A Performance Comparison of Time-Aware Recommendation Models

Pedro G. Campos[1,2], Fernando Díez[1] and Iván Cantador[1]

[1] Universidad Autónoma de Madrid, 28049, Madrid, Spain
[2] Universidad del Bío-Bío, Avda. Collao 1202, Concepción, Chile
{pedro.campos, fernando.diez, ivan.cantador}@uam.es

Abstract. Recommender Systems (RS) suggest items to users without requesting them a explicit query, but exploiting their personal preferences expressed in the form of profiles. In these profiles, timestamped information allows identifying changes of user interests through time. Time-aware RS (TARS) aim to benefit from this information. However, most TARS approaches have been evaluated by using different, non-consistently agreed methodologies, metrics and datasets, making it difficult to compare them. Moreover, some of the followed evaluation protocols give TARS unfair advantages, e.g. by allowing temporal overlaps between training and test data. We propose to use an evaluation protocol based on real-world assumptions, which allows assessing TARS results improvements effectively. Being also interested in determining differences in TARS performance among recommendation tasks, we conducted experiments with several approaches on rating prediction and (ranking-oriented) top-N recommendation tasks, using for the latter metrics from the information retrieval field. The obtained results show that most of the tested approaches fail to take significant advantage under the proposed protocol, and that some TARS approaches have dissimilar performances across tasks.

Keywords: Time-Aware Recommender Systems, Time-Aware Evaluation, Collaborative Filtering

1 Introduction

Let us suppose we are searching for an "interesting" book in a specialized e-commerce site. Easily, we may have to revise tens of book descriptions in a collection with hundreds, if not thousands, of titles until finding a potential item of interest. Recommender Systems (RS) [1] aim to help us in this type of information filtering task, by suggesting items according to personal preferences. Differently from traditional Information Retrieval (IR) systems, RS are not provided with a explicit user query, but exploit *user profiles* with personal preferences, which can be considered as user relevance feedback about items. Having information about the users' profiles, and about item descriptions, a RS may suggest a particular user items similar to those she liked in the past (the

content based recommendation approach), or items that other people with preferences similar to the active user liked in the past (the *collaborative filtering* (CF) recommendation approach). In both cases, the current availability of temporarily contextualized profiles allows detecting fluctuations in user preferences through time, which may help improving recommendations. In fact, the million dollars *Netflix Prize* competition winning team claimed that the time-awareness of their solution was a key success factor for their results [9].

Different time-aware RS (TARS) have been proposed in the literature (e.g. [6, 10, 9]), improving results from base methods that do not use temporal information. However, most of these TARS have been evaluated by using different, non-consistently agreed methodologies, metrics and datasets, making it difficult to compare them. The absence of standardized protocols has even raised unrealistic evaluation scenarios for TARS. For instance, it is common to find evaluations in which there exist overlaps between training and test data with respect to time, and this gives TARS an advantage that may not exist in a real world scenario. Additionally, most approaches have focused on attempting to reduce the error between known and predicted ratings (*rating prediction task*). However, nowadays the ranking of predicted relative user preferences (*top-N recommendation task*) is gaining attention. From the above, we identify two research questions to address: (**RQ1**) Can TARS effectively improve recommendation results using a realistic evaluation scenario? And, (**RQ2**) do TARS behave consistently for the prediction and ranking recommendation tasks?

In order to address these questions, we conducted evaluation experiments with different state-of-the-art time-aware extensions of CF RS using a common evaluation protocol and dataset. We aimed to discard unrealistic characteristics, as the aforementioned temporal overlap, thus helping to provide an argumented answer to RQ1. We evaluated the recommendation approaches on both the rating prediction and the top-N recommendation tasks. For the latter, we used well-known metrics from the IR field, such as Precision, Recall and nDCG, in order to appropriately assess differences of relative performance between algorithms, as posed in RQ2. Putting into practice realistic assumptions regarding time-aware training/test data splitting, we show that most of the prediction-based approaches fail to take significant advantage under the tested protocol. Regarding the evaluation on the different recommendation tasks, we identified performance differences for some approaches that may be caused by the distinct nature of the tasks.

The remainder of the paper is structured as follows. Section 2 presents a brief review of related work. Section 3 states the time-aware recommendation problem, and describes the tested time-aware models. Section 4 describes the used dataset and applied evaluation protocol, and discusses the obtained results. We end with some conclusions and devised future work in Section 5.

2 Related Work

In the last years, there has been an increasing interest in TARS. There are approaches built upon the assumption that recent preferences better reflect the user's current interests, and thus overweight these preferences in the recommendation process. Hence, *time decaying* has given raise to many variants of

time-aware recommendation algorithms [6, 11]. Other approaches, on the contrary, are based on the idea that older information is useful and should not be underweighted, posing models that incorporate temporal parameters reflecting fluctuations in user preferences over time [9]. There are also approaches to explicitly model the users' concept drift [12], and to identify repetitive patterns through time [2]. Finally, another type of approaches proposes dynamical adaptations through time of (originally static) model parameters [10]. In this context, it is important to note that most of the existing works on the topic have been developed in the movie recommendation domain, and have only addressed the rating prediction problem.

The topic of evaluation is also gaining increasing attention in the RS research community [7]. Traditionally, recommendation results in offline experimentation have been assessed by measuring the error on rating predictions from known rating values [8, 1], due to the focus on obtaining *accurate* predictions. However, nowadays, there is a main interest in generating *useful* recommendations, thus requiring the definition of new metrics to take into account, for example, the *relative ordering* of user preferences. With this purpose, metrics from the IR field, such as Precision and Recall, have been proposed to assess the utility of recommendation lists [7]. Nevertheless, differences between RS and IR systems had lead to different interpretations of how to apply these metrics [4]. We note that it is difficult to find equivalent comparison parameters among different authors' proposals. Regarding TARS, given that user preferences may change along time, their evaluation should also take into consideration the dynamic nature of the time-aware recommendation problem [10]. Nonetheless, many times this issue has not been tackled carefully, and it has been argued that sloppy evaluation protocols may give advantages to TARS result measurements –not possible in a real-world evaluation– e.g. admitting temporal overlaps between training and test data [5].

It is also important to note that any system helping users to select items in a personalized fashion can be considered as a RS. In this sense, RS could perform different tasks. In this work, we address two important recommendation tasks, known as rating prediction and top-N recommendation. In the rating prediction task, a RS is required to provide an estimation of the rating value for each item presented to the user, which indicates the user's potential interest in such item. Although the user is not presented with an explicit recommendation (the system does not explicitly invite the user to select a particular item), highly rated items should be considered to be recommended [7]. Much of the research in the RS field had focused on improving results particularly on this task, e.g. [6, 9, 12, 10, 3].

Top-N -or ranked- recommendation task, on the other hand, is probably the par excellence recommendation task. It requires a RS to provide the user a ranked list of relevant items (this is also referred as *recommending good items* [8, 7]). This list can be shortened to a fixed length (assuming the user does not have enough time or resources as to traverse the full list), and ordered according to the degree of expected utility. In the past this task was partially neglected by researchers. However, it has become more important in the last years, partly due to the realization that, the quality of ranked recommendation lists can be more valuable for users than obtaining accurate rating predictions, in order to

select items to use [4]. It has been argued that this task is different in its nature from the rating prediction task. Firstly, it focus on relevance (instead of on accuracy). Secondly, higher ratings are not necessarily related with relevance, a basic premise in the case of rating prediction task. Despite this, it is common to use accuracy-oriented algorithms for producing recommendation lists, by means of generating rating predictions for all items, and then ordering them according to the predicted rating values.

3 Time Aware Recommendation

Traditionally, the recommendation problem has been formulated as the estimation of ratings for items that have not been used[3] by a user. In this section, we formalize the recommendation tasks addressed in this work, and their time-aware extensions, and describe the evaluated models.

3.1 Problem Statement

Consider a utility function measuring the usefulness of an item for a user. Let $\mathcal{U}$ be the set of users, and let $\mathcal{I}$ be the set of items in a system. Let $\mathcal{F}$ be the utility function $\mathcal{F} : \mathcal{U} \times \mathcal{I} \to \mathcal{R}$, where $\mathcal{R}$ is a totally ordered set ($\mathcal{R}$ is usually but not necessarily a subset of $\mathbb{R}$).

Definition 1. *The rating prediction problem consists of estimating $\mathcal{F}$ for each u in $\mathcal{U}$ and each i in $\mathcal{I}$, whose utility is unknown a-priori, i.e.:*

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

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

Conceptually, if the function $\mathcal{F}$ can be computed for all the pairs in $\mathcal{U} \times \mathcal{I}$ domain, a RS may select for recommendation the item with highest utility for each user [1]:

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

This definition can be extended to the task of generating a ranked *top-N* recommendation list, i.e., selecting for recommendation an ordered set consisting of the N items with highest utility.

Definition 2. *The top-N recommendation task consists of determining for each user u the set of items $\mathcal{I}_N^*(u)$ such that (considering $\mathcal{I}_0^*(u) = \emptyset$):*

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

[3] Here *used* involves different senses or activities like, for example, seen, browsed, tasted and purchased, depending on the type of recommended items.

In CF, the main source of information for selecting items in $\mathcal{I}_N^*(\cdot)$ is the set of known utility values or ratings, i.e. the so called rating matrix R. This matrix can be enriched with additional information. As noted in [1], *contextual* information about *where* and *when* the ratings were provided can be also used. In this work, we use *temporal* context information associated with the rating creation time. This information could be exploited to detect changes in the users' tastes through time, in order to differentiate which items would be best rated by users depending on the request time. This way we can redefine the utility function to make it *time-dependent*: $\mathcal{F} : \mathcal{U} \times \mathcal{I} \times \mathcal{T} \to \mathcal{R}$, where $\mathcal{T}$ denotes the set of *timestamps* associated to ratings. Using this particular utility function, we also redefine the recommendation problem tasks stated before:

Definition 3. *The time-dependent rating prediction problem consists of estimating the utility of items for users at particular times:*

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

Similarly, the time-dependent top-N recommendation problem consists of recommending the highest utility items for users at a particular time ($\mathcal{I}_0^*(u,t) = \emptyset$):

$$\forall u \in \mathcal{U}, t \in \mathcal{T}, \mathcal{I}_N^*(u,t) = \bigcup_{j=1}^{N} i_j^*(u,t) : i_j^*(u,t) = \underset{i \in \mathcal{I} - \mathcal{I}_{j-1}^*(u,t)}{\arg\max} \mathcal{F}(u,i,t)$$

From the above, a TARS is a RS which is able to exploit time context information in order to perform a recommendation task.

3.2 Time-Aware Recommendation Models

In our study, we selected three families of CF algorithms, based on their good reported results, popularity in the field, and availability of time-aware extensions. In the following, we describe these three families, and the time-aware extensions proposed for each of them.

k-Nearest Neighbors. The k-Nearest Neighbors (kNN) model has been a base algorithm for many TARS approaches. Its *user-based* variant estimates $\mathcal{F}$ from the set of users (*nearest neighbors*) most similar to the target user. It extrapolates the rating the target user would give to a particular item, by using a user similarity measure as a weighting factor. The ratings given by the target user's neighbors to the item are then weighted in the following way (the *item-based* variant can be computed analogously, using the set of items similar to the target item):

$$\hat{r}_{u,i} = b \sum_{u' \in N_k(u)} sim(u, u') \cdot r_{u',i} \tag{1}$$

In (1), b is a normalization factor, computed as $b = 1/\sum_{u' \in N_k(u)} sim(u, u')$, where $sim(u, u')$ is the similarity value between users u and u', commonly computed as the correlation among co-ratings, or Pearson Correlation:

$$sim(u,u') = \frac{\sum_{i \in \mathcal{I}_{u,u'}} (r_{u,i} - \overline{r}_u)(r_{u',i} - \overline{r}_{u'})}{\sqrt{\sum_{i \in \mathcal{I}_{u,u'}} (r_{u,i} - \overline{r}_u)^2 \sum_{i \in \mathcal{I}_{u,u'}} (r_{u',i} - \overline{r}_{u'})^2}}$$

where $\mathcal{I}_{u,u'} = \{i \in \mathcal{I} : r_{u,i} \neq \emptyset \wedge r_{u',i} \neq \emptyset\}$, i.e., items rated by both users. The set $N_k(u)$ represents the k nearest neighbors of u computed as (with $N_0 = \emptyset$):

$$N_k(u) = \bigcup_{j=1}^{k} u'_j : u'_j = \underset{u' \in \mathcal{U} - N_{j-1}(u), u \neq u'}{\arg\max} sim(u,u')$$

Improvements to this method have been proposed e.g. by penalizing similarities computed on few common ratings, i.e., replacing $sim(\cdot,\cdot)$ by $sim'(\cdot,\cdot) = b' sim(\cdot,\cdot)$ being $b' = |\mathcal{I}_{u,u'}|/\tau_{common\_ratings}$ if $|\mathcal{I}_{u,u'}| \leq \tau_{common\_ratings}$.

Time Weight. Ding and Li [6] modified the common rating prediction computation used in kNN (eq. 1) incorporating a time weighting factor $w(m)$:

$$\mathcal{F}(u,i,t) = b \sum_{u' \in N(u)} sim(u,u') \cdot w(\mathcal{D}(r_{u',i},t)) \cdot r_{u',i}$$

where $\mathcal{D}(r,t)$ returns the amount of days between r and t, and $w(m) = e^{-\delta \cdot m}$ is the time weighting function, δ is the *decay rate*, set to $\delta = 1/M_0$. M_0 is known as the *half-life* of $w(m)$, a value such that $w(M_0) = (1/2)w(0)$. That is, the weight reduces by $1/2$ in M_0 days. This way, older ratings are underweighted. Note that the normalization factor b must be recomputed accordingly.

Adaptive kNN. In [10], the authors propose a method to automatically update per-user neighborhood sizes, attempting to outperform a global, static size k under the TA_RMSE metric, a variation of RMSE measured at different times such that $TA\_RMSE(t) = (\sum_{r_{u,i} \in TestSet_t} (\hat{r}_{u,i})/|TestSet_t|)^{\frac{1}{2}}$ where $TestSet_t$ is the set of test ratings made up to time t. The k values are computed by:

$$\forall u \in \mathcal{U} : k_{u,t+1} = \max_{k \in V} (error_{u,t} - TA\_RMSE_{u,t,k}) \quad (2)$$

where $k_{u,t+1}$ is the k value selected for predicting ratings of u in the time interval $[t, t+1]$, V is a set of potential k values to be tested, $error_{u,t}$ is the TA_RMSE achieved until time t between the ratings in the user's profile and predictions made with the k values selected before t, and $TA\_RMSE_{u,t,k}$ being the TA_RMSE on the user's profile that would be achieved with parameter value k. Thus, (2) selects the parameter value k that maximizes the improvement on the current user's error. We used $V = \{0, 25, 50, ..., 250, 275, 300\}$, and results from a Bias Model [9] when $k = 0$ was selected, similarly as in [10].

AutoSimilarity in Time. Min and Han [12] proposed to treat timestamped rating data as time series (TS), by using a (user, item category) pair basis rating pattern. Each item is assigned into a category, and a temporal rating pattern for each user on each item category[4] is obtained, by dividing rating data into

[4] A simple categorization scheme in the movies domain is to relate each movie with its genre (used here), though other schemes may be used.

different time intervals, and then computing a *categorical* rating of u on the item category c in each time interval t_i by $r_{u,c,t_i} = \frac{\sum_{i \in c} r_{u,i,t_i}}{|r_{u,i,t_i}|}$. Once the TS is obtained, different methods of TS analysis can be used. In this work, we follow the one proposed in [12], where the moment when a *concept drift* occurs is identified based on the user's auto similarity, that is, by analyzing how similar are the categorical ratings between different time intervals. For such purpose, we use the Pearson correlation coefficient on category ratings of the same user, but on different time intervals:

$$AS(u, t_i, t_j) = \frac{\sum_c (r_{u,c,t_i} - \hat{r}_{u,t_i})(r_{u,c,t_j} - \hat{r}_{u,t_j})}{\sum_c (r_{u,c,t_i} - \hat{r}_{u,t_i})^2 \sum_c (r_{u,c,t_j} - \hat{r}_{u,t_j})^2}$$

where $AS(u, t_i, t_j)$ is the auto similarity of u between time intervals t_i and t_j, and $\hat{r}_{u,t}$ is the mean category rating of u on all categories during time interval t. Given a similarity threshold τ_{sim}, if $AS(u, t_i, t_j) < \tau_{sim}$ then it is concluded that u changed her tastes on time interval t_j. If no change is detected, ratings may be predicted using e.g. (1). If a change is detected, the similarity computation is modified using differentiated weights according to rating time. Let $\mathcal{I}_{-t_j}$ and $\mathcal{I}_{+t_j}$ be the set of items rated by u before and after u's concept drift, respectively (which is considered to occur at time t_j), and w_{-t_j} and w_{+t_j} be the weights assigned accordingly (assigned with values 1 and $1+0.5|AS(u, t_{j-1}, t_j)|$ as in [12]). Then:

$$sim'(u, v) = b \left(\sum_{i \in \mathcal{I}_{-t_j}} (r_{u,i} - \hat{r}_u)(r_{v,i} - \hat{r}_v) w_{-t_j}(u) + \sum_{i \in \mathcal{I}_{+t_j}} (r_{u,i} - \hat{r}_u)(r_{v,i} - \hat{r}_v) w_{+t_j}(u) \right)$$

with $b = \sum_{i \in \mathcal{I}} (r_{u,i} - \hat{r}_u)^2 \sum_{i \in \mathcal{I}} (r_{v,i} - \hat{r}_v)^2$. Predictions are computed using (1), changing $sim(\cdot, \cdot)$ by $sim'(\cdot, \cdot)$. Note that for similarity computation, the individual item ratings are used instead of category ratings.

Bias Model. It is known that some users have systematic tendencies to give higher (or lower) ratings than others [9]. A bias model incorporates such information with the purpose of discounting such user (and item) effects; used in conjunction with other modeling techniques, can lead to more accurate predictions. A basic bias-aware estimator is [9]:

$$\hat{r}_{u,i} = \mu + b_u + b_i$$

where μ stands for the global average rating, b_u is the average rating bias of user u, and b_i is the average rating bias of item i.

Time-Aware Bias Model. A natural next step when considering time effects is to incorporate *time-changing* bias effects:

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

Following Koren's analysis [9], the item temporal bias is expected to change slowly over time, meanwhile the user temporal bias also includes sudden, day-specific drifts. Slow bias changes can be modeled either as a *temporal binning* of the bias (i.e., a different bias is computed on different time intervals or bins), or as a decaying function of time. In this work, we adhere to Koren's formulation, which also maintains a bias term whose value holds during all timespans, thus getting:

$$b_u(t) = b_u + \alpha_u \cdot sign(t - t_u)|t - t_u|^{\beta} + b_{u,t}$$
$$b_i(t) = b_i + b_{i,Bin(t)}$$

where t is the prediction date, t_u is the mean rating date of u, and α_u and β are parameters learned from data.

Matrix Factorization. Matrix Factorization (MF) is an extension of Singular Value Decomposition in which R is iteratively approximated by user and item factor matrices P and Q of lower dimension (f in our notation) such that:

$$\hat{r}_{u,i} = \sum_{j=0}^{f} P_{u,j} \cdot Q_{j,i} = p_u^T q_i$$

One advantage of this approach is that P and Q values may be computed for all users and items using only the known values in R, by minimizing an estimation of the difference, e.g. the Frobenius Norm: $\min \|R - PQ\|^2$. Overfitting can be alleviated using regularization, i.e., penalizing the magnitude of P and Q [9].

Time-Aware Matrix Factorization. When considering the incorporation of time effects in a Time-Aware MF (TA MF) model, Koren includes static and dynamic biases. Additionally, he highlights that users are due to changes in their tastes, so factors describing their rating behavior are more likely to be prone to temporal effects [9]. Thus, a modeling similar to those used on the user bias effects can be applied on the users' factors, leading to:

$$\hat{r}_{u,i} = \mu + b_u(t) + b_i(t) + p_u^T(t) q_i$$

with

$$p_{u,j}(t) = p_{u,j} + \alpha'_{u,j} \cdot sign(t - t_u)|t - t_u|^{\beta'} + p_{u,j,t}$$

The associated minimization problem must consider regularization terms, and can be solved using the stochastic gradient descent method [9]. Note that the results are sensitive to optimization parameter values (essentially learning rates and regularization values); we used the Simplex optimization method in order to seek for good parameter values.

4 Experiments

4.1 Dataset Description

In our experiments, we used the $MovieLens1M$ dataset[5], comprising 1,000,209 timestamped ratings on a 1-5 scale from 6,040 users on 3,706 movies. This dataset corresponds to users of the $MovieLens$ RS who joined the system in 2000. Rating timestamps span from April 26^{th}, 2000 to February 28^{th}, 2003. An interesting characteristic of the dataset is that many users made most of their ratings just after joining the system, as may be expected from the fact that initially users rate items in order to feed the system with their interests.

4.2 Evaluation Protocol

We selected the 500 users with longest rating timespan (in order to detect changes through time), who were grouped randomly into 5 subsets of 100 users each. We formed splits with 4 out of 5 of these subsets, thus obtaining 5 different overlapping samples of 400 users. We performed a time-aware training/test data split on each sample, using as test starting date (t_{test}) June 15^{th}, 2001. This condition discards any temporal overlapping among training an test data, assigning roughly 20% of ratings for test. We obtained rating predictions and recommendation lists (ranked according to the rating prediction value) for each sample and recommendation model.

We assessed the rating prediction error computing MAE and RMSE values [8]. For evaluating the quality of recommendation lists, we used *Precision* and *Recall* at cutoffs 5 and 10, and nDCG. The application of these metrics for RS evaluation has been uneven in the literature [4], particularly on what items are to be considered as eligible for the recommendation list (e.g. all items or items known by the user). In our case, as a trade-off between completeness and efficiency, we computed predictions for all items in the test set of any user (except those in the target user's training set). Those items rated equal or higher than 3.0 in the test set of the target user were consider as relevant items for metric computations. As noted in [4], independently from their absolute values, the usage of a common criterium for metric computation allows a fairly comparison of the different RS models.

The results were averaged among the 5 splits. Table 1 shows the parameter values used in the implementation of the models. In the experiments, we also evaluated two non-personalized recommenders: 1) a *popularity*-based recommender, which normalizes the popularity of an item on the rating scale as rating prediction, and puts in the top places of the recommendation list the most popular items (intended for the top-N prediction task); and 2) a random recommender.

4.3 Results

Table 2 presents the average results obtained for each model in both tasks. Regarding the rating prediction task, the best performing algorithm on MAE

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

Table 1. Parameter values

Model	Param.	Value
kNN	k	200 items
	$\tau_{common\_ratings}$	50
Time Weight	M_0	250 days
Adaptive kNN	time interval size	7 days
Auto Similarity	τ_{sim}	-0.1
TA Bias	number of temporal bins	4
MF	f	10 factors
TA MF	f	10 factors
	number of temporal bins	4

and RMSE is (non-TA) MF , followed by the Time Weight model. This reflects that, being MF a reliable rating prediction model [9], the modeling of temporal dynamics does not improve prediction error under the considered evaluation protocol. It should be noted that some terms in the TA MF model need training ratings dated with the date at which the prediction is required (i.e., they depend on the temporal overlapping of training and test data) [5]. This could in part explain the observed behavior (note that this model was developed in the context of the Netflix Prize competition, where this was valid). On the contrary, the TA Bias model, which does not require temporal overlapping, shows improvements from its base model on both metrics.

In the case of kNN variants, we can observe that the only TA extension that shows improvements from basic kNN is the Time Weight model. In any case, the TARS results obtained are not statistically different from non-TA models, except in the case of non-personalized models. Different factors may affect the potential of TARS models to improve results over non-TA models in this case, e.g. user and item characteristics, size of training data (note that the evaluated models were initially developed and used on different datasets and circumstances). However, we believe that an important and generic factor is the usage of a strict evaluation protocol, which avoids temporal overlapping of training and test data. This disable TA models to take advantage of knowledge about ratings made in the same (or later) time interval of the required prediction, and better reflects the real-world setting in which a deployed RS must perform.

With respect to top-N recommendation task, poor results were obtained in general. In fact, the non-personalized popularity-based recommender performs the best on all metrics considered for the task. Anyhow, when analyzing only the personalized CF models, the MF model is the best performing, nearly followed by the TA Bias and TA MF models respectively. Again, we observe that the TA MF model is outperformed by their non-TA counterpart (except on P@10). Likewise, the TA Bias model is again able to outperform its non-TA counterpart. In the case of kNN variants, Adaptive kNN and Autosimilarity models present better Precision values than kNN model, but lower Recall and nDCG values. It is notorious that the worst performing model in this task is the Time Weight model (even worst than Random), moreover at the light of the results on the rating prediction task.

Table 2. Results on Rating Prediction and Top-N recommendation tasks (mean values on 5 splits). Statistical significant differences (Wilcoxon $p < 0.01$) of TARS models are indicated with respect to Random(†), Popularity(‡), kNN(§), Bias(△) and MF(◊) models respectively. Bold indicates best column values.

	Rating Prediction		Top-N Recommendation				
Model	MAE	RMSE	P@5	P@10	R@5	R@10	nDCG
Random	2.4665	2.6918	0.0171	0.0186	0.0018	0.0037	0.3375
Popularity	2.4649	2.6906	**0.2209**	**0.1993**	**0.0280**	**0.0525**	**0.4733**
kNN	0.6963	0.8943	0.0360	0.0386	0.0058	0.0115	0.3822
Bias	0.7054	0.9019	0.0658	0.0737	0.0093	0.0202	0.3914
MF	**0.6925**	**0.8842**	0.0931	0.0858	0.0129	0.0242	0.4057
Time Weight	$0.6951^{\dagger\ddagger}$	$0.8929^{\dagger\ddagger}$	$0.0028^{\dagger\ddagger\S\triangle\diamond}$	$0.0031^{\dagger\ddagger\S\triangle\diamond}$	$0.0002^{\dagger\ddagger\S\triangle\diamond}$	$0.0005^{\dagger\ddagger\S\triangle\diamond}$	$0.3495^{\dagger\ddagger\S\triangle\diamond}$
Adaptive kNN	$0.7010^{\dagger\ddagger\diamond}$	$0.9009^{\dagger\ddagger\S}$	$0.0369^{\ddagger\triangle\diamond}$	$0.0399^{\dagger\ddagger\triangle\diamond}$	$0.0061^{\ddagger\diamond}$	$0.0116^{\ddagger\triangle\diamond}$	$0.3813^{\dagger\ddagger\triangle\diamond}$
Auto Similarity	$0.6965^{\dagger\ddagger}$	$0.8946^{\dagger\ddagger}$	$0.0364^{\ddagger\triangle\diamond}$	$0.0394^{\dagger\ddagger\triangle\diamond}$	$0.0055^{\ddagger\diamond}$	$0.0117^{\dagger\ddagger\triangle\diamond}$	$0.3815^{\dagger\ddagger\triangle\diamond}$
TA Bias	$0.6986^{\dagger\ddagger}$	$0.8928^{\dagger\ddagger}$	$0.0873^{\dagger\ddagger\S}$	$0.0891^{\dagger\ddagger\S}$	$0.0119^{\dagger\ddagger\S}$	$0.0222^{\dagger\ddagger\S\diamond}$	$0.4021^{\dagger\ddagger\S\triangle}$
TA MF	$0.6993^{\dagger\ddagger}$	$0.8934^{\dagger\ddagger}$	$0.0824^{\dagger\ddagger\S}$	$0.0875^{\dagger\ddagger\S}$	$0.0115^{\dagger\ddagger}$	$0.0234^{\dagger\ddagger\S}$	$0.3998^{\dagger\ddagger\S\triangle}$

5 Conclusions and Future Work

In this paper, we have presented an empirical comparison of several TA extensions of CF models evaluated in two recommendation tasks –rating prediction and top-N recommendation– using a common evaluation protocol that discards temporal overlapping between training and test data. The obtained results reveal that some of the TA extensions considered are not able to perform better than their non-TA counterparts under the followed protocol. This is an important conclusion indicating that more realistic evaluation scenarios lower the performance improvement ability of TARS. This provides a preliminary answer to our first research question **RQ1**. Nonetheless, a deeper analysis is required on the generalization of these results to other models, and on the suitability of the evaluation scheme. For example, results appear to indicate that, in general, the idea of generating recommendation lists based on predicting item ratings may not be a good choice, given that a popularity-based model is able to beat with statistical significance all CF models. Despite this result, we note that popularity-based recommendations are non-personalized, and thus may be of low value for RS users. Further research on better evaluation procedures and metrics for top-N recommendation task is thus required. All in all, the absence of statistical significant differences among TARS and basic CF models with this evaluation protocol should encourage the development of TARS models with more strict consideration of real-world conditions.

The obtained results also show that some TARS are not consistent across the above recommendation tasks (**RQ2**). The most clear example is the Time Weight model. However, we showed that a simple model like Bias enhanced with temporal information is able to outperform its non-TA counterpart on both tasks. A general remark may be that simpler models can be more easily enhanced with temporal information, whilst other more robust methods as MF require more careful extensions to achieve improvements when real-world scenarios are considered. We believe that further research is required to better explain these results.

Furthermore, an standardized (and TA) evaluation protocol should be adopted to allow fairly comparisons among models. Possible lines of future work aim at the performance of a deeper study on how different evaluation parameters affect metric results (e.g. ratings/users/items distribution, training/test splitting), and the development of specific TA models for top-N recommendation.

6 Acknowledgments

This research was supported by the Spanish Government (TIN2011-28538-C02-01) and by the Comunidad de Madrid and the Universidad Autónoma de Madrid (CCG10-UAM/TIC-5877). The authors thank Centro de Computación Científica at UAM for its technical support.

References

1. Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge and Data Engineering 17(6), 734–749 (2005).
2. Baltrunas, L., Amatriain, X.: Towards time-dependant recommendation based on implicit feedback. In: Proc. RecSys 2009 Workshop on Context-aware Recommender Systems (2009).
3. Bell, R.M., Bennett, J., Koren, Y., Volinsky, C.: The million dollar programming prize. IEEE Spectrum 46(5), 28–33 (2009).
4. Bellogín, A., Castells, P., Cantador, I.: Precision-based evaluation of recommender systems: An algorithmic comparison. In: Proc. RecSys 2011, pp. 333–336 (2011).
5. Campos, P.G., Díez, F., Sánchez-Montañés, M.: Towards a more realistic evaluation: Testing the ability to predict future tastes of matrix factorization-based recommenders. In: Proc. RecSys 2011, pp. 309–312 (2011).
6. Ding, Y., Li, X.: Time weight collaborative filtering. In: Proc. CIKM 2005, pp. 485–492 (2005).
7. Gunawardana, A., Shani, G.: A survey of accuracy evaluation metrics of recommendation tasks. The Journal of Machine Learning Research 10, 2935–2962 (2009).
8. Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.T.: Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems 22(1), 5–53 (2004).
9. Koren, Y.: Collaborative filtering with temporal dynamics. In: Proc. KDD 2009, pp. 447–456 (2009).
10. Lathia, N., Hailes, S., Capra, L.: Temporal collaborative filtering with adaptive neighbourhoods. In: Proc. SIGIR 2009, pp. 796–797 (2009).
11. Lee, T.Q., Park, Y., Park, Y.T.: An empirical study on effectiveness of temporal information as implicit ratings. Expert Systems with Applications 36(2), 1315–1321 (2009).
12. Min, S.H., Han, I.: Detection of the customer time-variant pattern for improving recommender systems. Expert Systems with Applications 28(2), 189–199 (2005).