An overview on the exploitation of time in collaborative filtering

João Vinagre[1,3], Alípio Mário Jorge[1,3], and João Gama[2,3]

[1]FCUP - Universidade do Porto, Portugal
[2]FEP - Universidade do Porto, Portugal
[3]LIAAD - INESC TEC, Porto, Portugal
jnsilva@inesctec.pt, amjorge@fc.up.pt, jgama@fep.up.pt

Abstract

Classic Collaborative Filtering (CF) algorithms rely on the assumption that data are static and we usually disregard the temporal effects in natural user-generated data. These temporal effects include user preference drifts and shifts, seasonal effects, new users and items entering the system – and old ones leaving –, user and item activity rate fluctuations and other similar time-related phenomena. These phenomena continuously change the underlying relations between users and items that recommendation algorithms essentially try to capture. In the past few years a new generation of CF algorithms has emerged, using the time dimension as a key factor to improve recommendation models. In this overview we present a comprehensive analysis of these algorithms and identify important challenges to be faced in the near future.

1 Introduction

Over the past two decades, an increasing number of researchers has focused on Recommender Systems. Collaborative Filtering (CF) is one of the most studied recommendation methods and has been successfully used in a large number of applications, such as e-commerce websites [55] and other on-line communities in a series of domains [78, 35, 89]. Competitions like the Netflix prize [7], the KDD-Cup 2011 [23], the Million Song Challenge [62], the successive ACM RecSys Conferences, Workshops and Challenges, as well as the increased demand from both the academic community and the industry, have motivated many notable contributions in the field.

Although a great amount of research has been dedicated to the development and improvement of CF algorithms, many challenges still prevail. Some are related with technical dimensions such as accuracy and scalability – easy to measure offline. Others are more closely related to the users' perspective, such as trust, novelty and serendipity, diversity and general user engagement and perceived quality, which are more difficult to measure. This overview focuses on algorithms that try to exploit *temporal dynamics* to improve accuracy.

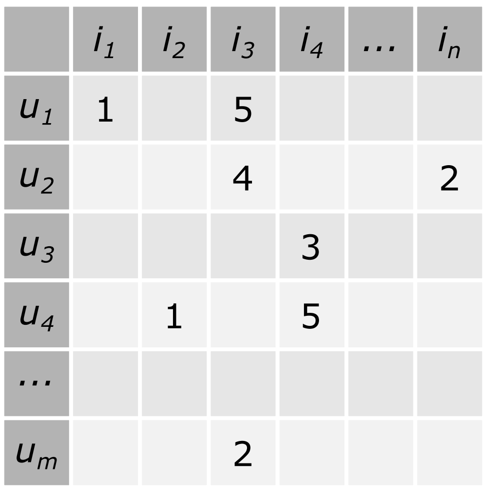
The contributions studied in this overview try to solve the problems that arise from temporal effects. It is reasonable to assume that user preferences change over time. Some of these preference changes may occur in a short time frame, while others may occur more slowly and gradually over a large period of time. Moreover, the systems themselves are dynamic. New users and items enter the systems and old ones leave. These phenomena that occur over time continuously change the underlying user-item relations CF algorithms try to capture. This overview reviews the literature on state-of-the-art algorithms and techniques that try to deal with time related issues in CF recommenders, and identifies future challenges of related research.

In this overview we distinguish between two categories of algorithms according to how they approach the time dimension: time-aware and time-dependent. The first explicitly model time features, such as the day of month, hour or week. The latter approach treats ratings as a chronological sequence, implicitly trying to capture temporal dynamics. The distinction between time-aware and time-dependent algorithms is introduced in a recent survey by Shi et al. [90]. For the sake of clarity, we also separate between neighborhood-based algorithms, matrix factorization algorithms, and other fundamentally different approaches. In another recent survey on temporal CF [12], Campos et al. review recommendation algorithms that deal with time, emphasizing evaluation. In this overview, we add other approaches to temporal CF, as well as the most recent work. Our approach is meant to be a quick reference both to academics and practitioners that intend to gain a general and not very technical introduction to the state-of-the-art in temporal recommendation.

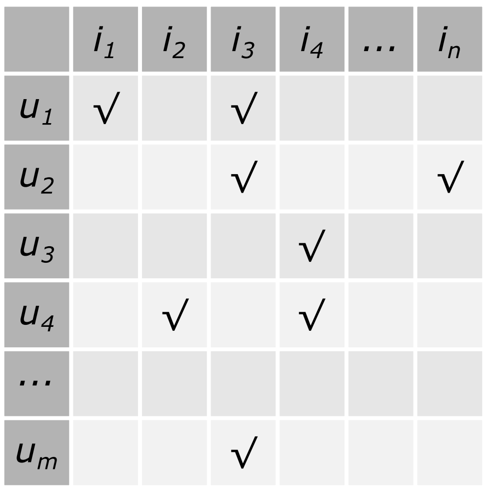
The remainder of this overview is structured as follows. We begin in Section 2 by briefly describing the most relevant CF methods that do not use time information. In Section 3 we review all relevant literature related to time-aware and time-dependent CF. The issues related to evaluation are described in Section 4. We present a discussion in Section 5 and possible future challenges in Section 6. Finally, we conclude in Section 7.

2 Classic collaborative filtering methods

In many on-line virtual communities there is a large number of users that browse through items in the system. Items can be movies, music, books, touristic attractions, restaurants or any other kind of product of interest. In such systems, users are frequently allowed to give their personal opinion about items, by rating that item either explicitly – e.g. using a "like" button or a 5 star rating scale – or implicitly – e.g. number of times a user listens to a music track, or whether a user has bought some item or not. Suppose a system has n users and m items. By collecting feedback from users, it is possible to build a user-item feedback matrix $R_{n\times m}$ containing all ratings given by users to items. Typically R is a very sparse matrix – users usually only rate a very small proportion of the items in the system. The core task of CF algorithms is to predict those missing values and, based on these predictions, offer personalized recommendations to individual users.



	i_1	i_2	i_3	i_4	...	i_n
u_1	1		5			
u_2			4			2
u_3				3		
u_4		1		5		
...						
u_m			2			



	i_1	i_2	i_3	i_4	...	i_n
u_1	√		√			
u_2			√			√
u_3				√		
u_4		√		√		
...						
u_m			√			

Figure 1: Example of feedback matrices. On the left, a typical numerical ratings matrix. On the right a positive-only ratings matrix.

In this section, we introduce the two main variants of the recommendation task and introduce the traditional CF methods that do not use time information.

2.1 Task definition

The ultimate task of any recommender system is simple: given a set of preferences provided by a user, recommend a set of items that are *relevant* for that user. For example, given that a user has shown interest in a certain set of books, the main task of a book recommender system is to provide suggestions to the user of other books which might be interesting to her. Typically, the kind of items that users have shown interest for – e.g. books, movies, music – is the same as the recommended items, although cross-domain recommendation [8, 15, 24] is also possible. The recommendation problem can be cast in several ways, depending on the domain, interface, customer profiles and the business model of the system as a whole. In [34], Herlocker et al. identify and describe the most common recommendation tasks. In this paper we focus on the most conventional task of recommender systems, as defined by Herlocker et al., which is the *Find good items* task. This task consists of presenting the users personalized recommendation of items – typically in the form of a list – which may be of interest – *good* – to them.

2.1.1 Types of feedback

One important distinction, which is determinant on the choice or development of an algorithm, is the one between the two possible types of user feedback data:

- Numeric ratings feedback: typically composed of triples in the form (u, i, r), consisting of the rating value r being given by user u to item i;
- Positive-only or unary feedback: a set of pairs in the form (u, i), representing a positive interaction between user u and item i.

When numeric ratings are available, the main task consists of predicting missing values in the user-item matrix. This is a natural formulation when numeric ratings are available, and the problem is often seen as a regression task. However, some systems employ positive-only ratings. These systems are quite common – e.g. "like"/"favorite" buttons, music streaming, shopping carts, news reading. In these cases, the matrix R is a boolean value matrix (Fig. 1), where *true* values indicate a positive user preference, and $false$ – typically the vast majority – may indicate one of two things: the user either does not like or does not know the item. In systems with positive-only user feedback, the task is to predict *true* values in R, which is more closely related to classification problems. This type of feedback is also known in the literature as "binary ratings" or "implicit feedback". We adopt the term *positive-only*, since the term *binary* may suggest the existence of both positive and negative feedback and the term *implicit* may not be accurate – for instance, clicking a "like" button can hardly be considered an implicit preference. Recommendation using positive-only feedback is also known in the literature as *one-class collaborative filtering* [69].

Considering our focus on the most common recommendation task – to recommend lists of good items to users – the type of feedback being used is highly relevant. For example, when using numeric ratings, a recommendation list can be easily produced by sorting items by descending predicted rating. This, however, is not so trivial when using positive-only data. Generally, we either need to predict some kind of preference level for items – in order to sort them for each user – or to directly approach the task as a ranking prediction problem.

2.2 Classic algorithms

Most state-of-the-art CF algorithms are based on either neighborhood methods or matrix factorization methods. Fundamentally, these differ on the strategy used to process data in the feedback matrix R.

2.2.1 Neighborhood-based algorithms

Neighborhood-based CF algorithms essentially compute user or item neighborhoods using similarity measures such as the cosine or Pearson correlation [85]. If the rows of R represent users and the columns correspond to items, similarity between two users u and v is obtained using the rows corresponding to those users, R_u and R_v. Similarity between two items i and j can be obtained between the columns corresponding to those items R_i^T and R_j^T. Recommendations are computed by searching and aggregating through user or item neighborhoods. The main advantages of neighborhood methods are their simplicity and ease of implementation, as well as the easy explainability of recommendations – user and item similarities are intuitive concepts. The main downside of neighborhood-based methods is the lack of scalability, since that both size and space complexity grow simultaneously with the number of users and the number of items in the system.

2.2.2 Matrix factorization methods

Over the last decade several Matrix Factorization (MF) algorithms for CF have been proposed and greatly popularized. So far, MF methods have proved to be generally superior to neighborhood methods in large scale problems, in terms of both predictive ability and run-time complexity [90].

Matrix Factorization for CF was initially inspired by Latent Semantic Indexing [19], a popular technique to index large collections of text documents, used in the field of information retrieval. LSI performs the Singular Value Decomposition (SVD) of large document-term matrices. In a CF problem, the same technique can be used in the user-item matrix, uncovering a latent feature space that is common to both users and items. One problem with SVD is that classic factorization algorithms, such as Lanczos methods, are not defined for sparse matrices. This issue has been addressed by performing some form of value imputation in the user-item matrix [9, 86]. This, however, is a potential source of systematic error, especially taking into account that the user-item matrix in CF problems is typically very sparse.

As an alternative to classic SVD, optimization methods [6, 26, 72, 93] have been proposed to decompose (very) sparse user-item matrices[1]. Supposing we have a user-item matrix R, the algorithms decompose R in two factor matrices A and B that, similarly to SVD, cover a common latent feature space. Matrix A spans the user space, while B spans the item space. Given this formulation, the predicted rating by user u to item i is given by the dot product $\hat{R}_{ui} = A_u.B_i^T$.

Training is performed by minimizing the L_2-regularized squared error for known values in R and the corresponding predicted ratings:

$$\min_{A.,B.} \sum_{(u,i)\in D} (R_{ui} - A_u.B_i^T)^2 + \lambda(||A_u||^2 + ||B_i||^2) \tag{1}$$

In the above equation, D is the set of user-item pairs for which ratings are known and λ is a parameter that controls the amount of regularization. The regularization term $\lambda(||A_u||^2 + ||B_i||^2)$ is used to avoid overfitting. This term penalizes parameters with high magnitudes, that typically lead to overly complex models with low generalization power. The most successful methods to solve this optimization problem are Alternating Least Squares (ALS) [6] and Stochastic Gradient Descent (SGD) [26]. It has been shown [26, 72] that SGD based optimization generally performs better than ALS when using very sparse datasets, both in terms of model accuracy and run time performance.

Given a training dataset consisting of tuples in the form $< user, item, rating >$, SGD performs several passes through the dataset – iterations – until some stopping criteria is met – typically a convergence bound and/or a maximum number of iterations. At each iteration, SGD sweeps over all known ratings R_{ui} and updates the corresponding rows A_u and B_i^T, correcting them in the opposite direction of the gradient of the error, by a factor of $\eta \leq 1$ – known as step size or learn rate. For each known rating, the corresponding error is calculated as $err_{ui} = R_{ui} - \hat{R}_{ui}$, and the following update operations are

[1]In some of the literature, these methods are often referred to as SVD, despite being formally different methods.

performed:

$$\begin{aligned} A_u &\leftarrow A_u + \eta(err_{ui}B_i - \lambda A_u) \\ B_i &\leftarrow B_i + \eta(err_{ui}A_u - \lambda B_i) \end{aligned} \tag{2}$$

One obvious advantage of SGD is that complexity grows linearly with the number of known ratings in the training set, actually taking advantage of the high sparsity of R.

Other proposed factorization methods include probabilistic Latent Semantic Analysis (pLSA), used in [36] and [92], and CF via Principal Component Analysis (PCA) [31].

2.2.3 Other methods

Although the majority of research on algorithms for recommender systems is focused on either neighborhood or matrix factorization methods, other alternatives have been proposed to solve CF problems. Since early work [10] probabilistic strategies have been presented, such as *clustering* and Bayesian networks. Clustering is motivated by the assumption that it is possible to group users in clusters according to their preferences. In [30, 91, 17], *co-clustering* – or *bi-clustering* – simultaneously clusters users and items, with gains in computational performance and without significant accuracy loss. The reasoning behind co-clustering it that it is frequent that users in a cluster prefer specific subsets of items.

Other probabilistic approaches are based on Graph Theory [4, 40, 39, 25, 33, 94]. For systems with binary ratings, Association Rules [84, 65, 54] and Markov Chains [88, 76] have also been proposed.

In [44], the MF problem is reformulated as an Euclidean embedding problem that joins users and items in a common Euclidean space. The model is learned with SGD in a analogous process to MF and Euclidean distances between users and items are directly used to predict ratings.

3 Collaborative filtering and time

The relevance of time information in recommender systems is based on the assumption that one or more concepts modeled by recommender systems – e.g. user preferences, item popularity – naturally change over time. Traditional CF algorithms do not account for this and need to be frequently retrained to adjust to time related phenomena. For example, travel destination recommendation models would probably need to be retrained to adjust to the current season. This has motivated researchers to investigate new techniques that automatically adjust models to time-evolving phenomena, therefore avoiding complicated maintenance of recommendation models. The main distinction between algorithms that deal with time lies on how the time dimension is approached.

3.1 Approaching the time dimension

One strategy to exploit time is to use it as context information when training recommendation models. Using time as context, CF algorithms use timestamps as an additional source of information, thereby enriching the model. The natural reasoning behind this strategy is that users tend to repeat habits at regular time intervals or moments. By capturing the time at which user preferences are observed in the past, time-aware algorithms make better predictions in similar time patterns occurring in the future. For example, when requesting movie recommendations for the weekend, predictions would be expected to match the general preferences of the user during weekends. These predictions would possibly be different from those that would have been made for weekdays. Following the terminology used in [90], we refer to algorithms using this approach as *time-aware* algorithms. Generally, they are a special case of *context-aware* algorithms [2], in which the context is given by some kind of temporal information, such as the time of day, day of week or other similar time feature. This information is typically exploited to adjust recommendations to the time for which they are requested.

Another way to approach time is to look at user preference data as a chronologically ordered sequence, such as a time series or a data stream. For this approach, timestamps are not strictly required, since time information itself is not necessarily used. Instead, the approach is to train the model in a way that the chronological order is captured and used to improve predictions. Model training is preferably – but not necessarily – performed in incremental steps, and some kind of recency-based modeling scheme can be used to tackle time-varying concepts. As in [90], we refer to algorithms using this approach as *time-dependent* algorithms, although terms such as *sequence-aware* CF or simply *sequential* CF can be used, since these are algorithms that exploit user feedback sequentially. Within this approach we also identify some contributions in which time is used to categorize the users' preferences in terms of their temporal stability. If one considers that users have some preferences more persistent than others, these can be explicitly modeled as long-term and short-term preferences.

3.2 Time-aware algorithms: time as context

Time-aware algorithms specifically use time-related information as context. Typical time features are time of day, day of week, working/non-working hours and seasonal information – e.g. winter/summertime, holidays. In the context-aware framework, these time features can be used in three ways, identified by Adomavicius et al. in [2]:

1. *Pre-filtering*: filter input data considered by the recommender according to the time for which recommendation is requested. One extreme example is to ignore all ratings given by a user in weekdays when she requests recommendations for weekends. In general, pre-filtering is achieved by modifying the input data previously to modeling;
2. *Post-filtering*: filter out irrelevant items from recommendations originally produced by a conventional time-agnostic model. This generally works by filtering out from conventional, time-unaware recommendations the items that do not match the time for which the recommendation is requested;
3. *Modeling*: explicitly model the time features at the learning stage. Here the model should be able to produce adequate recommendations according to the time for which recommendations are generated.

A considerable body of work is available specifically on context-aware recommendation, although most of it is beyond the scope of this overview. We will only refer to those works that deal specifically with time.

3.2.1 Time-aware factorization models

Tensor decomposition models By projecting the user-item matrix in the time dimension, a three-dimensional tensor can be obtained. By factorizing this tensor, it is possible to model ratings according to the time at which past ratings occurred – a time modeling approach. The assumption is that users tend to have cyclic habits, – e.g., watching comedies on sundays, or listening to classical music in the evening. Let $R \in \mathbb{R}^{|U| \times |I| \times |T|}$ be a three-dimensional tensor where dimensions span U, I and T, respectively the set of users, set of items and set of time features – e.g time of day, day of week. There is a considerable number of methods to perform the actual decomposition [46], however the CONDECOMP/PARAFAC (CP) decomposition model has shown to be well suited for sparse tensors [1]. The CP decomposition model is illustrated in Fig. 2. Three factor matrices can be obtained, each spanning one of the tensor's dimensions, by minimizing the prediction error on known ratings. The tensor can then be rewritten as ($\circ$ denotes the outer product):

$$\hat{R} = \sum_{d=1}^{k} U_d \circ I_d \circ T_d \tag{3}$$

In the 2010 Challenge in Context-aware Movie Recommendation – CAMRa2010 [81] –, one of the tasks consisted of recommending movies for specific weeks. Two contributions to this challenge rely on tensor factorization. In [28], Gantner et al. use a tensor factorization model (Pairwise Interaction Tensor Factorization - PITF) originally developed for tag recommendation [77]. The PITF model separately performs pairwise factorizations between every two dimensions. Formally,

$$\hat{R} = \sum_{d=1}^{k} U_d . I_d + \sum_{d=1}^{k} U_d . T_d + \sum_{d=1}^{k} I_d . T_d \tag{4}$$

The authors use weekly time-bins – with some overlapping – according to the week of rating. By adding this new dimension to the user-item matrix and obtaining the corresponding tensor factorization the model is capable of taking advantage of the temporal context. Despite this, the method was unable to outperform a state-of-the-art time-agnostic algorithm [75] that does not use time information.

Also within CAMRa2010, in another contribution [56], Liu et al. propose two time-aware methods for the same task in the competition. The first method is based on CP tensor factorization – by adding time bins as a dimension to the user-item matrix – and the other is a sequential matrix factorization model that consists of several factorizations, one for each time bin – a pre-filtering method. Both methods were successful in outperforming baselines that consisted of time-agnostic versions of the proposed algorithms. In [57], Liu et al. provide a more detailed description of sequential matrix factorization and show that combining temporal and social network context information, enables the method to outperform state-of-the-art time-independent algorithms.

Other time-aware factorization models In [5], Baltrunas and Amatriain use a time-aware factorization algorithm that subdivides user profiles in *micro-profiles.* Each micro-profile is a representation of a user within a specific time frame – e.g. weekend, weekday, morning, winter. One advantage is that pre-filtering can be performed using overlapping criteria. For example, recommendations for a user in a summer weekend morning can be

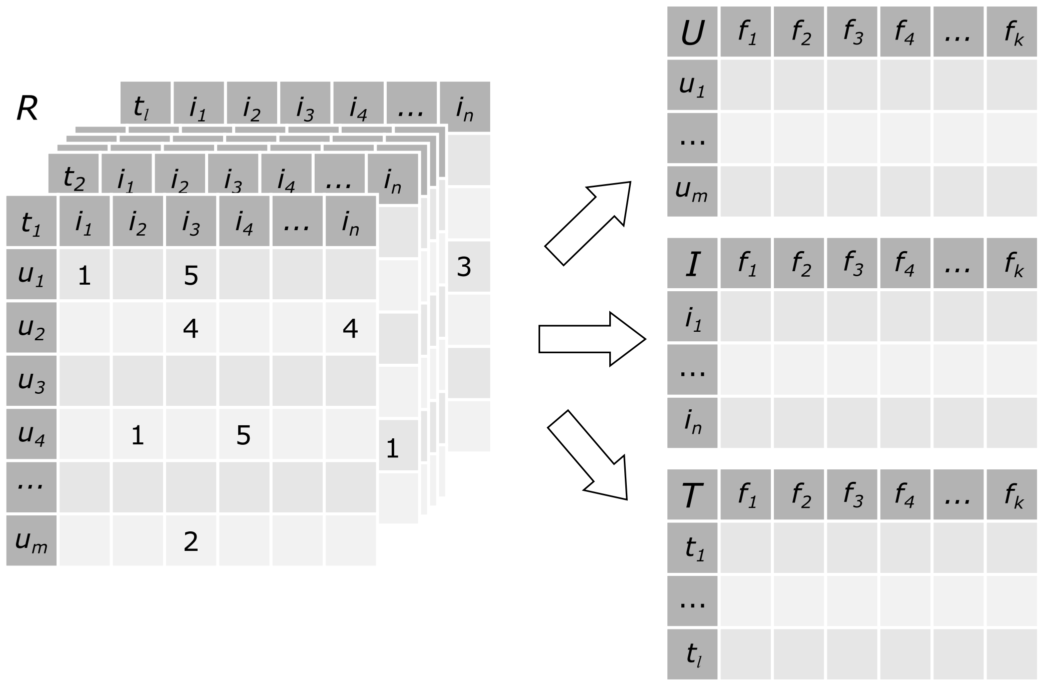


Figure 2: CANDECOMP/PARAFAC tensor factorization model: three matrices are obtained covering the same latent feature space $f_{1...k}$. The time-related dimension $t_1 \ldots t_l$ is a time feature extracted from the ratings timestamp – e.g. day of week, month, hour of day. Every cell in the original tensor can be predicted using the inner product of the three corresponding vectors $\hat{R}_{uit} = \sum_{d=1}^{k} U_{ud} I_{id} T_{td}$.

based on three – *summer, weekend, morning* – micro-profiles. One surprising finding in this work is that the best improvement is achieved when using an apparently meaningless micro-profile – even vs odd hours –, which actually challenges the authors' reasoning.

A different approach is followed by Gao et al. in [29]. The authors use non-negative matrix factorization using the time of day on a Point-Of-Interest (POI) recommender system. The algorithm is able to build a model that is sensitive to check-in times in referenced locations. A total of 24 user factor matrices are obtained by factorizing separately for each of the day's 24 hours, while using the same location factor matrix throughout the learning process. Additionally, consecutiveness is exploited by means of a regularization term that penalizes predictions based on distant times of day. Recommendations are obtained by aggregating recommendations obtained from each of the 24 time-based sub-models. The algorithm is able to considerably outperform the baseline time-agnostic factorization method as well as a classic neighborhood-based algorithm.

3.2.2 Time-aware neighborhood models

Youan et al. propose a time-aware Point-Of-Interest (POI) recommendation system in [102], by incorporating the time-of-day information in the cosine similarity calculation between users in a classic user-based CF algorithm. This explicit modelling approach is based on the assumption that users that check-in in the same locations *at the same time of day* share a higher similarity than users that check-in in the same places but at different times of the day. Additionally, the authors use a probabilistic model to capture popularity patterns over time – e.g. restaurants at meal time, theaters in the evening. Results suggest that modeling the time context for POI recommendation can significantly improve accuracy.

3.2.3 Other time-aware contributions

There are several other examples of algorithms that use time as context available in the literature on context-aware recommendation. Some examples are [3, 22, 71, 32, 52, 70]. We do not elaborate on these since they use broader contexts in which time is not studied separately from the other context features, and therefore the impact of the usage of time features is not evaluated.

3.3 Time-dependent algorithms: time as sequence

Time-dependent algorithms try to capture the phenomena related to continuous temporal dynamics. These phenomena encompass individual user preference changes, drifts in item popularity, fluctuations in activity rates and virtually any temporal effect that may underlie in sequential usage data. Unlike time-aware algorithms, the main objective is not to model cyclic phenomena, but rather to be able to adjust to unprecedented changes. A large variety of contributions have been made to deal with this. Like time-aware algorithms, both neighborhood and factorization time-dependent algorithms have been proposed and/or adapted. Additionally, given the approach to data as a chronologically ordered sequence, there is a natural approximation to the field of data stream mining. As a result, algorithms designed for streaming data have been used or adapted for recommendation tasks as well. Combined short-term and long-term user modeling have also been used to tackle natural variations in users' preferences. We also reference two data pre-processing approaches that drive time-agnostic algorithms to reflect time-dependency, and another approach that embeds users, items and time in a common Euclidean space.

3.3.1 Time-dependent neighborhood models

One simple way to adapt neighborhood-based algorithms to temporal effects is to give more relevance to recent observations, and less to past observations, based on the assumption that recent data is more representative of the current reality. This can be achieved using a series of techniques, most of which are based on either discrete time windows [66, 50, 95] or continuous decay functions [20, 58, 95].

Decay function algorithms The main idea of using a decay function is to reduce the importance of past data *gradually.* An example of a time-decay function is:

$$f(t) = e^{-\alpha t} \qquad 0 < \alpha \leq 1 \tag{5}$$

In (5) t is the time elapsed since a given moment in time. The longer the time, the lower the function value. The parameter α controls the amount of decay.

Early work on time-dependent neighborhood-based CF is presented by Ding and Li in [20], in an item-based algorithm. The authors use (5) in the recommendation step, giving higher weight to recently rated items. The rating prediction for an item is weighted by the most recent ratings given to similar items. Formally, rating prediction R_{ui} is obtained by using $f(t)$ to weight the similarities between the candidate item and its neighbors $sim(i, j)$ (J is the set of top-k neighbors). In practice, recently rated items have a stronger weight than those rated longer ago.

$$\hat{R}_{ui} = \frac{\sum_{j \in J}(sim(i,j)R_{uj}f(t))}{\sum_{j \in J}(sim(i,j)f(t))} \tag{6}$$

In later work, the same authors propose in [21] a recency-based weighting method. The rating prediction is weighted by the most recent ratings given to similar items.

$$\hat{R}_{ui} = \frac{\sum_{j \in J}(sim(i,j)R_{uj}W_j)}{\sum_{j \in J}(sim(i,j)W_j)} \tag{7}$$

where the weight $W_j = 1 - \frac{|R_{uj} - R_{cj}|}{M}$, with r_{cj} being the most recent rating given to item j and M the maximum value in the rating scale.

In [58], Liu et al. introduce decaying time functions in both the similarity computation and the rating prediction steps of an item-based algorithm. The authors use (5) in the similarity computation which, in practice, causes pairs of items to be less and less similar as their ratings are given farther apart in time. At the rating prediction step a similar decay function $g(t) = e^{-\beta t}$ is used to make rating predictions, using the same method as in [20]. The only difference between f and g are the decay parameters α and β – which can optionally be the same.

Given two items i and j:

$$sim(i,j) = \frac{\sum_{u \in U_{ij}} R_{ui} f(t_{ui}) \, . R_{uj} f(t_{uj})}{\sqrt{\sum_{u \in U_i} (f(t_{ui}) R_{ui})^2} \cdot \sqrt{\sum_{u \in U_j} (f(t_{uj}) R_{uj})^2}} \tag{8}$$

In the above equation U_i and U_j are the sets of the users that rated items i and j respectively and $U_{ij} = U_i \cap U_j$. The times t_{ui} and t_{uj} are the times at which user u rated items i and j, respectively. This penalizes similarities between pairs of items when large time intervals occur between the corresponding ratings.

In [95] Vinagre and Jorge apply a decay function in incremental user-based and item-based neighborhood algorithms for binary ratings. This is done by multiplying the frequencies of items by a constant factor $\alpha < 1$ at each incremental step, when new data is processed. In practice the result is the same as applying a decay function at the similarity computation step in non-incremental algorithms, but retaining the scalability benefits of incremental algorithms.

Sliding-window algorithms Sliding-window algorithms work by considering only data in a window (sliding window) that contains either the latest N instances – for example, the latest 1000 ratings – or the instances contained in the latest time interval – for example, all ratings given in the last 24 hours.

A user-based neighborhood algorithm is proposed by Nasraoui et al. in [66]. The algorithm uses a sliding window containing a fixed number of instances. The algorithm computes similarities between the latest user sessions. Each user session consists of a number of ratings given by a user in a short period of time – for instance, during 1 hour.

A different approach is made by Lathia et al. [50] using time intervals. The authors use a set of item-based algorithms differing only in the number of nearest neighbors considered to predict ratings. Algorithms are retrained at fixed 7 day intervals with data in the last interval. Error is continuously monitored for all algorithms, and the algorithm with the lowest error so far is selected to provide recommendations.

Vinagre and Jorge [95] present a user-based and an item-based algorithm that compute similarities using the latest n user sessions. These algorithms use a bounded and approximately fixed amount of data to rebuild the similarity matrix, with obvious scalability improvements, but still far worse than the run-time performance of incremental algorithms.

Other approaches Min and Han [64] use a mixture of item hierarchies, user clustering and time-weighted correlation to improve product recommendations.

3.3.2 Time-dependent factorization models

Matrix Factorization (MF) algorithms have also been adapted to deal with sequential data. In [48], Koren extends his SVD++ algorithm [47] to tackle temporal dynamics. This is done by considering the model's time dependent variables as time functions. In the original SVD++ algorithm, Koren separates the model in a baseline model that relies solely in user and item biases b_u and b_i – individual deviations from the global average rating μ – and the factor model, that captures the actual user-item preferences, based on (1), with the addition of implicit data extracted from the same dataset. In the time-dependent algorithm both the user/item biases and the factor model are approached as time functions. To model temporal item biases, time is split in discrete time windows (bins) and for user biases, a decay function is used. Another decay function is used to weigh past ratings at the prediction step. According to the author, time-varying item biases capture item popularity changes, time-varying user biases capture the variations in how individual users use the rating scale and time-varying factors essentially capture the actual preference changes. Empirically, this time-dependent algorithm significantly improves accuracy with the well-known Netflix dataset.

Tensor factorization revisited Tensor factorization has been also proposed for time-dependent recommendations. Here, the tensor's time dimension contains actual time intervals, so the tensor can be viewed as a periodic collection of ratings over time. The time dimension essentially captures the latent features' trends over time.

In [100], Xiong et al. use a CP tensor factorization model by adding a time dimension to the user-item matrix, splitting time in equal length intervals. The authors use Probabilistic Matrix Factorization algorithms studied in [82] and [83] to estimate optimal hyper-parameters. Another contribution using CP tensor factorization is made by Rafailidis and Nanopoulos in [74]. The authors use a smoothing factor based on the observed levels of preference change to weigh down the corresponding user-item interactions in previous time intervals. In practice, this scheme produces a tensor where past preferences are given less importance in the same measure as the user changes her habits.

Other temporal factorization models Another approach was presented by Das et al. [16]. The proposed algorithm combines a neighborhood model with pLSA and MinHash clustering [42] using the parallel computation with MapReduce [18] in a news recommender. In their work, the authors introduce a time-decaying function in click-rates made by each cluster in news items and a time-based window on co-visitation of news.

In [60] Matuszyk and Spiliopoulou propose and evaluate several selective forgetting strategies for incremental matrix factorization algorithms with ratings data. In [61], Matuszyk et al. extend the study with several other forgetting methods and test them with positive-only data in addition to ratings data. In both publications, the authors show that selectively forgetting *some* of the past users' feedback is beneficial to the system. Also using incremental matrix factorization, Vinagre et al. [97] use a recency-based scheme to artificially introduce negative feedback data, tackling a known problem [69, 38] that arises from the absence of negative feedback.

Pálovics et al. [68] use the social influence between users in a social network to improve recommendations. Social influence is modeled using common preferences shown close together in time by two users connected in the social graph.

3.3.3 Short-/Long-term preference modeling

Another way to approach sequentially ordered data is to model the users' short-term and long-term preferences separately. Conceptually, this means that each user has potentially two models, one for long-term preferences and another for short-term preferences. In [79], Ricci et al. identify the importance to deal with the duality of short-term preferences – goal oriented and highly dependent on context – and long-term preferences – durable and stable. A hybrid – both content-based and CF-based – recommender is developed and evaluated online with real-users, showing a considerable reduction on the users' effort in finding travel-related products.

Xiang et al. [99] propose a graph-based model to capture the users' short-term and long-term preferences. User nodes in the graph – connected to all the user's preferred items – encode long-term preferences, and *session* nodes, which are time-restricted, encode the user's short-term preferences. Items that match both the long-term and the short-term preferences of a user are recommended. In [37], Hong et al. explicitly model short-term user profiles by using consecutive stages, which correspond to periods of time where the user's shopping habits are dominated by items belonging to set of categories within an existing taxonomy. Long-term user preferences are given by multiple stages. The authors use clustering-based and graph-based recommendation algorithms to exploit stage information, outperforming other time-dependent algorithms.

Jannach et al. [43] also emphasize the importance of short-term preferences by re-ordering recommendation lists using short-term preference data, with considerable accuracy gains in an e-commerce dataset.

By maintaining a model consisting of an offline component and an online component, Liu and Aberer [59] are able to capture long-term influences – offline component – and short-term preferences – online component. The online component is updated frequently with fresh incoming data, and is therefore more sensitive to context and short-term infulences, while the offline component, containing more stable preferences, is updated much less frequently using the data meanwhile stored in the online compontent. Using this approach, combined with contextual text reviews, the proposed method is able to outperform other state-of-the-art time-dependent algorithms on a dataset extracted from a large ratings website.

3.3.4 Data-stream algorithms

In [53], Li et al. propose an approach to drifting preferences of individual users using the CVFDT algorithm [41]. This is a popular classification algorithm for high speed data streams that automatically adapts to concept drifts. The CVFDT algorithm is used to build a decision tree for each item in the dataset, given the ratings of other highly correlated items. The ratings given by users to these correlated items are used to predict the ratings for the target item. The algorithm can be extended to use item hierarchies – if they exist – with considerable improvements. The mechanics of CVFDT provides automatic adjustment to drifts in user interests, avoiding accuracy degradation.

In the previously mentioned work by Nasraoui et al. [66] a second algorithm uses the TECHNO-STREAMS stream clustering algorithm [67], using a sliding window through user sessions.

3.3.5 Data pre-processing

Two time-dependent methods in the literature consist of data pre-processing techniques – rather than algorithms. The objective is to encode temporal or sequence information in the data itself, which can therefore be used with any time-agnostic algorithm. In [103], Zimdars et al. approach the recommendation as a univariate time series problem, an apply two data transformations that encode sequence in the data. Using a decision tree model, the authors are able to improve accuracy over the baseline that ignores data order. Cao et al. use a data pre-processing approach in [13], which the authors claim to be possible to use with any algorithm. The process consists of identifying four common user behavior patterns and manipulating user data according to the detected pattern. The authors identify one of the patterns as being noisy behavior and remove data generated according to this pattern. Additionally, some pruning is performed on data generated by users identified as having drifting preferences, by retaining the latest interest. The technique is evaluated using neighborhood-based algorithms.

3.3.6 Euclidean embedding

A different proposal is made by Yin et al. in [101], as an extension to the Euclidean embedding framework proposed in [44], where users and items are embedded in the same Euclidean space. By adding time factors to that user-item Euclidean space, the authors claim superior accuracy over the baseline algorithm.

3.4 Algorithms both time-aware and time-dependent

Theoretically, time-aware and time-dependent are not mutually exclusive approaches. However we only found a single contribution that encompasses both techniques. This is done by Campos et al. [11] in the context of the aforementioned CAMRa2010 [81] competition, where the authors use a user neighborhood algorithm that computes recommendations considering recently rated items in the neighborhoods – a time-dependent technique – and ratings given on the same months and days in previous years – in a pre-filtering time-aware technique.

4 Evaluation of temporal algorithms

Reliable and thorough evaluation methodologies for recommender systems are still a very active research topic in the field. However, the variety of evaluation metrics in Tables 1 and 2 indicates that a solid, consensual and easily applicable evaluation framework for recommender systems is still not available. As a result, interpretability and reproducibility problems may occur.

This problem has been recently noted by Said and Bellogín in [80], where three widely used recommendation libraries yield sometimes radically different results for the same algorithm/dataset/parameters/evaluation method combination. Online evaluation – e.g. with A/B testing [45] or user surveys –, while being more informative about the real performance of a recommender system interacting with real users, is not possible to perform in many cases, because access to large scale production systems is not easily available.

In this section we describe important issues with the offline evaluation of time-aware and time-dependent CF. We will not focus on online evaluation, since we do not find significant implications of time-awareness and time-dependency in online evaluation methodologies.

4.1 Offline evaluation

Offline evaluation protocols allow researchers to evaluate and compare algorithms by simulating user behavior. This typically begins by splitting the ratings dataset in two subsets – training set and testing set – randomly choosing data elements from the initial dataset. The training set is initially fed to the recommender algorithm to build a predictive model. To evaluate the accuracy of the model, different protocols can be used. Generally, these protocols group the test set by user – or user session – and "hide" user-item interactions randomly chosen from each group. These hidden interactions form the hidden set. Rating prediction algorithms are usually evaluated by comparing predicted ratings with the hidden ratings. Item recommendation algorithms are evaluated performing user-by-user – or session-by-session – comparison of the recommended items with the hidden set.

The most common protocols using this strategy are *All-but-N* and *Given-N*. The All-but-N protocol hides exactly N items from each user in the test set. One often used sub-protocol is the *All-but-One* protocol, which hides exactly one item from each user in the test set. The *Given-N* protocol keeps exactly N items in the test set and hides all others. In the classic versions of both protocols, hidden ratings are randomly chosen from each user.

Offline protocols try to simulate user activity. However there are a few limitations to consider:

- Dataset ordering: randomly selecting data for training and test sets first, and random hidden set selection second, shuffles the natural sequence followed by users in rating items. However some algorithms are designed to deal with chronologically ordered data;
- Time awareness: many recent recommenders use timestamped ratings to produce time-aware models. By shuffling data, protocols break the

logic of such systems. For example, by potentially using future ratings to predict past ratings;

- Previous user activity requirement: in order to evaluate recommendations using *All-but-N* or *Given-N*, only users with at least $N + 1$ can be considered for evaluation;

- Online updates: some recommender systems have the ability to perform online updates of their models as new ratings are available. This means that neither models or training and test data are static. Models are continuously being retrained with new data;

- Recommendation bias: in an online system, user behavior is – expectedly – influenced by recommendations themselves. For example, it is reasonable to assume that recommended items will have a substantially higher probability to be rated or consumed than if they were not recommended. Simulating this off-line usually requires complicated user behavior modeling which can be expensive and prone to systematic error.

These limitations weaken the assumption that user behavior can be accurately modeled or reproduced in offline experiments. Nevertheless, they still provide a useful tool. In fact, some solutions hav been proposed to solve most of the problems.

In [87] some clues are provided on how to solve some of these problems. One straightforward solution to the first two problems is simply not to shuffle data – or if timestamps are available, pre-order ratings accordingly. That is, to pick a moment in time or a number of ratings in the dataset as the split point for the dataset. All ratings given before the split point are used to train the model and all subsequent ratings are used as testing data. One problem with this approach is how to select the hidden set. In [87] and [49] the authors suggest that all ratings in the test set should be hidden, assuming that users already have had activity before the split point – those who had not are simply ignored. Another possibility is to adapt the *Given-N* and *All-but-N* protocols to preserve order. The hidden set can contain the last N items for each user – *All-but-N* – or hiding all but the first N items – *Given-N*. One downside with order-preserving splits, is that cross-validation is not applicable because it necessarily breaks the order of the data. As an alternative, a *prequential* validation scheme [27] can be used instead.

In our overview, we found that the majority of the evaluation methods is not substantially altered by time related constraints. In most cases, the most important difference from time-agnostic algorithm evaluation is the preservation of the datasets' natural order. This is perhaps because the main objective of most evaluations performed is to prove that by adding temporal information, time-enabled algorithms improve with respect to their baselines and/or other time agnostic algorithms. This requires evaluation methodologies that are compatible with classic algorithms. The obvious exceptions are algorithms that learn incremental models. To solve this particular issue, Vinagre et al. [96] propose the usage of a prequential methodology [27], frequently used in the field of data stream mining. Evaluation is made approaching incoming ratings data as a data stream and evaluation is continuously performed in a test-then-learn scheme: whenever a new rating arrives, the corresponding prediction is scored according to the actual rating. This new rating is then used to update the model.

One important contribution on temporal evaluation is made in [51]. By conducting a user survey, Lathia et al. identify *temporal diversity* as an important factor for recommendation, from the users' perspective. Then the authors propose new metric that measures diversity over time, and develop a method that maximizes this metric.

We also include the use of accuracy metrics in the overviewed literature. When ratings are available, Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) are frequently used – in our overview, ten contributions are evaluated using these metrics. These metrics – and some variants – essentially measure the difference between predicted ratings and the actual ratings given by real users. RMSE is especially practical in matrix factorization using gradient descent, because this is typically the error measure that the algorithm minimizes.

On positive-only data, measuring the error in the same way would be less informative, since we are typically not dealing with predictions within a scale. Instead, metrics are adapted from information retrieval, and measure the accuracy directly on recommendation lists. For example, given a recommendation list consisting of 10 items, accuracy can be measured by finding how many relevant items are within that list. Hit ratio – the proportion of correct recommendations in a recommendation list – is perhaps the simplest metric of this kind and is used in two contributions. However the most common metrics, observed in ten overviewed contributions, are Precision / Recall / F1. We group them together, since in the overviewed papers they either appear together as complementary measures or are easily obtained as a function of the other – e.g. F1 is a function of both precision and recall, and precision and recall are easily obtained from one another if the number of relevant items – all possible good recommendations – and the recommendation list size are known. Precision, recall and derived metrics do not account for the order of recommendations. However, when dealing with recommendation lists, the items' position in the list may be important. Mean Average Precision (MAP) and Discounted Cumulative Gain (DCG) are metrics that score both correctness of the recommended items and the order by which they are recommended. In our overview, two contributions use at least one of these metrics. The Area Under Curve (AUC) – used in two time-aware contributions – is intended to measure the discrimination power of the algorithms.

4.2 Online evaluation

Offline evaluation is important to assess the predictive ability and runtime performance of algorithms. However it is arguable if it is enough [63]. There is no guarantee that an algorithm with good offline results will have good online performance, from the users' perspective. The only way to perform user-centric evaluation is to interact with real users [73]. Obviously, this also applies to algorithms that use time. While time may have relevant implications in offline evaluation methodologies, it has little or no implications on online evaluation methods. This is because online evaluation is not dependent on the algorithms' mechanics. In fact the same online evaluation setting can be used to evaluate any kind of recommender. At most, metrics may be added or adjusted to measure the impact of time exploitation in the algorithms' performance. The major downside of online evaluation is the difficulty to obtain access to an online production system. Moreover, as a consequence, reproducibility is nearly impossible. This is maybe the explanation to why, in our overview, only Ricci et al. [79] evaluate a recommender system online, with real users.

One relevant note about online evaluation methods is that prequential evaluation [27] is designed to run online. One of the benefits is that it can be used to perform online measurements on the user side for posterior analysis. Additionally these measurements are available in real time, which can be useful, for instance, to perform automatic online adjustments.

5 Discussion

All time-aware and time-dependent contributions studied in this paper significantly improve the predictive ability of algorithms. This is clear in the various comparisons between algorithms capable of capturing temporal dynamics and the equivalent algorithms without this capability. This means that adding time-awareness or time-dependency to algorithms that do not have it is clearly beneficial. However, it is also shown by some authors that some state-of-art algorithms without temporal capabilities are quite hard to outperform. Until these algorithms are given the ability to deal with the dynamics of time – and compared with the respective baselines –, it will be hard to adopt them as state-of-the-art algorithms.

5.1 Comparison of time-aware algorithms

Table 1 summarizes the main aspects of time-aware algorithms described in Sections 3.2.1 and 3.2.2. Besides the distinction between matrix/tensor factorization and neighborhood models, we provide the type of *filtering* – as defined for context-aware recommendation in [2] –, the dataset(s) used, evaluation metric and what we identify as the key distinctive strategy – or strategies – of the corresponding contribution.

From our literature review it became clear that time-aware strategies are advantageous in most cases, when compared with the corresponding baseline time-agnostic algorithms. However, when compared with other state-of-the-art algorithms these are not clearly outperformed – e.g. [28]. From the five time-aware algorithms, four rely on matrix factorization – of these, two use tensor factorization – and one is a neighborhood-based algorithm. One explanation for this is that context information – including time – can be easily integrated in an existing factorization model, either by using tensor factorization or by using multiple matrix factorizations. One other possible explanation is related to research timing. The earliest contribution for time-aware recommendation is quite recent. Given that many of the state-of-the-art recommendation algorithms are based on factorization, it makes sense to use these as a basis for research.

Regarding the context-driven – or time-driven – filtering approach, as specified in Section 3.2, the studied algorithms either use a pre-filtering approach or a direct modeling of time features. Post-filtering was not used. All of the datasets used for evaluation are originally positive-only, although Baltrunas and Amatriain [5] map the implicit information to ordinal ratings. This is consistent with the accuracy metrics used to evaluate the algorithms, which are MAE for the contribution that uses ratings, and Precision / Recall or AUC for the others. One important note is that from the five datasets used in all works, only one is publicly and openly available.

Approach	**Algorithm**	**Filtering**	**Datasets**	**Metric**	**Key strategies**
Factorization	Baltrunas and Amatriain 2009 [5]	Pre-filtering	Last.fm[†]	MAE	Time-based segmentation of user profiles (micro-profiling).
	Gantner et al. 2010 [28]	Modeling	CAMRa 2010[†]	AUC / Precision	PITF tensor factorization.
	Liu et al. 2010 / 2013 [56, 57]	Modeling / Pre-filtering	CAMRa 2010[†]	AUC / Precision	CP Tensor factorization / Multiple sequential matrix factorizations.
	Gao et al. 2013 [29]	Modeling / Pre-filtering	Crawled from Foursquare[§]	Precision / Recall	Multiple matrix factorizations based on time of day.
Neighborhood	Yuan et al. 2013 [102]	Both Pre-filtering and Modeling	Foursquare and Gowalla check-in data[†]	Precision / Recall	Time-based user similarity.

[†] Proprietary / Subject to request
[§] Available from http://www.public.asu.edu/~hgao16/dataset.html

Table 1: Comparison of time-aware CF algorithms.

5.2 Comparison of time-dependent methods

In Table 2, we summarize the main characteristics of the overviewed time-dependent algorithms. We identify some key characteristics of the contributions, namely the type of approach – neighborhood-based, factorization-based or other –, the mechanics of model learning – batch or incremental –, the data type, as defined in Section 2.1.1, for which the algorithms are developed, the datasets and metrics used for evaluation, as well as what we identify as the key distinctive strategy – or strategies – of the contributions.

Table 2: Time-dependent CF algorithms and main strategies.

Approach	**Algorithm**	**Training**	**Data type**	**Datasets**	**Evaluation metric**	**Key strategies**
Neighborhood	Ding and Li 2005 [20]	Batch	Ratings	EachMovie‡, Movielens-1M$^{\|}$	MAE	Decay function in rating prediction.
	Ding et al. 2006 [21]	Batch	Ratings	EachMovie‡, Movielens-1M$^{\|}$	MAE	Recency-based weighting in rating prediction.
	Liu et al. 2010 [58]	Incremental	Ratings	Netflix†	RMSE	Decay function in both similarity computation and rating prediction.
	Nasraoui et al. 2007 [66]	Incremental	Pos-only	Web log§	Precision / Recall / F1	Data stream approach with sliding window.
	Lathia et al. 2009 [50]	Batch	Ratings	Netflix†	RMSE	Adaptive neighborhood size controlled by error rate.
	Vinagre et al. 2012 [95]	Batch and incremental	Pos-only	Music streaming log§ / web log§	Precision / Recall	Forgetting with sliding windows (batch) and decay functions (incremental).
Factorization	Koren 2010 [48]	Batch	Ratings	Netflix†	RMSE	SVD model with time varying biases and factors.
	Xiong et al. 2010 [100]	Batch	Ratings	Sales history§ / Movielens-1M$^{\|}$ / Netflix†	MAE / RMSE	Probabilistic tensor factorization.
	Rafailidis and Nanopoulos 2014 [74]	Batch	Pos-only	Lastfm$^{\#}$	Precision / Recall	Coupled CP tensor factorization.

Continued on next page

Table 2 – continued from previous page

Approach	**Algorithm**	**Training**	**Data type**	**Datasets**	**Evaluation metric**	**Key strategies**
	Das et al. 2007 [16]	Batch w/ incremental adjustments	Pos-only	Movielens-100k[‖] / News reading[§]	Precision / Recall	pLSA with MinHash clustering.
	Pálovics et al. 2014 [68]	Incremental	Pos-only	Lastfm[#]	Discounted Cumulative Gain	Temporal social influence in social graph.
	Vinagre et al. 2015 [97]	Incremental	Pos-only	Movielens-1M[‖] / Lastfm[#] Music streaming[§]/ Playlisting[§]	Precision / Recall	Online updates of factor matrices with recency-based negative feedback.
	Matuszyk and Spiliopoulou 2014 [60]	Incremental	Ratings	Netflix[†] / Epinions[*]	RMSE	Selective forgetting.
	Matuszyk et al. 2015 [61]	Incremental	Ratings and Pos-only	Netflix[†] / Epinions[*] / Movielens-1M[‖] / Lastfm[#] Music streaming[§]/ Playlisting[§]	Incremental RMSE and Precision/Recall	Selective forgetting.
Data Streams	Li et al. 2007 [53]	Incremental	Ratings	Eachmovie[†]	MAE / Squared-MAE	CVFDT stream mining algorithm for drifting user preferences.
	Nasraoui et al. 2007 [66]	Incremental	Pos-only	Web log[§]	Precision / Recall / F1	TECHNO-STREAMS stream clustering algorithm.
	Ricci et al. 2003 [79]	Batch	Ratings	Travel website	Online evaluation	Hybrid recommender.

Long-/Short-term

Continued on next page

Table 2 – continued from previous page

Approach	Algorithm	Training	Data type	Datasets	Evaluation metric	Key strategies
	Xiang et al. 2010 [99]	Batch	Pos-only	CiteULike° / Delicious+	Hit ratio	Graph-based with discriminatory nodes (long- vs short-term).
	Hong et al. 2012 [37]	Batch	Pos-only	Shopping history§	Precision / Recall / F1	User profile separation in temporal stages.
	Jannach et al. 2013 [43]	Batch	Pos-only	E-commerce website data§	Precision / Recall	Re-ordering of recommendations according to latest context.
	Liu and Aberer 2014 [59]	Batch	Ratings	Review website data§	NDCG / MAP	Combination of long-term and short-term models.
Data processing	Zimdars et al. 2001 [103]	Batch	Pos-only	Two web access logs§	Non-standard accuracy measure	Data transformations to encode temporal dynamics.
	Cao et al. 2009 [13]	Batch	Pos-only	Movielens-1M‖	Hit Ratio	Data filtering according to user profile type.
Euclidean embedding	Yin et al. 2012 [101]	Batch	Ratings	Movielens-1M‖ / Netflix†	RMSE and Precision / Recall	Extension of user-item embedding with time.

† No longer available
§ Proprietary or subject to request
‖ http://grouplens.org/datasets/movielens (accessed Jan 2015)
* http://www.trustlet.org/wiki/Extended_Epinions_dataset (accessed Jan 2015)
# http://ocelma.net/MusicRecommendationDataset/lastfm-1K.html (accessed Jan 2015)
° http://www.citeulike.org (accessed Apr 2015)
+ http://www.dai-labor.de (accessed Apr 2015)

A total of 24 time-dependent algorithms are overviewed. This is a much larger amount of contributions than time-aware algorithms. Besides six neighborhood-based contributions and eight factorization-based contributions – of which two use tensors –, we also study two data stream mining algorithms adapted for recommendation, five contributions that model the duality of short- and long-term user preferences, two data pre-processing approaches and one algorithm based on Euclidean embedding.

Although the ability to build models incrementally is not exclusive to time-dependent algorithms, its application is more natural in this approach. In the field of data stream mining, for instance, algorithms are typically designed to work incrementally. This facilitates the adaptation to temporal effects such as concept drifts [98]. Recommender systems deal with data that is generated in a similar fashion to a data stream – potentially unbounded, unpredictable rates and ordering – and that are subject to changes over time. The main purpose of time-dependent algorithms is to deal with these changes. This appears to be consistent with the fact that ten of the time-dependent contributions use incremental learning.

Regarding the type of data, thirteen contributions on time-dependent use positive-only data, while twelve use ratings data[2], in a total of 21 different datasets[3]. Of these, six are available to the public. The distribution of the metrics naturally follows the type of data: precision, recall, F1, DCG and MAP for positive-only data, and RMSE and MAE for ratings. Two exceptions to this are [101] and [61] that use Precision / Recall to evaluate recommmendations obtained with ratings data. One contribution uses an ad-hoc, non-standard accuracy metric.

5.3 Findings and general discussion

Analyzing all overviewed contributions, time information is definitely useful to improve recommender systems. This can happen in two dimensions. First all time-aware and time-dependent studies report accuracy improvements over time-agnostic baselines, and many times over state-of-the-art alternatives that do not use time. Second, many contributions, even if not improving accuracy dramatically, present other advantageous features, such as the ability to adjust very fast and without external intervention – e.g. human triggered updates – to different contexts, trends or drifting user preferences. One downside of most of the reported results is the absence of statistical significance tests in comparative assessments. This is especially noticed when

[2][61] uses both ratings and positive-only data

[3]We consider Movielens-1M and Movielens-100k to be distinct datasets

relative differences are low.

One aspect of most of the aforementioned work – and perhaps a relevant research issue – is that run time performance and scalability are somewhat overlooked in the majority of the presented work. While the accuracy of CF algorithms is undoubtedly important, scalability and run time complexity are also major issues in this field of research, and can be decisive factors in choice of a recommender system from the practitioner's point of view. For instance, tensors are highly expressive and conceptually are quite simple to handle, however tensor factorization is also known to be particularly demanding in terms of computational resources.

Regarding reproducibility, from the 26 datasets used in the evaluation of both time-aware and time-dependent methods, only six are publicly available, although some others may possibly be obtained by request. This, associated to slight differences in the evaluation methodology and metrics eventually leads to reproducibility problems. Additionally, direct comparison between results reported by different authors is meaningless [80].

One more general remark about accuracy, and one that has been debated in the community is the scope of accuracy results obtained in offline experiments, and how it impacts on the overall quality from the users' perspective [14].

6 Future challenges

Although temporal CF is still a recent topic, many contributions have appeared in the past years. One big advantage of temporal algorithms is that it has the potential to simultaneously improve recommendations and reduce maintenance. The accuracy improvement is obtained by better reflecting the current reality, given recent and/or periodic usage patterns. However, many open issues persist. In real-world systems, usage data keeps adding up as new users and items enter the system and new ratings are provided. With time-agnostic algorithms, frequent retraining of models is required to keep up with the evolving data. It is therefore a fundamental requirement that CF algorithms are scalable enough to cope with increasing amounts of data. Regarding this, the research in recommender systems algorithms seems to be pointing at the ability of the algorithms to update their models incrementally and online. The ideal CF algorithm should be able to update the recommendation model at a rate faster than the arrival of new data. Thus, a data stream approach to recommender systems seems a viable way to pursue future work in the field of recommender systems, since it is capable of simultaneously capturing time dynamics and require less computational resources.

As computational resources become cheaper, distributed models become more mature, flexible and widespread. Algorithms for recommendation have been successfully used with parallel processing. Distributed models enable the use of algorithms and techniques that otherwise would not be applicable. Research on distributed algorithms for recommendation may allow, for instance, the use of very large data structures – such as tensors – that may have more expressive and accurate temporal models.

One important aspect of the research in recommender systems is evaluation. The majority of the literature focuses on off-line accuracy and scalability evaluation using well studied evaluation protocols and metrics. However, production systems are usually sensitive to a large number of environmental variables that cannot be reproduced in the laboratory. Users of online systems are humans with naturally biased perspectives on the quality and the utility of a recommender [63]. Moreover, the main task of a recommender system may vary considerable with the application [34]. For instance, users might be willing to sacrifice accuracy to obtain serendipitous, less obvious recommendations. In other applications, such as news recommendation, the recency of recommended items is a key factor and may be preferred to high accuracy. On the other hand, scalability issues typically have considerable investment implications. This type of factors cause algorithms with good off-line performance to not translate directly into good online performance. For this reason, online evaluation and user feedback may be determinant to the choice of algorithms and their parameters. One practical way of evaluating online recommenders is by conducting controlled experiments [45] involving real users in a live environment.

7 Conclusions

In this overview, we review the literature on recommender systems able to capture temporal dynamics, and separate these algorithms in two categories: time-aware algorithms and time-dependent algorithms. We refer to five time-aware algorithms, able to capture temporal patterns such as seasonal effects, daily and hourly effects and similar periodic activity patterns that tend to repeat. We also study twenty-four contributions regarding time-dependent algorithms, that capture temporal dynamics by using the natural sequence of the datasets. Additionally, a single contribution using both time-aware and time-dependent techniques is presented.

Our main conclusion is that capturing temporal dynamics, both in time-aware and time-dependent algorithms, is clearly beneficial. This conclusion is drawn from evidence provided in all contributions that time-enabled algorithms systematically outperform their time-disabled baselines. This however, does not guarantee that time-aware algorithms are able to outperform *other* state-of-the-art alternatives in production systems, since evaluation is predominantly performed offline. This suggests that some important future work in the research of temporal recommender systems needs to be undertaken. We identify three lines of work for the future: a) the ability to learn models online at fast rates, b) taking full advantage of distributed computing model and c) narrowing the gap between offline and online evaluation.

8 Acknowledgements

This work is partially supported by the European Commission through project MAESTRA (Grant no. ICT-2013-612944) and is partially funded by FCT/MEC through PIDDAC and ERDF/ON2 within project NORTE-07-0124-FEDER-000059. The first author is also supported by FCT - Fundação para a Ciência e Tecnologia (Portuguese Foundation for Science and Technology) with grant SFRH / BD / 77573 / 2011.

References

[1] Evrim Acar, Daniel M. Dunlavy, Tamara G. Kolda, and Morten Mørup. Scalable tensor factorizations with missing data. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2010, April 29 - May 1, 2010, Columbus, Ohio, USA*, pages 701–712. SIAM, 2010.

[2] Gediminas Adomavicius, Bamshad Mobasher, Francesco Ricci, and Alexander Tuzhilin. Context-aware recommender systems. *AI Magazine*, 32(3):67–80, 2011.

[3] Gediminas Adomavicius, Ramesh Sankaranarayanan, Shahana Sen, and Alexander Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst.*, 23(1):103–145, 2005.

[4] Charu C. Aggarwal, Joel L. Wolf, Kun-Lung Wu, and Philip S. Yu. Horting hatches an egg: A new graph-theoretic approach to collaborative filtering. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 15-18, 1999*, pages 201–212. ACM, 1999.

[5] Linas Baltrunas and Xavier Amatriain. Towards Time-Dependant Recommendation based on Implicit Feedback. In *Workshop on Context-aware Recommender Systems (CARS 2009) at RecSys, New York*, 2009.

[6] Robert M. Bell and Yehuda Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007), October 28-31, 2007, Omaha, Nebraska, USA*, pages 43–52. IEEE Computer Society, 2007.

[7] James Bennett, Stan Lanning, and Netflix Netflix. The netflix prize. In *In KDD Cup and Workshop in conjunction with KDD*, 2007.

[8] Shlomo Berkovsky, Tsvi Kuflik, and Francesco Ricci. Cross-domain mediation in collaborative filtering. In *User Modeling 2007, 11th International Conference, UM 2007, Corfu, Greece, June 25-29, 2007, Proceedings*, volume 4511 of *Lecture Notes in Computer Science*, pages 355–359. Springer, 2007.

[9] Daniel Billsus and Michael J. Pazzani. Learning collaborative information filters. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML 1998), Madison, Wisconson, USA, July 24-27, 1998*, pages 46–54. Morgan Kaufmann, 1998.

[10] John S. Breese, David Heckerman, and Carl Myers Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI '98: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, July 24-26, 1998, University of Wisconsin Business School, Madison, Wisconsin, USA*, pages 43–52. Morgan Kaufmann, 1998.

[11] Pedro G. Campos, Alejandro Bellogín, Fernando Díez, and J. Enrique Chavarriaga. Simple time-biased knn-based recommendations. In *Proceedings of the Workshop on Context-Aware Movie Recommendation*, CAMRa '10, pages 20–23, New York, NY, USA, 2010. ACM.

[12] Pedro G. Campos, Fernando Díez, and Iván Cantador. Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. *User Model. User-Adapt. Interact.*, 24(1-2):67–119, 2014.

[13] Huanhuan Cao, Enhong Chen, Jie Yang, and Hui Xiong. Enhancing recommender systems under volatile userinterest drifts. In *Proceedings*

of the 18th ACM Conference on Information and Knowledge Management, CIKM 2009, Hong Kong, China, November 2-6, 2009, pages 1257–1266. ACM, 2009.

[14] Paolo Cremonesi, Franca Garzotto, Sara Negro, Alessandro Vittorio Papadopoulos, and Roberto Turrin. Looking for "good" recommendations: A comparative evaluation of recommender systems. In *Human-Computer Interaction - INTERACT 2011 - 13th IFIP TC 13 International Conference, Lisbon, Portugal, September 5-9, 2011, Proceedings, Part III*, volume 6948 of *Lecture Notes in Computer Science*, pages 152–168. Springer, 2011.

[15] Paolo Cremonesi, Antonio Tripodi, and Roberto Turrin. Cross-domain recommender systems. In *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on, Vancouver, BC, Canada, December 11, 2011*, pages 496–503. IEEE Computer Society, 2011.

[16] Abhinandan Das, Mayur Datar, Ashutosh Garg, and ShyamSundar Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 271–280. ACM, 2007.

[17] P.A.D. de Castro, F.O. de Franca, H.M. Ferreira, and F.J. Von Zuben. Applying biclustering to perform collaborative filtering. In *Intelligent Systems Design and Applications, 2007. ISDA 2007. Seventh International Conference on*, pages 421–426, Oct. 2007.

[18] Jeffrey Dean and Sanjay Ghemawat. Mapreduce: Simplified data processing on large clusters. In *6th Symposium on Operating System Design and Implementation (OSDI 2004), San Francisco, California, USA, December 6-8, 2004*, pages 137–150. USENIX Association, 2004.

[19] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407, 1990.

[20] Yi Ding and Xue Li. Time weight collaborative filtering. In *Proceedings of the 2005 ACM CIKM International Conference on Information and Knowledge Management, Bremen, Germany, October 31 - November 5, 2005*, pages 485–492. ACM, 2005.

[21] Yi Ding, Xue Li, and Maria E. Orlowska. Recency-based collaborative filtering. In *Database Technologies 2006, Proceedings of the 17th Australasian Database Conference, ADC 2006, Hobart, Tasmania, Australia, January 16-19 2006*, volume 49 of *CRPIT*, pages 99–107. Australian Computer Society, 2006.

[22] Marcos Aurélio Domingues, Alípio Mário Jorge, and Carlos Soares. The impact of contextual information on the accuracy of existing recommender systems for web personalization. In *Web Intelligence*, pages 789–792. IEEE Computer Society, 2008.

[23] Gideon Dror, Noam Koenigstein, Yehuda Koren, and Markus Weimer. The yahoo! music dataset and kdd-cup '11. *Journal of Machine Learning Research - Proceedings Track*, 18:8–18, 2012.

[24] Ignacio Fernández-Tobías, Iván Cantador, Marius Kaminskas, and Francesco Ricci. Cross-domain recommender systems: A survey of the state of the art. In *CERI '12: Proceedings of the 2nd Spanish Conference on Information Retrieval, Valencia, Spain, June 2012*, 2012.

[25] François Fouss, Alain Pirotte, Jean-Michel Renders, and Marco Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Trans. Knowl. Data Eng.*, 19(3):355–369, 2007.

[26] Simon Funk, 2006. http://sifter.org/ simon/journal/20061211.html [Accessed Jan 2013].

[27] João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. On evaluating stream learning algorithms. *Machine Learning*, 90(3):317–346, 2013.

[28] Zeno Gantner, Steffen Rendle, and Lars Schmidt-Thieme. Factorization models for context-/time-aware movie recommendations. In *Proceedings of the Workshop on Context-Aware Movie Recommendation*, CAMRa '10, pages 14–19, New York, NY, USA, 2010. ACM.

[29] Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. Exploring temporal effects for location recommendation on location-based social networks. In *Seventh ACM Conference on Recommender Systems, RecSys '13, Hong Kong, China, October 12-16, 2013*, pages 93–100. ACM, 2013.

[30] Thomas George and Srujana Merugu. A scalable collaborative filtering framework based on co-clustering. In *ICDM 2005: Proceedings of the*

5th IEEE International Conference on Data Mining, 27-30 November 2005, Houston, Texas, USA, pages 625–628. IEEE Computer Society, 2005.

[31] Kenneth Y. Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Inf. Retr.*, 4(2):133–151, 2001.

[32] Michele Gorgoglione and Umberto Panniello. Including context in a transactional recommender system using a pre-filtering approach: Two real e-commerce applications. In *AINA Workshops*, pages 667–672. IEEE Computer Society, 2009.

[33] Marco Gori and Augusto Pucci. Itemrank: A random-walk based scoring algorithm for recommender engines. In Manuela M. Veloso, editor, *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 2766–2771, 2007.

[34] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004.

[35] William C. Hill, Larry Stead, Mark Rosenstein, and George W. Furnas. Recommending and evaluating choices in a virtual community of use. In *Human Factors in Computing Systems, CHI '95 Conference Proceedings, Denver, Colorado, USA, May 7-11, 1995.*, pages 194–201. ACM/Addison-Wesley, 1995.

[36] Thomas Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):89–115, 2004.

[37] Wenxing Hong, Lei Li, and Tao Li. Product recommendation with temporal dynamics. *Expert Syst. Appl.*, 39(16):12398–12406, 2012.

[38] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*, pages 263–272. IEEE Computer Society, 2008.

[39] Zan Huang, Wingyan Chung, and Hsinchun Chen. A graph model for e-commerce recommender systems. *JASIST*, 55(3):259–274, 2004.

[40] Zan Huang, Wingyan Chung, Thian-Huat Ong, and Hsinchun Chen. A graph-based recommender system for digital library. In *JCDL*, pages 65–73. ACM, 2002.

[41] Geoff Hulten, Laurie Spencer, and Pedro M. Domingos. Mining time-changing data streams. In Doheon Lee, Mario Schkolnick, Foster J. Provost, and Ramakrishnan Srikant, editors, *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, San Francisco, CA, USA, August 26-29, 2001*, pages 97–106. ACM, 2001.

[42] Piotr Indyk. A small approximately min-wise independent family of hash functions. In *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, 17-19 January 1999, Baltimore, Maryland.*, pages 454–456. ACM/SIAM, 1999.

[43] Dietmar Jannach, Lukas Lerche, and MatthÃus Gdaniec. Re-ranking recommendations based on predicted short-term interests - a protocol and first experiment. In *ITWP 2013: Proceedings of the workshop Intelligent Techniques for Web Personalization and Recommender Systems at AAAI 2013, Bellevue, Washington, 2013*, 2013.

[44] Mohammad Khoshneshin and W. Nick Street. Collaborative filtering via euclidean embedding. In *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010*, pages 87–94. ACM, 2010.

[45] Ron Kohavi, Roger Longbotham, Dan Sommerfield, and Randal M. Henne. Controlled experiments on the web: survey and practical guide. *Data Min. Knowl. Discov.*, 18(1):140–181, 2009.

[46] Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

[47] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*, pages 426–434. ACM, 2008.

[48] Yehuda Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009*, pages 447–456. ACM, 2009.

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

[50] Neal Lathia, Stephen Hailes, and Licia Capra. Temporal collaborative filtering with adaptive neighbourhoods. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009, Boston, MA, USA, July 19-23, 2009*, pages 796–797. ACM, 2009.

[51] Neal Lathia, Stephen Hailes, Licia Capra, and Xavier Amatriain. Temporal diversity in recommender systems. In *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2010, Geneva, Switzerland, July 19-23, 2010*, pages 210–217. ACM, 2010.

[52] Dongjoo Lee, Sung Eun Park, Minsuk Kahng, Sangkeun Lee, and Sang goo Lee. Exploiting contextual information from event logs for personalized recommendation. In Roger Y. Lee, editor, *Computer and Information Science*, volume 317 of *Studies in Computational Intelligence*, pages 121–139. Springer, 2010.

[53] Xue Li, Jorge M. Barajas, and Yi Ding. Collaborative filtering on streaming data with interest-drifting. *Intell. Data Anal.*, 11(1):75–87, 2007.

[54] Weiyang Lin, Sergio A. Alvarez, and Carolina Ruiz. Efficient adaptive-support association rule mining for recommender systems. *Data Min. Knowl. Discov.*, 6(1):83–105, 2002.

[55] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, 2003.

[56] Nathan N. Liu, Bin Cao, Min Zhao, and Qiang Yang. Adapting neighborhood and matrix factorization models for context aware recommendation. In *Proceedings of the Workshop on Context-Aware Movie Recommendation*, CAMRa '10, pages 7–13, New York, NY, USA, 2010. ACM.

[57] Nathan Nan Liu, Luheng He, and Min Zhao. Social temporal collaborative ranking for context aware movie recommendation. *ACM TIST*, 4(1):15, 2013.

[58] Nathan Nan Liu, Min Zhao, Evan Wei Xiang, and Qiang Yang. Online evolutionary collaborative filtering. In *Proceedings of the 2010 ACM Conference on Recommender Systems, RecSys 2010, Barcelona, Spain, September 26-30, 2010*, pages 95–102. ACM, 2010.

[59] Xin Liu and Karl Aberer. Towards a dynamic top-n recommendation framework. In *Eighth ACM Conference on Recommender Systems, RecSys '14, Foster City, Silicon Valley, CA, USA - October 06 - 10, 2014*, pages 217–224. ACM, 2014.

[60] Pawel Matuszyk and Myra Spiliopoulou. Selective forgetting for incremental matrix factorization in recommender systems. In Saso Dzeroski, Pance Panov, Dragi Kocev, and Ljupco Todorovski, editors, *Discovery Science - 17th International Conference, DS 2014, Bled, Slovenia, October 8-10, 2014. Proceedings*, volume 8777 of *Lecture Notes in Computer Science*, pages 204–215. Springer, 2014.

[61] Pawel Matuszyk, João Vinagre, Myra Spiliopoulou, Alípio Mário Jorge, and João Gama. Forgetting methods for incremental matrix factorizatin in recommender systems. In *SAC 2015: Proceedings of the 30th ACM SIGAPP Symposium on Applied Computing, April 13-17, 2015, Salamanca, Spain*, pages 947–953. ACM, 2015.

[62] Brian McFee, Thierry Bertin-Mahieux, Daniel P. W. Ellis, and Gert R. G. Lanckriet. The million song dataset challenge. In *Proceedings of the 21st World Wide Web Conference, WWW 2012, Lyon, France, April 16-20, 2012 (Companion Volume)*, pages 909–916. ACM, 2012.

[63] Sean M. McNee, John Riedl, and Joseph A. Konstan. Being accurate is not enough: how accuracy metrics have hurt recommender systems. In *Extended Abstracts Proceedings of the 2006 Conference on Human Factors in Computing Systems, CHI 2006, Montréal, Québec, Canada, April 22-27, 2006*, pages 1097–1101. ACM, 2006.

[64] Sung-Hwan Min and Ingoo Han. Detection of the customer time-variant pattern for improving recommender systems. *Expert Systems with Applications*, 28(2):189 – 199, 2005.

[65] Bamshad Mobasher, Honghua Dai, Tao Luo, and Miki Nakagawa. Effective personalization based on association rule discovery from web usage data. In *3rd International Workshop on Web Information and Data Management (WIDM 2001), Friday, 9 November 2001, In Conjunction with ACM CIKM 2001, Doubletree Hotel Atlanta-Buckhead, Atlanta, Georgia, USA. ACM, 2001*, pages 9–15. ACM, 2001.

[66] Olfa Nasraoui, Jeff Cerwinske, Carlos Rojas, and Fabio A. González. Performance of recommendation systems in dynamic streaming environments. In *Proceedings of the Seventh SIAM International Conference on Data Mining, April 26-28, 2007, Minneapolis, Minnesota, USA*. SIAM, 2007.

[67] Olfa Nasraoui, Cesar Cardona Uribe, Carlos Rojas Coronel, and Fabio A. González. Tecno-streams: Tracking evolving clusters in noisy data streams with a scalable immune system learning model. In *Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003), 19-22 December 2003, Melbourne, Florida, USA*, pages 235–242. IEEE Computer Society, 2003.

[68] Róbert Pálovics, András A. Benczúr, Levente Kocsis, Tamás Kiss, and Erzsébet Frigó. Exploiting temporal influence in online recommendation. In *Eighth ACM Conference on Recommender Systems, RecSys '14, Foster City, Silicon Valley, CA, USA - October 06 - 10, 2014*, pages 273–280. ACM, 2014.

[69] Rong Pan, Yunhong Zhou, Bin Cao, Nathan Nan Liu, Rajan M. Lukose, Martin Scholz, and Qiang Yang. One-class collaborative filtering. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*, pages 502–511. IEEE Computer Society, 2008.

[70] Umberto Panniello, Alexander Tuzhilin, and Michele Gorgoglione. Comparing context-aware recommender systems in terms of accuracy and diversity. *User Model. User-Adapt. Interact.*, 24(1-2):35–65, 2014.

[71] Umberto Panniello, Alexander Tuzhilin, Michele Gorgoglione, Cosimo Palmisano, and Anto Pedone. Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems. In *Proceedings of the 2009 ACM Conference on Recommender Systems, RecSys 2009, New York, NY, USA, October 23-25, 2009*, pages 265–268. ACM, 2009.

[72] A. Paterek. Improving regularized singular value decomposition for collaborative filtering. In *Proceedings of KDD Cup and Workshop*, volume 2007, pages 5–8, 2007.

[73] Pearl Pu, Li Chen, and Rong Hu. Evaluating recommender systems from the user's perspective: survey of the state of the art. *User Model. User-Adapt. Interact.*, 22(4-5):317–355, 2012.

[74] Dimitrios Rafailidis and Alexandros Nanopoulos. Modeling the dynamics of user preferences in coupled tensor factorization. In *Eighth ACM Conference on Recommender Systems, RecSys '14, Foster City, Silicon Valley, CA, USA - October 06 - 10, 2014*, pages 321–324. ACM, 2014.

[75] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: bayesian personalized ranking from implicit feedback. In *UAI 2009, Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, June 18-21, 2009*, pages 452–461. AUAI Press, 2009.

[76] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 811–820. ACM, 2010.

[77] Steffen Rendle and Lars Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *Proceedings of the Third International Conference on Web Search and Web Data Mining, WSDM 2010, New York, NY, USA, February 4-6, 2010*, pages 81–90. ACM, 2010.

[78] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *CSCW '94, Proceedings of the Conference on Computer Supported Cooperative Work, October 22-26, 1994, Chapel Hill, NC, USA*, pages 175–186, 1994.

[79] Francesco Ricci, Adriano Venturini, Dario Cavada, Nader Mirzadeh, Dennis Blaas, and Marisa Nones. Product recommendation with interactive query management and twofold similarity. In *Case-Based Reasoning Research and Development, 5th International Conference on Case-Based Reasoning, ICCBR 2003, Trondheim, Norway, June 23-26, 2003, Proceedings*, volume 2689 of *Lecture Notes in Computer Science*, pages 479–493. Springer, 2003.

[80] Alan Said and Alejandro Bellogín. Comparative recommender system evaluation: benchmarking recommendation frameworks. In *Eighth ACM Conference on Recommender Systems, RecSys '14, Foster City, Silicon Valley, CA, USA - October 06 - 10, 2014*, pages 129–136. ACM, 2014.

[81] Alan Said, Shlomo Berkovsky, and Ernesto W. De Luca. Putting things in context: Challenge on context-aware movie recommendation. In *Proceedings of the Workshop on Context-Aware Movie Recommendation*, CAMRa '10, pages 2–6, New York, NY, USA, 2010. ACM.

[82] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pages 1257–1264. Curran Associates, Inc., 2007.

[83] Ruslan Salakhutdinov and Andriy Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In William W. Cohen, Andrew McCallum, and Sam T. Roweis, editors, *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008), Helsinki, Finland, June 5-9, 2008*, volume 307 of *ACM International Conference Proceeding Series*, pages 880–887. ACM, 2008.

[84] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. Analysis of recommendation algorithms for e-commerce. In *ACM Conference on Electronic Commerce*, pages 158–167, 2000.

[85] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001*, pages 285–295. ACM, 2001.

[86] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John T. Riedl. Application of dimensionality reduction in recommender system - a case study. In *WEBKDD'2000: Web Mining for E-Commerce – Challenges and Opportunities August 20, 2000, Boston, MA*, 2000.

[87] Guy Shani and Asela Gunawardana. Evaluating recommendation systems. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 257–297. Springer, 2011.

[88] Guy Shani, David Heckerman, and Ronen I. Brafman. An mdp-based recommender system. *Journal of Machine Learning Research*, 6:1265–1295, 2005.

[89] Upendra Shardanand and Pattie Maes. Social information filtering: Algorithms for automating "word of mouth". In *Human Factors in*

Computing Systems, CHI '95 Conference Proceedings, Denver, Colorado, USA, May 7-11, 1995., pages 210–217. ACM/Addison-Wesley, 1995.

[90] Yue Shi, Martha Larson, and Alan Hanjalic. Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Comput. Surv.*, 47(1):3, 2014.

[91] Panagiotis Symeonidis, Alexandros Nanopoulos, Apostolos Papadopoulos, and Yannis Manolopoulos. Nearest-biclusters collaborative filtering with constant values. In *Advances in Web Mining and Web Usage Analysis, 8th International Workshop on Knowledge Discovery on the Web, WebKDD 2006, Philadelphia, PA, USA, August 20, 2006, Revised Papers*, volume 4811 of *Lecture Notes in Computer Science*, pages 36–55. Springer, 2006.

[92] G. Takacs, I. Pilaszy, B. Nemeth, and D. Tikk. On the gravity recommendation system. In *Proceedings of KDD Cup and Workshop*, volume 2007, 2007.

[93] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. Scalable collaborative filtering approaches for large recommender systems. *Journal of Machine Learning Research*, 10:623–656, 2009.

[94] Amit Tiroshi, Shlomo Berkovsky, Mohamed Ali Kâafar, David Vallet, and Tsvi Kuflik. Graph-based recommendations: Make the most out of social data. In Vania Dimitrova, Tsvi Kuflik, David Chin, Francesco Ricci, Peter Dolog, and Geert-Jan Houben, editors, *User Modeling, Adaptation, and Personalization - 22nd International Conference, UMAP 2014, Aalborg, Denmark, July 7-11, 2014. Proceedings*, volume 8538 of *Lecture Notes in Computer Science*, pages 447–458. Springer, 2014.

[95] João Vinagre and Alípio Mário Jorge. Forgetting mechanisms for scalable collaborative filtering. *J. Braz. Comp. Soc.*, 18(4):271–282, 2012.

[96] João Vinagre, Alípio Mário Jorge, and João Gama. Evaluation of recommender systems in streaming environments. In *Proceedings of the Workshop on Recommender Systems Evaluation: Dimensions and Design in conjunction with the 8th ACM Conference on Recommender Systems (RecSys 2014), Foster City, CA, USA, October 10, 2014.*, pages 1–6, 2014.

[97] João Vinagre, Alípio Mário Jorge, and João Gama. Collaborative filtering with recency-based negative feedback. In *SAC 2015: Proceedings of the 30th ACM SIGAPP Symposium on Applied Computing, April 13-17, 2015, Salamanca, Spain*, pages 963–965. ACM, 2015.

[98] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(1):69–101, 1996.

[99] Liang Xiang, Quan Yuan, Shiwan Zhao, Li Chen, Xiatian Zhang, Qing Yang, and Jimeng Sun. Temporal recommendation on graphs via long- and short-term preference fusion. In Bharat Rao, Balaji Krishnapuram, Andrew Tomkins, and Qiang Yang, editors, *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, July 25-28, 2010*, pages 723–732. ACM, 2010.

[100] Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff G. Schneider, and Jaime G. Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2010, April 29 - May 1, 2010, Columbus, Ohio, USA*, pages 211–222. SIAM, 2010.

[101] Li'ang Yin, Yongqiang Wang, and Yong Yu. Collaborative filtering via temporal euclidean embedding. In *Web Technologies and Applications - 14th Asia-Pacific Web Conference, APWeb 2012, Kunming, China, April 11-13, 2012. Proceedings*, volume 7235 of *Lecture Notes in Computer Science*, pages 513–520. Springer, 2012.

[102] Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun, and Nadia Magnenat-Thalmann. Time-aware point-of-interest recommendation. In *The 36th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '13, Dublin, Ireland - July 28 - August 01, 2013*, pages 363–372. ACM, 2013.

[103] Andrew Zimdars, David Maxwell Chickering, and Christopher Meek. Using temporal data for making recommendations. In *UAI '01: Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, University of Washington, Seattle, Washington, USA, August 2-5, 2001*, pages 580–588. Morgan Kaufmann, 2001.