the effectiveness of the sliding window model, we use windows of different sizes W. The window size $W = 1$ stands for the

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, distance thresholds and σ values are presented in Fig. 7(a) (for $\sigma = 1.0$), Fig. 7(b) (for $\sigma = 0.95$), Fig. 8(a) (for $\sigma = 0.9$), Fig. 8(b) (for $\sigma = 0.85$), Fig. 9(a) (for $\sigma = 0.8$) and Fig. 9(b) (for $\sigma = 0.75$). Regarding the effect of the different window sizes, 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 $W = 3$, 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 $W = 2$, the predictions are improved around 4% compared to $W = 8$ (Fig. 7(a)). Moreover, the larger the window, the smaller the improvement. 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. Note that the specific distance threshold values are selected with respect to the numbers of similar users they return; the experiment presents similar behavior for different such values. Regarding the effect of σ, the best recommendation quality is given when $\sigma = 0.9$. More specifically, the quality is improved from $\sigma = 1.0$ to $\sigma = 0.9$, while we notice the opposite behavior for smaller σ values. That is, for $\sigma \in [0.9, 1.0]$, the lower the σ, the better the predictions, and, for $\sigma < 0.9$, the higher the σ, the better the predictions. For example, for $W = 2$ and distance equal to 0.06, the predictions for $\sigma = 0.9$ are improved around 7% compared to the predictions for $\sigma = 1.0$ and 16% for $\sigma = 0.8$. The corresponding improvements for distance equal to 0.09 are 4.5% and 12%, respectively. In overall, our studies show that support has an effect on the recommendations quality and could be used for improving the recommendation process. However, the choice of an optimal value for σ to achieve the highest quality of recommendations is application dependent, due to the different amounts of ratings given at specific time instances, or periods.

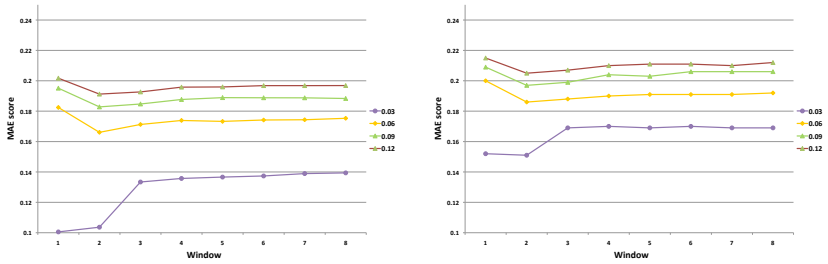
Damped window model. Next, we evaluate the effect of the decay rate λ in the recommendations accuracy. We use different values for λ; the higher the λ is, the less the historical data count. The value $\lambda = 0$ corresponds to the time-free model. We downgrade the original ratings based on the decay 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 for $\sigma = 0.9$, which is the optimal σ value according to the previous analysis, are shown in Fig. 10. This aging model offers a small improvement in this setting, i.e., for the employed dataset. In particular,



(a) (b)

Fig. 7. MAE scores for the sliding window model with (a) $\sigma = 1.0$ and (b) $\sigma = 0.95$.

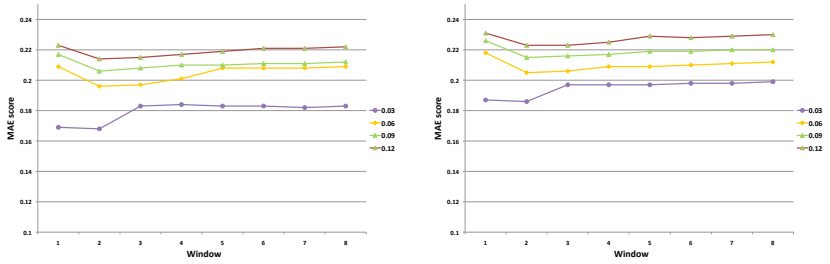


(a) (b)

Fig. 8. MAE scores for the sliding window model with (a) $\sigma = 0.9$ and (b) $\sigma = 0.85$.

the best MAE scores are obtained when $\lambda = 0.004$. So, for $\lambda = 0.004$ and distance equal to 0.03, the predictions are improved around 1.3% compared to the time-free model. The improvements for distance equal to 0.06, 0.09 and 0.12 are 0.6%, 1.6% and 1%, respectively. Larger λ values lead to worst predictions compared to the predictions of the time-free model. Practically, $\lambda = 0.004$ means that the ratings loose 10% of their value after 10 years. As above, larger distance thresholds lead to larger MAE scores and worst recommendations quality.

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



(a) (b)

Fig. 9. MAE scores for the sliding window model with (a) $\sigma = 0.8$ and (b) $\sigma = 0.75$.

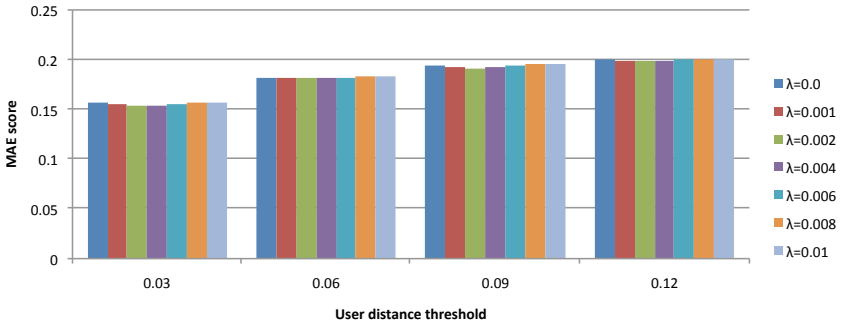


Fig. 10. MAE scores for the damped window model.

($D_{weekdays}$). The predicted values are compared to the actual values given by the user within the same temporal context through the MAE metric.

Fig. 11 displays the results for $\sigma = 0.9$. 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 13% 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 11%. Also, larger distance thresholds values result in larger MAE scores, that is, the quality of the recommendations decreases with the user distance threshold.

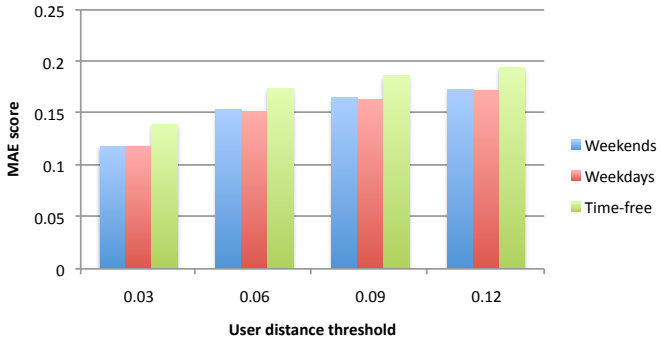


Fig. 11. MAE scores for the context-based approach.

Finally, we have performed t-tests to see if there are statistically significant differences between the proposed approaches and the time-free model. The results of the tests demonstrate that the probability of the difference being due to chance is less than 0.005, 0.0005 and 0.0005 for the sliding window, damped window and temporal context model, respectively. So clearly, our approaches produce statistically significant recommendations compared to the time-free model.

To summarize, 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. In our current work, we aim at designing 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.

7 Prototype Implementation: *Movie Guide*

To demonstrate the feasibility of our approach, we have developed a research prototype for a movie recommendation application, called tRecs: *A Time-aware Movie Guide* (Fig. 12). The overall system architecture of tRecs is the one depicted in Fig. 4. We maintain information about movies, users and ratings. The movies database schema consists of a single relation with schema: *Movies(mid, title, year, director, genre, language, duration)*. The prototype is implemented in Java and MySQL.

When a user joins the system, he/she registers his/her ratings for computing recommendations and his/her value and attribute preferences for constructing summaries and presenting the results (Fig. 13). Users express their ratings for movies by providing a numerical score between 0.0 and 1.0. Furthermore, users are allowed to define their value-based and attribute-based preferences following either the qualitative or the quantitative preference model. For instance, a user

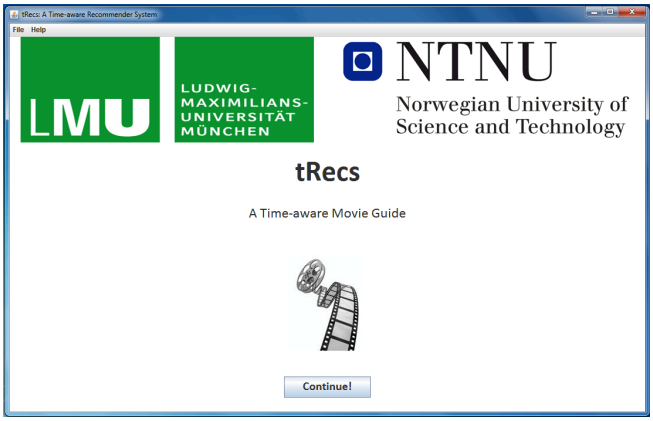


Fig. 12. tRecs: A Time-aware Movie Guide.

may define that *A. Hitchcock* $\succ$ *M. Scorsese* or may give the score 0.8 to *A. Hitchcock* and the score 0.6 to *M. Scorsese*. Similarly, for the attribute preferences. Clearly, a user can add, delete or modify ratings and preferences at any time and not only at registration time.

Besides user registration, the other part of the application includes recommendations computation, summary construction and presentation. Recommendations computation runs in two modes: time-free and time-aware. In the time-free mode, the temporal aspects of the user ratings are completely ignored. The time-aware mode distinguishes between fresh-based and context-based recommendations. In this mode, the user query is enhanced with some temporal information which is provided by the user as part of his/her query or postulated by the application. In the latter case, the information implicitly associated with the query corresponds to the current temporal characteristics, that is, the context at the time of the submission of the query. The user should also provide the exact scheme that will be used (damped window, sliding window or temporal context).

Presentation summaries are produced with respect to the top-100 time-aware recommendations. User preferences are employed for constructing ordered sets of keywords, in the form of a lattice. Following this ordering, keyword-based summaries, that is, keywords extended with movie titles and production years, are presented to users. As a case study scenario, suppose that a user gave two value-based preferences (*A. Hitchcock* is preferred over *M. Scorsese* and *horror* movies are preferred over *crime* movies) and one attribute-based preference (the *director* of a movie is as important as its *genre*). Suppose also that our user opts to follow the damped window model and would like to know the recommended movies according to his/her previously submitted ratings and preferences. The results of this query example are depicted in Fig. 14. Note that the summary for {*A. Hitchcock, crime*} does not appear in the results, since there are no *crime* movies directed by *A. Hitchcock* in our database instance.

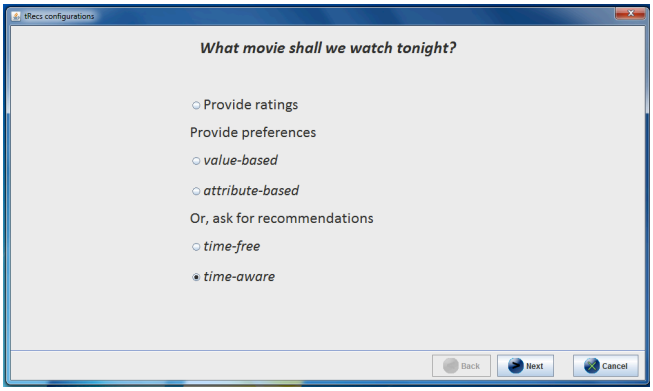


Fig. 13. tRecs configurations.

8 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., [33, 28]), *collaborative filtering*, that recommend items that users with similar preferences liked (e.g., [21, 11]) and *hybrid*, that combine content-based and collaborative ones (e.g., [8]). Several extensions have been proposed, such as employing multi-criteria ratings (e.g., [2]) and defining recommendations for groups (e.g., [7, 31, 30]).

Recently, there are also approaches focusing on enhancing recommendations with further contextual information (e.g., [3, 32]). 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. [37] presents a graph-based recommendation system that mixes long-term and short-term user preferences to improve predictions accuracy, while [36] considers how time can be used into matrix factorization models by examining changes in user and society tastes and habits, and items popularity. [15] 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. [10] 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

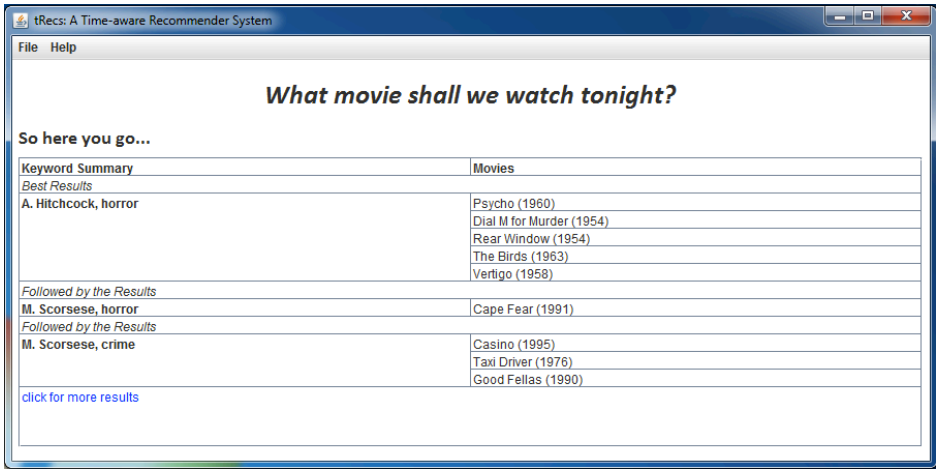


Fig. 14. tRecs: A Time-aware Movie Guide.

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.

The temporal aspect of the data has been also studied in different application domains like time series [29] and temporal database queries [14]. Recently the focus has been on huge amounts of data that are collected over time, the so called data streams [4, 18]. Due to the theoretically infinite nature of these data, it is impossible to consider them all for answering a query or for a data mining task. So, the rationale is to use the temporal information in order to "reduce" the dataset complexity, e.g., by focusing on a specific period of time instead of the whole stream (e.g., [5]) or by considering the aging of the data (e.g., [12]).

Finally, the general concept of summaries resembles the notion of *tag clouds*. A tag cloud is a visual representation for text data. Tags are usually single words, alphabetically listed and in different font size and color in order to show their importance[4]. Tag clouds have appeared on several Web sites, such as Flickr and del.icio.us, while recently tag cloud drawing has also received attention (e.g., [24]). With regard to summaries for keyword queries, *data clouds* [23] are the most relevant. This work proposes algorithms that try to discover good, not necessarily popular, keywords within the query results. Our approach follows a preference-based technique to locate important keywords. From a different perspective, [16] introduces the notion of *object summary* for summarizing the data in a relational database about a particular *data subject*, or keyword. An object summary is a tree with a tuple containing the keyword as the root node

[4] en.wikipedia.org/wiki/Tag_cloud

and its neighboring tuples containing additional information as child nodes. [17] extends this work by presenting a partial object summary of size l, composed of only l representative tuples.

9 Conclusions

In this paper, we study different semantics to exploit the time information associated with user ratings in order to improve the accuracy of recommendations. We consider various types of time effects, and thus, propose different time-aware recommendation models. Fresh-based recommendations care mainly for recent and novel ratings, while context-based recommendations are computed with respect to ratings with temporal context equal to the query context. To help users receive a broader view of the recommended items, we add some structure to the presentation of the results. In particular, we rank the recommended items based on user preferences and organize equally important items through summaries. Finally, we evaluate our approach using a real dataset of movie ratings and demonstrate its feasibility through a prototype implementation of a movie guide application.

There are several directions for future work. We envision 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.

Acknowledgments

The work of the second author is supported by the project "IdeaGarden" funded by the Seventh Framework Programme under grand n^o 318552.

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. C. Aggarwal, editor. *Data Streams – Models and Algorithms*. Springer, 2007.
5. C. C. Aggarwal, J. Han, J. Wang, and P. Yu. A framework for clustering evolving data streams. In *VLDB*, 2003.
6. R. Agrawal and E. L. Wimmers. A framework for expressing and combining preferences. In *SIGMOD Conference*, pages 297–306, 2000.
7. S. Amer-Yahia, S. B. Roy, A. Chawla, G. Das, and C. Yu. Group recommendation: Semantics and efficiency. *PVLDB*, 2(1):754–765, 2009.

8. M. Balabanovic and Y. Shoham. Content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72, 1997.
9. 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.
10. L. Baltrunas and X. Amatriain. Towards time-dependant recommendation based on implicit feedback. In *CARS*, pages 1–5, 2009.
11. J. S. Breese, D. Heckerman, and C. M. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI*, pages 43–52, 1998.
12. F. Cao, M. Ester, W. Qian, and A. Zhou. Density-based clustering over an evolving data stream with noise. In *SDM06*, 2006.
13. J. Chomicki. Preference formulas in relational queries. *ACM Trans. Database Syst.*, 28(4):427–466, 2003.
14. U. Dayal and G. T. J. Wuu. A uniform approach to processing temporal queries. In *VLDB*, pages 407–418, 1992.
15. Y. Ding and X. Li. Time weight collaborative filtering. In *CIKM*, pages 485–492, 2005.
16. G. J. Fakas. A novel keyword search paradigm in relational databases: Object summaries. *Data Knowl. Eng.*, 70(2):208–229, 2011.
17. G. J. Fakas, Z. Cai, and N. Mamoulis. Size-l object summaries for relational keyword search. *PVLDB*, 5(3):229–240, 2011.
18. J. Gama. *Knowledge Discovery from Data Streams*. CRC Press, 2010.
19. P. Georgiadis, I. Kapantaidakis, V. Christophides, E. M. Nguer, and N. Spyratos. Efficient rewriting algorithms for preference queries. In *ICDE*, pages 1101–1110, 2008.
20. W. Kießling. Foundations of preferences in database systems. In *VLDB*, pages 311–322, 2002.
21. 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.
22. G. Koutrika and Y. E. Ioannidis. Personalizing queries based on networks of composite preferences. *ACM Trans. Database Syst.*, 35(2), 2010.
23. G. Koutrika, Z. M. Zadeh, and H. Garcia-Molina. Data clouds: summarizing keyword search results over structured data. In *EDBT*, pages 391–402, 2009.
24. B. Y.-L. Kuo, T. Hentrich, B. M. Good, and M. D. Wilkinson. Tag clouds for summarizing web search results. In *WWW*, pages 1203–1204, 2007.
25. P. Melville and V. Sindhwani. Recommender systems. In *Encyclopedia of Machine Learning*, pages 829–838. 2010.
26. A. Miele, E. Quintarelli, and L. Tanca. A methodology for preference-based personalization of contextual data. In *EDBT*, pages 287–298, 2009.
27. B. Mobasher, R. Cooley, J. Srivastava, and J. Srivastava. Automatic personalization based on web usage mining. *Commun. ACM*, pages 142–151, 2000.
28. R. J. Mooney and L. Roy. Content-based book recommending using learning for text categorization. In *ACM DL*, pages 195–204, 2000.
29. Y. Nong. *The Handbook of Data Mining*. Mahwah, New Jersey, Lawrence Erlbaum Associates, 2003.
30. I. Ntoutsi, K. Stefanidis, K. Nørvåg, and H.-P. Kriegel. Fast group recommendations by applying user clustering. In *ER*, 2012.
31. 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.

32. 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.
33. M. J. Pazzani and D. Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27(3):313–331, 1997.
34. F. Ricci, L. Rokach, B. Shapira, and P. B. Kantor. *Recommender Systems Handbook*. Springer-Verlag New York, Inc., 2010.
35. K. Stefanidis, E. Pitoura, and P. Vassiliadis. Managing contextual preferences. *Inf. Syst.*, 36(8):1158–1180, 2011.
36. L. Xiang and Q. Yang. Time-dependent models in collaborative filtering based recommender system. In *Web Intelligence*, pages 450–457, 2009.
37. 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.