Towards a More Realistic Evaluation: Testing the Ability to Predict Future Tastes of Matrix Factorization-based Recommenders

Pedro G. Campos[1,2], Fernando Díez[1], Manuel Sánchez-Montañés[1]
{pedro.campos, fernando.diez, manuel.smontanes}@uam.es

[1]Universidad Autónoma de Madrid
Francisco Tomás y Valiente 11
28049, Madrid, Spain

[2]Universidad del Bío-Bío
Av. Collao 1202
4081112, Concepción, Chile

ABSTRACT

The use of temporal dynamic terms in Matrix Factorization (MF) models of recommendation have been proposed as a means to obtain better accuracy in rating prediction task. However, the way such models have been tested may not be a realistic setting for recommendation. In this paper, we evaluated rating prediction and top-N recommendation tasks using a MF model with and without temporal dynamic terms under two evaluation settings. Our experiments show that the addition of dynamic parameters do not necessarily yield to better results on these tasks when a more strict time-aware separation of train/test data is performed, and moreover, results may vary notably when different evaluation schemes are used.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information Filtering, Retrieval Models, Selection Process*; I.5.1 [**Pattern recognition**]: Models

General Terms

Algorithms, Performance

Keywords

Time-Aware Recommender Systems, Matrix Factorization, Time-Aware Evaluation

1. INTRODUCTION

Given that many Recommender Systems (RS) have been operating for years, the temporal dimension is acquiring more importance. One example of temporal variations is the change of tastes (evolution) of users through time. Many new algorithms try to incorporate this information e.g. [2, 6, 4]. Moreover, the *Netflix Prize*[1] competition wining team has mentioned the time-awareness of their solution as one of the key factors of their successful ensemble [4]. Despite the above mentioned, is notable that some evaluation settings used to test recommendation methods are not as realistic as should be expected. For example, in the Netflix Prize evaluation scheme, there are test data time-stamped before some training data. That motivated us to test the performance of time-aware recommendation models when using a more strict (realistic) setting. Given the outstanding performance of Koren's Time-Aware Matrix Factorization algorithm [4], we tested if it really could improve future ratings' prediction when no "future" data are available. Our intuition was that, given the bunch of additional information the algorithm uses, there is too much over-fitting to train data and thus prediction of future ratings (an important purpose of a RS) would not be as desirable with this model. We hypothesize that a simpler, less prone to over-fitting model can produce better predictions of future ratings. With this purpose, we have made a comparative of algorithms' performance under commonly used vs. a more strict experimental setting [7]. In section 2 we describe the related model and the evaluation scheme used previously to test it. In section 3 we present and analyze the results obtained. Finally we present some preliminary conclusions and possible future work.

2. RELATED WORK

The first time-aware RS were mainly based on models dependent linearly or exponentially (a.k.a. *time decay*) on the *time distance* to past ratings (e.g., days ellapsed) [2]. These models work under the assumption that more recent ratings better reflects users' present tastes. Most of these previous models work under a *kNN* framework, where some weight computations (e.g. similarity) are modified by the time distance. However as Koren states [4], there are punctual fluctuations, e.g. a bad mood day where the user finds everything worse than on a normal day, or longer-lasting but any case time-bounded effects, such as temporal item trends. He developed a model mixing these considerations, which may be incorporated into a *kNN* or a Matrix Factorization (MF) framework. Though the former have been the most used framework in the Collaborative Filtering (CF) field due to its simplicity and good performance, the latter is gaining increasing attention because of its superior performance. In this work we centered on the MF framework.

2.1 Time-Aware Matrix Factorization Models

2.1.1 Matrix Factorization Models

The MF technique is an extension of the Singular Value Decomposition (SVD) approach. In SVD a data matrix R is decomposed into new matrices such that $R = P\Sigma Q^T$. Given that typically the rating matrix R is sparse, and thus SVD can not be applied directly, in the case of MF, P and Q are used to iteratively approximate R using that [6]:

$$\hat{r}_{u,i} = \sum_{j=0}^{f} P_{u,j} \cdot Q_{j,i} = p_u^T q_i \tag{1}$$

This way R can be approximated (factorized) by P and Q (f user and item factors respectively) using only the known values of R, minimizing for example the Frobenius Norm between the difference on them: $\min \|R - PQ\|^2$. Overfitting can be alleviated using regularization, i.e., penalizing the magnitude of the approximated vectors [5].

2.1.2 Time Modeling

In his KDD 2009[1] Best Research Paper [4], Koren describes the incorporation of biases, temporal biases and temporal user/item interaction factors into a MF model. The purpose of incorporating bias terms into the model is to discount such user and item effects thus obtaining a more accurate prediction. A basic bias-aware estimator is [5]:

$$\hat{r}_{u,i} = \mu + b_u + b_i \tag{2}$$

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. A natural next step when considering time effects is to incorporate *time-changing* bias effects [4]:

$$\hat{r}_{u,i} = \mu + b_u(t) + b_i(t) \tag{3}$$

Following Koren's analysis [4] 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 by a *temporal binning* of the bias (i.e. a different bias is computed on different time intervals or bins) or a decaying function of time. Here we adhere to Koren's formulation which also maintains a bias term whose value holds during all timespan, thus getting [4]:

$$\begin{aligned} b_u(t) &= b_u + \alpha_u \cdot sign(t - t_u)|t - t_u|^{\beta} + b_{u,t} \\ b_i(t) &= b_i + b_{i,Bin(t)} \end{aligned} \tag{4}$$

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

2.1.3 Time-Aware Matrix Factorization

When considering the incorporation of time effects in a MF model, besides static and dynamic biases, Koren also discusses how temporal dynamic affects user-item interactions. He highlights that users are due to personal changes in their tastes, so factors describing their rating behavior are more likely to be prone to temporal effects than factors related with items. Thus, a modeling similar to those used on the user bias effects can be applied on the users' factors, leading to [4]:

$$\hat{r}_{u,i} = \mu + b_u(t) + b_i(t) + p_u^T(t) q_i \tag{5}$$

with

$$p_{u,j}(t) = p_{u,j} + \alpha'_{u,j} \cdot sign(t - t_u)|t - t_u|^{\beta'} + p_{u,j,t} \tag{6}$$

The associated minimization problem must consider regularization terms, and can be solved using the stochastic gradient descent method[2].

[1]19th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, www.sigkdd.org/kdd2009/.

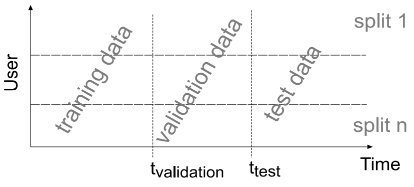

Figure 1: Schematic view of data division.

2.2 Evaluation of the Model

Even though the logical evaluation framework to test a time-aware RS would be a time-centered one, where predictions of the model are to be realized without knowledge of "future" ratings (i.e all training data having a date prior to prediction date), it has not been the case on the aforementioned recommendation method. As user studies are too costly to test every new algorithm, the common way is to use rating data recorded for a timespan, hiding a part of it for testing, and leaving the rest of the data for training. As noted in [6], commonly this evaluation is performed without a temporal perspective, whilst data changes as time goes by. Due to data sparsity and user behavior of rating most items at incorporation time, it is not always feasible to make a strict time-aware train/test data division. For instance, the renowned Netflix Prize competition used as testing data a fixed number of the most recent ratings of each user [1]. Although this scheme seems reasonable, as noted in [7] it is not expected that all users rate the same number of items. Moreover, from a time-aware perspective, it must be noted that some training data correspond to "future" data w.r.t. some test data within this scheme. Given that the discussed model is heavily dependent on such data, we consider that a more strict evaluation is required to test its true benefits.

3. EVALUATION

3.1 Evaluation Protocol

We used the *MovieLens 1M* dataset[3], comprising 1.000.209 ratings (with timestamps) from 6.040 users on 3.706 items. This dataset correspond to users of the *MovieLens* movie RS who joined the system during 2000. Ratings' timestamps span from April 26th, 2000 to February 28th, 2003.

[2]due to space constraints the interested reader is pointed to [4] and references therein for additional details.
[3]http://www.grouplens.org/node/73

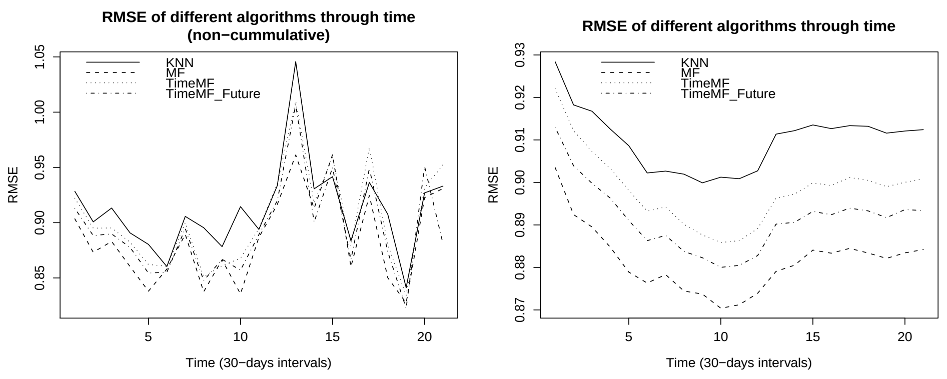

Figure 2: Average RMSE vs. time on 5 time-aware splits of MovieLens 1M dataset

As a basic evaluation protocol, the dataset must be divided into train and test sets. The strictest time-aware approach would be to simulate the arrival of ratings based on timestamps (having different training/test sets for each user/item/rating tuple) [7], with the disadvantage of being very computationally expensive. As our main concern is to restrict the usage of "future" data in training, a natural date-based data division would be to take as training data those ratings done until a predefined date t_{test}, and leave as test data ratings after such a date. We also would like to perform a sort of cross-validation. As discussed in the previous section, a common n-folds based splitting approach with ratings randomly divided into n splits (that is, n-1 splits used as training data and the one left as testing data), is not suitable given the temporal nature of the data (and our aim of detecting temporal trends). Therefore, we decided to make a user-based sub sampling of the dataset, thus allowing the selection of different users in different time-ordered "splits". Thus, ratings prior to t_{test} from users in a split become the training set, and ratings posterior to t_{test} from users in the same split become the test set, i.e. recommendations generated with training set can be contrasted against the same users' future ratings (posterior to t_{test}). This scheme also allows building an additional validation set (required to adjust parameters of the model) whose ratings lie in between training and test ratings. Figure 1 shows a schematic view of the data selection design used.

In order to facilitate the fair comparison among different algorithms, we decided to select only those users with at least 1 rating in each interval (train, validation and test). 504 users of the dataset met the selection criteria (with $t_{validation}$= January 1st, 2001 and t_{test}=June 15th, 2001. We divided them randomly into 5 subsets of 100 users each and form a split with 4 of the 5 subsets. Thus we got 5 different overlapping samples of 400 users. We call these the *Time-Aware* splits. In order to contrast our results, w.r.t. the used *leave last ratings as test* approach, we generated another split using the 504 selected users, but using the 9 last ratings of each user as test, and letting as training set the remaining ratings. We call this the *Last Ratings* split.

3.2 Tested Algorithms and Metrics

We used the basic MF algorithm [5] (without biases consideration) and a Weighted Pearson similarity user based kNN algorithm [3] with $k = 200$ as baselines to contrast performance against the Time-Aware MF model [4]. This model considers many variables whose values depend on specific dates, which are useful in a NetFlix-like evaluation (e.g. if there are training ratings of a user on a day, and there is a test rating on the same day, the model can take advantage of knowing such day's specific user rating bias). Thus, we tested two variants of this algorithm, TimeMF and TimeMF_Future. The former sets to zero every time dependent members of the model (e.g. $b_{i,Bin(t)} = 0$ in (4) and $\alpha'_{u,j} \cdot sign(t - t_u)|t - t_u|^{\beta'} = 0$ in (6)), thus in this case the model should only express the long-lasting user tastes. The latter sets these members with extrapolated values from training data if possible. In particular, we use the value of the last temporal bin of item bias for any test date, and the resulting values of $\alpha_u \cdot sign(t - t_u)|t - t_u|^{\beta}$ and $\alpha'_{u,j} \cdot sign(t - t_u)|t - t_u|^{\beta'}$ setting t as the test rating date; However, $b_{u,t}$ and $p_{u,j,t}$ terms can not be extrapolated, as they depend on the specific prediction day, thus no training data is available to train them. All parameters of MF variants were optimized using the Simplex optimization method, using $f = 10$ factors. We evaluated performance on the rating prediction task using the typical RS metric Root Mean Squared Error (RMSE), meanwhile the top-N recommendation task was evaluated using the well-known IR metrics Prediction and Recall, choosing as relevant items those in the test set of each user with a rating ≥ 3.0. For ranking generation, algorithms were forced to predict ratings for each user on all items in the test set (among all users), then ranked each item according to prediction (leaving out items already rated by the user). Prediction and Recall were computed for each user, and then averaged among all users.

3.3 Results Analysis

Figure 2 shows RMSE results, averaged on the 5 time-aware splits. The right plot was computed using cumulative test data (e.g. on time interval 5 all test data from time interval 1 to 5 are considered), whilst left plot was computed using only data from the corresponding time interval. Interestingly, all algorithms showed a similar behavior, decreasing error as time increases (from t_{test}) during first year. It seems that changes in test ratings distribution affect algorithms almost equally. More important, the lowest RMSE value is achieved by the MF algorithm almost during the whole period considered, which confirms our suspected hypothesis. In the case of Precision, we centered the analysis

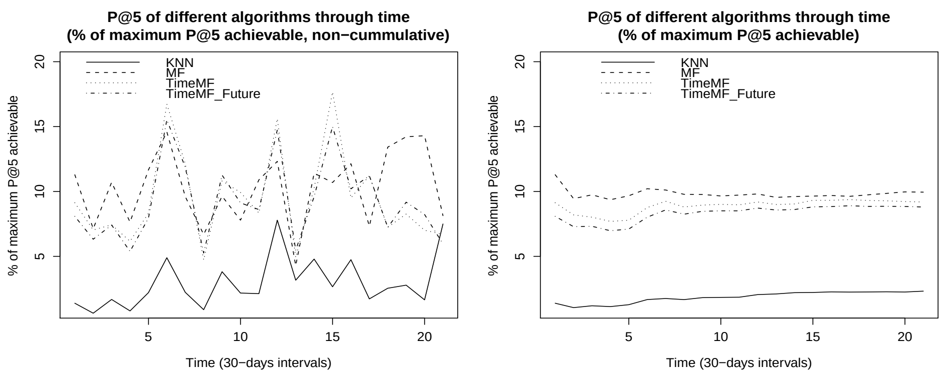

Figure 3: Average P@5 vs. time on 5 time-aware splits of MovieLens 1M dataset

Table 1: Metric values on MovieLens 1M dataset (Last Ratings split)

Algorithm	RMSE	% of max. P@5 achievable
kNN	0,9199	1,56%
MF	0,8949	2,49%
TimeMF	0,9139	2,01%
TimeMF_Future	0,8854	2,06%

on top-5 recommendation. We obtained increasing values of P@5 (Precision on top-5 recommended items) in the range 3%-9% for cumulative data, meanwhile the non-cumulative were more stable around 3%. Due to low rating frequency of users in periods after incorporation, the maximum P@5 achievable is lower than 1 (increasing as more test data is available). To take into account this effect, in Figure 3 we present the percentage of P@5 achieved out of the maximum P@5 achievable. As in the case of rating prediction, MF is the best performing algorithm on top-N recommendation on accumulated test. However, with non-cumulative data, there is no clear best performing algorithm, although all MF variants clearly outperform kNN. We measured Precision at other cut offs, obtaining similar results. In the case of Recall results (not presented due to lack of space) a similar tendency is shown. In contrast, Table 1 shows results on the *Last Ratings* split. In this case, as expected, the Time-Aware MF with future data variant showed the better RMSE results. P@5 achieved (out of the maximum P@5 achievable) obtained is very poor, although the best performance is achieved by MF.

4. CONCLUSIONS AND FUTURE WORK

We have presented results of ongoing work aiming to evaluate the true capability of time-aware RS proposed in the literature to effectively predict future users' tastes and rating behavior using a more strict time-aware evaluation. In this case, we focused on determining if the incorporation of additional, more complex terms in MF models help in these tasks. Results obtained show two major findings: 1) On evaluated tasks, simpler is better, i.e. using a basic MF model yields to better results than more complex models, which we attribute to over-fitting. We must note however that any time-aware MF variant beats another simpler model, KNN. The latter is probably due to the more global nature of MF. 2) Using a more strict time-aware evaluation (i.e. nearer to real world settings) may lead to important differences in results, when compared with results obtained with a less restrictive scheme, particularly on rating prediction task. Although top-N recommendations seems less affected, it is notable that the addition of time information leads to poorer results, independently of the evaluation protocol used, although we must note that the used algorithms are not intended for this task. With respect to the dataset, we reckon that the used one is very small compared to nowadays available data (with hundred of millions of ratings), and outdated. However, it does allow to see the differences in results when the evaluation protocol is changed. Thus, we plan to continue researching on this topic, replicating it on larger datasets, including other time aware models proposals, and also including other kind of datasets, e.g. in other domains, or with implicit feedback information. We think that for a correct selection of RS models, a proper, cost-effective though realistic evaluation setting must be used. With this work, we hope to contribute to this purpose.

5. ACKNOWLEDGMENTS

This work is supported by the Spanish Government (TIN 2008-06566-C04-02) and by the Comunidad de Madrid and Universidad Autónoma de Madrid (CCG10-UAM/TIC-5877).

6. REFERENCES

[1] J. Bennet and S. Lanning. The netflix prize. *KDD Cup and Workshop*, 2007.

[2] Y. Ding and X. Li. Time weight collaborative filtering. In *CIKM 2005*. Bremen, Germany, 2005, 485–492.

[3] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *SIGIR 1999*. Berkeley, CA, 1999, 230–237.

[4] Y. Koren. Collaborative filtering with temporal dynamics. In *KDD 2009*. Paris, France, 2009, 447–456.

[5] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[6] N. Lathia. Evaluating collaborative filtering over time.*PhD Thesis, University College London*, 2010.

[7] G. Shani and A. Gunawardana. *Evaluating Recommendation Systems*, 257–297. Recommender Systems Handbook. Springer US, 2011.