User Model User-Adap Inter (2014) 24:67–119
DOI 10.1007/s11257-012-9136-x

ORIGINAL PAPER

Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols

Pedro G. Campos · Fernando Díez · Iván Cantador

Received: 22 March 2012 / Accepted in revised form: 4 December 2012 /
Published online: 15 February 2013

Abstract Exploiting temporal context has been proved to be an effective approach to improve recommendation performance, as shown, e.g. in the Netflix Prize competition. Time-aware recommender systems (TARS) are indeed receiving increasing attention. A wide range of approaches dealing with the time dimension in user modeling and recommendation strategies have been proposed. In the literature, however, reported results and conclusions about how to incorporate and exploit time information within the recommendation processes seem to be contradictory in some cases. Aiming to clarify and address existing discrepancies, in this paper we present a comprehensive survey and analysis of the state of the art on TARS. The analysis show that meaningful divergences appear in the evaluation protocols used—metrics and methodologies. We identify a number of key conditions on offline evaluation of TARS, and based on these conditions, we provide a comprehensive classification of evaluation protocols for TARS. Moreover, we propose a methodological description framework aimed to make the evaluation process fair and reproducible. We also present an empirical study on the impact of different evaluation protocols on measuring relative performances of well-known TARS. The results obtained show that different uses of the above evaluation

P. G. Campos (✉)
Department of Information Systems, Universidad del Bío-Bío,
Concepción, Chile
e-mail: pgcampos@ubiobio.cl

P. G. Campos, F. Díez, I. Cantador
Department of Computer Science, Universidad Autónoma de Madrid,
Madrid, Spain

F. Díez
e-mail: fernando.diez@uam.es

I. Cantador
e-mail: ivan.cantador@uam.es

conditions yield to remarkably distinct performance and relative ranking values of the recommendation approaches. They reveal the need of clearly stating the evaluation conditions used to ensure comparability and reproducibility of reported results. From our analysis and experiments, we finally conclude with methodological issues a robust evaluation of TARS should take into consideration. Furthermore we provide a number of general guidelines to select proper conditions for evaluating particular TARS.

Keywords Time-aware recommender systems · Context-aware recommender systems · Evaluation methodologies · Survey

1 Introduction

Recommender systems (RS) aim to help users with information access and retrieval applications when large collections of items are involved. In general, they work by means of suggesting those items that should be the most appealing ones to the users based on their past personal preferences. In such process, exploiting the *context* (e.g. location, time, weather, device and mood) in which users express their preferences has been proven very valuable for increasing the performance of recommendations (Adomavicius and Tuzhilin 2011; Baltrunas and Ricci 2013).

Among existing contextual dimensions, time information can be considered as one of the most useful ones. It facilitates tracking the evolution of user preferences (Xiang et al. 2010), enabling e.g. to identify periodicity in user habits and interests (Campos et al. 2011a), which is a key input for many context-aware RS (CARS) (Baltrunas and Amatriain 2009; Panniello et al. 2009b). It may also lead to significant improvements on recommendation accuracy, as found by the winning team of the Netflix Prize competition (Koren 2009b). Moreover, temporal context information is in general easy to collect without additional user efforts and strict device requirements. Due to these benefits, recent years have been prolific in the investigation and development of time-aware RS (TARS), that is, CARS that exploit the time dimension for both user modeling and recommendation strategies.

However, despite the benefits of TARS, some studies have shown divergences on the ground assumption in which recommendation models are built, casting doubt on the generalization of time-aware recommendation capabilities. Ding and Li (2005) obtained recommendation improvements by means of applying an exponential decay rate on old ratings in order to give more importance to recent ratings, arguing the latter should better reflect the users' current tastes. Koren (2009b), on the other hand, showed that better prediction results on the Netflix Prize dataset can be obtained when no time decay is considered. Baltrunas and Amatriain (2009) built contextual *micro-profiles* representing a user's song-listening behavior in different contexts, and utilized such profiles as input for computing contextualized music recommendations. Although recommendation improvements were obtained by using time-dependent data partitioning such as {morning, evening} and {weekend, workday}, the highest improvement was obtained when considering the scarce {even hours, odd hours} partitioning which, in the words of the authors, corresponds to a "meaningless split", and calls for further research. Finally, Lathia et al. (2009) showed that improvements obtained by some non-contextualized algorithms on the Netflix Prize probe set[1] do not hold when computing predictions on an iterative basis by strictly using past ratings to predict future ratings—the actual setting for a real-world recommender system.

Despite the fact that a number of reasons could be enumerated for explaining such contradictory findings (e.g. different application domains, item characteristics and contextualization schemas for time information), we believe evaluation plays a prominent role. The existence of multiple evaluation methodologies and metrics, each of them with distinct assumptions and purposes, makes it easy to find an evaluation protocol suitable for a particular algorithmic approach, but ineligible or retributive for others. Problems that arise from this situation thus represent an increasing impediment to fairly compare results and conclusions reported in different studies (Bellogín et al. 2011), and make the selection of the best recommendation solution for a given task more difficult (Gunawardana and Shani 2009).

Aiming to shed light and better understand the contrasts and impact of the different protocols used for evaluating TARS, in this paper we revise most of the work existing in the literature on the topic, and characterize the evaluation protocols used, by identifying and describing a set of key methodological conditions that the performed evaluations are driven by. These conditions address a number of general methodological issues to be faced in the experimental design of an offline evaluation of TARS, and are mostly related to the training-test data splitting process, namely rating ordering for training and test sets, size of those sets, base data used for building the splits, and cross-validation methods. Based on these conditions, we propose a methodological description framework aimed to make the evaluation process fair and reproducible under different circumstances.

We also report an empirical study on the impact that some key conditions have on the performance of well-known TARS in the movie and music recommendation domains. The results obtained show that different uses of the above evaluation conditions lead to distinct rankings of the recommendation approaches for certain metrics. This reveals the need of clearly stating the used evaluation conditions to ensure comparability and reproducibility of results from different TARS.

Finally, we provide a number of general guidelines to select proper conditions for evaluating particular TARS. We believe the results and guidelines presented herein could help researchers and practitioners to conduct robust evaluations of TARS.

In summary, the main contributions of this paper are:

– A comprehensive survey of TARS-related literature. From this survey we identify key conditions used in offline evaluation of TARS, and based on such conditions we classify the revised TARS.
– An analysis of important methodological issues which a robust evaluation of TARS should take into account. A proper understanding and treatment of these issues could help in conducting fair evaluations of new TARS, and may facilitate comparisons between published performance results. From this analysis we provide a number of guidelines regarding the selection of appropriate conditions to evaluate TARS.

1 A public test dataset used for evaluating performance of participants in the Netflix Prize competition.

- A methodological description framework aimed to facilitate the description and adoption of evaluation protocols, and make the evaluation process fair and reproducible.
- An empirical comparison of results obtained from different TARS evaluation protocols aimed to assess the influence of evaluation conditions on measured performance results, by means of accuracy and ranking metrics, and more recent metrics of novelty and diversity.

The remainder of the paper is organized as follows: in Sect. 2 we formulate the problem of evaluating time-aware recommendation; in Sect. 3 we describe and classify an extensive number of TARS proposed in the literature; in Sect. 4 we identify sources of divergence between TARS evaluation protocols, and establish related methodological issues; in Sect. 5 we introduce key conditions addressing the above methodological issues and propose a comprehensive framework for describing such conditions; in Sect. 6 we report empirical results comparing the effect of some of such conditions, and in Sect. 7 we provide a summary and classification of the revised TARS according to the evaluation conditions identified. From such classification and the experimental results obtained, we provide general guidelines to select proper evaluation conditions for TARS, and state open questions on the topic. Finally, in Sect. 8 we present a number of conclusions drawn from our study.

2 Problem statement

2.1 The recommendation problem

The recommendation problem consists of suggesting items that should be the most appealing ones to a user according to her preferences. In the literature several types of RS have been proposed, varying, e.g. in the types of data used, and in the methods with which recommendations are generated. Burke (2007) distinguishes four main recommendation techniques: *content-based* techniques (CB), which suggest to the target user similar items to those preferred by her in the past, *collaborative filtering* techniques (CF), which suggest items preferred by users with similar tastes to the target user's, *demographic* techniques, which exploit the users' demographics for generating item recommendations, and *knowledge-based* techniques, which exploit specific domain knowledge about the items to recommend. Additionally, it is possible to distinguish *hybrid* recommenders, which combine two or more of the above techniques in order to overcome some of their limitations.

In a widely used formulation (Adomavicius and Tuzhilin 2005), the recommendation problem relies on the notion of *ratings* as a mechanism to capture user preferences for different items. Let U be a set of users, and let I be a set of items. A recommender system models a function F that computes a predicted rating $\hat{r}_{u,i}$ for an unknown rating $r_{u,i}$ which user $u \in U$ would assign to item $i \in I$:

$$F : U \times I \rightarrow R$$

where R denotes a totally ordered set (e.g. non-negative integer or real numbers in a particular range) of allowed rating values.

The rating values express a scale of the users' preferences for the items, or are related to the users' consumption of items. In the former case the ratings are usually considered as *explicit ratings*, since they are explicitly provided by the users. In the latter case, on the other hand, the ratings are commonly considered as *implicit ratings* because they are collected without explicit user feedback. In this paper we refer to both forms simply as ratings, except when the distinction between them is needed.

Alternatively, the recommendation problem can be modeled as the task of finding relevant (appealing) items for the target user (Sarwar et al. 2000). This task is consistent with the use of RS in many applications where a system does not predict ratings, but delivers lists of items that may be relevant for the user (Shani and Gunawardana 2011). In this case, ratings can be interpreted as a measure of relevance (score), and thus, those items scored over a certain threshold value can be considered as relevant.

For either of these two formulations, the recommendation problem can be reduced to solve a rating prediction problem, which consists of predicting unknown ratings for pairs (u, i) by providing an approximation of the function F (Adomavicius and Tuzhilin 2005). In this context, when a RS is required to provide an item recommendation, it could return rating predictions for a particular set of items unknown to the target user—the *rating prediction task*—or a list of top ranked items the user may prefer—the *top-N recommendation task* (Shani and Gunawardana 2011). In the latter case rating predictions[2] are generally used to rank the items, and select (recommend) the top ranked ones.

In both tasks CB and CF recommendation techniques exploit the target user's ratings to compute F. CB systems take advantage of available item descriptions for looking for items whose content is similar to that of items positively rated by the target user. CF systems make use of other users' ratings to detect items rated similarly to those positively rated by the target user. There are many variants of both approaches in the literature, and also hybrid combinations aimed to solve weaknesses from each one—(Adomavicius and Tuzhilin 2005) and (Burke 2002) provide extensive surveys on these topics.

In this paper we focus on CF techniques, since they have been widely used to take advantage of contextual (and, particularly, temporal) information associated to user ratings. We thus assume that user preferences are represented in the form of user profiles that contain ratings with contextual information. We denote the set of user profiles available to the RS, i.e. the collected (known) ratings, as the rating matrix $M = \left\{ r_{u,i} | u \in U, i \in I, r_{u,i} \neq \emptyset \right\}$.

2.2 Incorporating context into the recommendation problem

Context is a multifaceted concept that has been studied across different research disciplines, and thus has been defined in multiple ways (Adomavicius and Tuzhilin 2011; Hussein et al. 2014). According to Dey (2001), "context is any information that can be used to characterize the situation of an entity", where in the case of a RS an entity can

[2] In this case it may be more precise to talk about score prediction, but for the sake of simplicity we will refer to this also as rating prediction.

be a user, an item, or an experience the user is evaluating (Baltrunas 2011). Hence, any information regarding the situation in which a user experiences an item—e.g. location, time, weather, device, and mood—can be considered as context.

The importance of including context in the recommendation process can be observed from a practical viewpoint through a classic, simple example in the tourism domain: although a user may love skiing, recommending her a ski resort during summer is questionable. It could be detrimental for the user's trust in the system if interpreted as an 'out of context' recommendation. In fact, in a user study comparing RS that use and do not use contextual information, Gorgoglione et al. (2011) found that the former provide more user trust in their item recommendations. Moreover, they detected that trust affect purchasing behavior, and observed that RS exploiting context information increase user's trust and levels of sales.

The recommender systems that exploit any of the above types of information are called context-aware RS (CARS) (Adomavicius and Tuzhilin 2011). Extending the definition of recommendation problem given in the previous section, Adomavicius et al. (2005) pose a generic model for CARS by incorporating additional dimensions of contextual information C into F:

$$F : U \times I \times C \rightarrow R$$

This model assumes that the context can be *known* and represented as a set of contextual dimensions $C_1, C_2, \ldots, C_n \in C$ (Dourish 2004), where each dimension C_i may have its own type and range of values. Moreover, a dimension C_i may have different representations reflecting the complex nature of its contextual information (Adomavicius and Tuzhilin 2011). For instance, the contextual dimension *location* can be defined as a plain list of values {home, abroad}, or can be defined by means of a hierarchical structure such as room → building → neighborhood → city → country. In general, according to Palmisano et al. (2008) and Adomavicius et al. (2005), each contextual dimension $C_i \in C$ can be defined as a set of attributes $C_i = \left\{C_i^1, C_i^2, \ldots, C_i^q\right\}$ that may be independent or related through some kind of structure.

According to Adomavicius and Tuzhilin (2011), CARS can be classified as *contextual pre-filtering*, *contextual post-filtering*, and *contextual modeling* systems. In contextual pre-filtering the target recommendation context—i.e. the context in which the target user expects to consume the recommended items—is used to filter user profile data relevant to such context before the rating prediction computation. In contextual post-filtering rating predictions are adjusted according to the target context after being computed (on entire user profiles). In both cases traditional non-contextualized recommendation algorithms can be employed, as the contextualization involves an independent pre- or post-processing computation. On the other hand, contextual modeling incorporates context information directly into the model used to estimate rating predictions.

2.3 Specifying time context in recommender systems

Among existing contextual dimensions, the *time dimension*—i.e., the contextual attributes related to time, such as *time of the day*, *day of the week*, and *season of the* *year*—has the advantage of being easy to collect, since almost any system can record item rating/consumption timestamps. Moreover, as noted in a previous example, the time dimension can serve as a valuable input for improving recommendation quality (Baltrunas and Amatriain 2009; Panniello et al. 2009b). Since our aim is to analyze evaluation practices related to this dimension, from now on we focus on *time context*.

Time-aware recommender systems can be considered a specialized type of CARS. Their main characteristic is the usage of time context information at some stage of the rating prediction process, being able to provide differentiated recommendations depending on the target recommendation time (i.e. the desired time for item usage or consumption, which may be different from the recommendation delivery time), according, e.g. to the preferences expressed by the users at similar time contexts in the past. Thus, the general formulation of context-dependent rating prediction can be particularized for the time dimension of context, T, as follows:

$$F : U \times I \times T \rightarrow R$$

where T can be represented in several ways. According to Merriam-Webster[3], time is defined as “a non-spatial continuum that is measured in terms of events that succeed one another from past through present to future”. From this definition, it follows that it is possible to establish an order between time events (or time values), e.g. night is after evening, and Monday is before Tuesday. A second sense of time is “the measured or measurable period during which an action, process, or condition exists or continues”. Given the huge differences in duration of various processes (consider, e.g. the duration of a movie, and the human lifetime), several time units have been defined, e.g. hours, days, months and years (Whitrow 1988), forming a hierarchy of time units (e.g. a day “is formed” by 24 h, and a week “is formed” by 7 days). This hierarchical structure and the fact that time is a continuum, lead to a cyclic conception of time where its values repeat periodically (e.g. 12 a.m., 1 a.m., 2 a.m., … 11 p.m., 12 a.m., …, and Monday, Tuesday, …, Sunday, Monday, …).

Due to this flexibility in the time conception and measurement, different representations of time context information can be used. For example, time may be modeled as a continuous variable whose values are the specific times at which items are rated/consumed (e.g. a *timestamp* like “*January 1st, 2000 at 00:00:00*”). Another option is to specify categorical values, regarding time periods of interest in the recommendation domain at hand. For instance, in the tourism domain, a seasonal variable like *seasonOfTheYear*={*hot_season*, *cold_season*} may be convenient, whereas in the music or movies domain, the variable *timeOfTheWeek*={*workday*, *weekend*} may have more sense. A hierarchical modeling may also be used, enabling to control the degree of granularity of the time context information (e.g. *dayOfTheWeek*=$\{Monday, Tuesday, \ldots, Sunday\} \quad \rightarrow$ *timeOfTheWeek*). In this sense, it has to be noted that storing the *timestamp* of ratings is the most flexible option, since it enables exploiting diverse representations of time context, including both continuous and (may be overlapping) categorical values.

3 http://www.merriam-webster.com/dictionary/time

In general, time-aware recommendation models exploit collected time information related to explicit past user preferences, e.g. the timestamps associated to ratings. However, other sources of time information could be collected and exploited. Examples of interesting events are the time of the items' consumptions / purchases, the time of the items' incorporation into the system's catalog, and the time of the users' registration into the system's community. We denote by $t(e)$ the function returning the time associated to an event e.

2.4 Evaluating recommender systems

The evaluation of recommender systems can be performed either *online* or *offline* (Shani and Gunawardana 2011). In *online evaluation* real users interactively test one or more deployed recommendation functionalities or models, and in general, empirical comparisons of user satisfaction for different item recommendations are conducted by means of A/B tests (Kohavi et al. 2009). Online evaluation thus provides valuable information about user preferences and satisfaction regarding recommendations provided. However, it is not always a feasible option due to the need of having an operative system deployed, and a high cost, requiring a large number of people using the system. In *offline evaluation*, on the other hand, past user behavior recorded in a database is used to assess the system's performance, by testing whether recommendations match the users' declared interests. Thanks to historical data availability, offline evaluation brings a low cost and easy to reproduce experimental environment for testing new algorithms and distinct settings of a particular algorithm. Because of these advantages, the majority of past work on TARS, including that presented herein, has been focused on offline evaluation protocols.

In any evaluation protocol there are two fundamental components: the *evaluation metrics*, which define what to assess, and the *evaluation methodologies*, which define how to assess. In the recommender systems field, despite the fact that certain metrics have been accepted and are commonly used (Herlocker et al. 2004; Gunawardana and Shani 2009), there is no consensus on the methodologies used (Bellogín et al. 2011).

In general, a recommendation model is built (trained) with available user data, and afterwards its ability to deliver *good*[4] recommendations is assessed somehow with additional (test) user data. From this, in an offline evaluation scenario, we have to simulate the users' actions *after* receiving recommendations. This is achieved by splitting the set of available ratings into a *training set* (Tr)—which serves as historical data to learn the users' preferences—and a *test set* (Te)—which is considered as knowledge about the users' decisions when faced with recommendations, and is commonly referred to as *ground truth* data. Since test data should not be accessible during the model building/learning process, in general, the only restriction that must hold is to avoid pairwise user-item rating overlaps between training and test sets, i.e. $Tr \cap Te = \emptyset$.

[4] There is no general definition of what *good* recommendations are. Nonetheless, a commonly used approach is to establish the quality (goodness) of recommendations by computing different metrics that assess various desired characteristics of a RS output.

If one or more ratings from a user u are selected for Tr, u becomes part of the set of training users U_{Tr}. Similarly, if one or more ratings from a user u are selected for Te, u becomes part of the set of test users U_{Te}. In this sense, a single user may have some ratings in Tr and some ratings in Te, and thus belongs to both U_{Tr} and U_{Te}. Finally, we denote by Tr_u and Te_u the set of training and test ratings of user u, respectively, and by M_u the set of all the ratings collected from u (M denotes the matrix with the ratings of all the users and items).

2.5 Evaluating time-aware recommender systems

The availability of rating timestamps can be considered as the source of major differences in the evaluation of TARS compared to other types of RS. In particular, the ability to order ratings according to timestamps before training-test data splitting lets define such data splitting in various ways. The possibilities range from maintaining a random split of data—as mostly done in time-unaware RS evaluation—to a strict time-aware split. In the latter, a test rating $r_{Te} \in Te$ has a timestamp posterior to any training rating $r_{Tr} \in Tr$, i.e. a time-dependent order of rating data is used: $\forall r_{Te}, r_{Tr}, t(r_{Te}) > t(r_{Tr})$. We note that this case is the most similar to a real-world setting, where a RS may only use past recorded data in order to estimate future user preferences.

The methodologies used for TARS evaluation make use of several intermediate approaches that highly differ from one work to another, and existing differences may have a significant effect on the assessment of TARS performance. As a matter of fact, in many methodologies time-dependent order of data is not performed to build Te, which may represent an unfair setting for TARS with respect to time-unaware methods unable to exploit time information. We thus aim to get a deeper understanding of the impact of existing differences in TARS evaluation, by means of identifying and analyzing the key conditions that drive evaluation methodologies. Prior to digging into these conditions, in the next section we survey the usage of the time dimension in the CARS literature.

3 Time as a contextual dimension in recommender systems

As explained in Sect. 2, time-aware recommender systems can be considered as specialized CARS focusing on exploiting contextual information in the form of time. A wide range of approaches on modeling and exploiting such information have been proposed. In order to get a comprehensive landscape on existing approaches, in this section we review and classify the literature on TARS.

First approaches considering time information in RS date back to 2001. Zimdars et al. (2001) treated the CF recommendation problem as a time series prediction problem, encoding the time-dependent order of the data. In a more generic perspective, Adomavicius and Tuzhilin (2001) proposed the use of a multi-dimensional representation of RS in order to deal with contextual information, including the time dimension among others.

Despite this early work, the topic recalled the attention of researchers recently; Adomavicius et al. (2005) and Koren (2009b) have had a strong influence in the field, which is producing an increasing number of RS approaches exploiting some form of time information. For the sake of simplicity, we roughly group related work by two characteristics, the type of CF approach used, and the treatment given to time information.

Following the common classification of CF systems, we distinguish between *heuristic-based* (or memory-based) approaches and *model-based* approaches (Adomavicius and Tuzhilin 2005). Heuristic-based algorithms essentially compute F from the entire collection of user profiles by means of certain heuristics. That is, they compute rating predictions directly from all known ratings by means of a particular mathematical expression. For instance, in the user-based CF approach (Herlocker et al. 1999) the collection of ratings is used to determine similarity between the users' preferences, and then, the ratings of the most similar users (the neighborhood) to the target user are used to estimate unknown ratings of the target user. Model-based approaches, on the other hand, learn a predictive model from the collection of known ratings, which represents an approximation of F. This requires a prior learning process where the model is built, but thereafter the model directly generates rating predictions, which leads to a faster response at recommendation time. A widely used example of model-based CF is the Matrix Factorization (MF) algorithm (Koren 2009b). MF models user-item interactions in a latent factor space, where latent factors are used to efficiently predict unknown ratings.

Regarding the usage of time information, we identify approaches that adapt rating predictions depending on the target recommendation time, and which represent time as a continuous contextual variable —*continuous time-aware approaches*—or as categorical contextual variables—*categorical time-aware approaches*. Additionally, there are approaches that exploit time information in a more subtle way, without differentiating rating prediction according to the target time, but rather adjusting some parameters or data dynamically—*time adaptive approaches*. In the following paragraphs we explain the fundamentals of these approaches.

3.1 Heuristic-based approaches

These approaches include continuous time-aware heuristics, categorical time-aware heuristics, and time-adaptive heuristics. All of them, differently to model-based approaches, directly use the rating collection for computing predictions.

3.1.1 Continuous time-aware heuristics

In this type of TARS, heuristics for computing rating predictions incorporate continuous time information into their analytic formulas. In the general formulation of the CF heuristics, the rating prediction of item i for user u consists of an aggregation of ratings of other (usually the most similar) users:

$$F(u, i) = \underset{v \in N(u)}{\operatorname{aggr}} \; r_{v,i} \tag{1}$$

where $N(u)$ represents the set of users most similar to (*neighbors* of) u, and similarity between users is computed by means of comparing user profiles with some metric, e.g., the Pearson's correlation (Adomavicius and Tuzhilin 2005). This approach is generally referred to as user-based k-nearest neighbors (kNN), where k defines the number of neighbors to consider in formula (1). Alternatively, the rating prediction can be computed as an aggregation of ratings of items similar to the target item, which is referred to as item-based kNN.

In continuous time-aware heuristics, time information T is represented as a continuous variable. The rating prediction becomes an explicit function of the target recommendation time, $\hat{r} = F(u, i, t)$. Note that this formulation lets use a target time distinct from the current time; the recommendation may be required for a time different from the requesting time, e.g. "what movie could I see tomorrow?"

A common approach in this type of TARS is to differently weight ratings according to their "age" (time distance) with respect to the target time, generally in the form of an increasing penalization on older data, under the assumption that more recent ratings better reflect current user tastes and interests. In user-based kNN this leads to:

$$F(u, i, t) = \underset{v \in N(u)}{\text{aggr}}\ r_{v,i} \cdot w_t(t(r_{v,i}), t) \tag{2}$$

where $t \in T$ denotes the target (recommendation) time[5], and $w_t(\cdot)$ returns a time-dependent weight. For instance, Ding and Li (2005) proposed an exponential time decay weight $w_t\left(t\left(r_{v,i}\right), t\right) = e^{-\lambda \cdot (t - t(r_{v,i}))}$, being λ a constant value. The time weight w_t can also be used to estimate the similarity between users or items. For example, Hermann (2010) used the weight $w_t\left(t\left(r_{u,i}\right), t\left(r_{u,j}\right), t\right) = 1/\left(\left|t\left(r_{u,i}\right) - t\left(r_{u,j}\right)\right| + \left|\min\left(t\left(r_{u,i}\right), t\left(r_{u,j}\right)\right) - t\right|\right)$ as a measure of time similarity between items consumed by user u.

The most extreme case of this approach is that in which $w_t(\cdot)$ is 0 (i.e. the data is discarded) if the time distance between $t(r)$ and t is over some specified threshold (Gordea and Zanker 2007; Campos et al. 2010). This is sometimes referred to as instance selection, time window, or time truncation.

3.1.2 Categorical time-aware heuristics

In this type of TARS, the time dimension T is modeled as one or more discrete variables $T^1, T^2, \cdots T^n \in T$ that let treat ratings differently depending on their contextual values. Under this formulation the possible target time is one of the values of the contextual variables, and no references to time ordering can be made (e.g. it is possible to ask "what movies may I see on weekends?", but not "what movies may I see *next* weekend?"). Generic context-based algorithms exploiting time by *contextual pre-filtering* and *contextual post-filtering* strategies belong to this category.

A particular time context is represented as $t = \bigcup_j t^j \mid t^j \in T^j$. For instance, given $T^1 = \{morning,\ evening\}$ and $T^2 = \{workday,\ weekend\}$, two allowed values of

5 We abuse of the notation by denoting as $t(e)$ the function returning the time of event e, and denoting as t a particular value of T.

t are $t^1 = \{morning, weekend\}$, and $t^2 = \{evening, workday\}$. Thus, in this case there is no "older" data, but rather data relevant (or not) for a particular context t. This change in modeling time information leads to a different interpretation of $w_t(\cdot)$—and the way it is used in the computation of $F(u, i, t)$—from that used in continuous heuristic-based TARS. In this case $w_t(\cdot)$ can be viewed as a penalty applied to non-relevant data (for the target context), and depending on the contextualization strategy, it plays a different role in the computation of $F(u, i, t)$.

In contextual pre-filtering $w_t(\cdot)$ is used as a filter to select relevant ratings for neighborhood and prediction computation. That is, a set M^t of ratings relevant to context t, $M^t = \{r_{u,i} | r_{u,i} \in M,\ w_t\left(r_{u,i}\right) = 1\}$, is selected, and neighborhood and prediction computation are performed using M^t and a model like (1). In contextual post-filtering $w_t(\cdot)$ is used to adapt rating prediction values previously computed with the original rating matrix M and a model as (1):

$$F(u, i, t) = F(u, i) \cdot w_t(u, i, t)$$

For instance, Baltrunas and Amatriain (2009) created contextual micro-profiles, each of them containing ratings in a particular context, as a pre-filtering strategy aimed to better detect the user's preferences for specific time contexts. They tested several contextual schemes, such as *timeOfTheDay* = {*morning*, *evening*}, *timeOfTheWeek* = {*workday*, *weekend*} and *timeOfTheYear* = {*hot_season*, *cold_season*}, obtaining improvements on accuracy metrics. It is important to note that if rating timestamps are available, multiple time context attributes can be exploited. For instance, Lee et al. (2010) derived the time variables *season* = {*fall, winter, spring, summer*}, *dayOfWeek* = {*sun, mon, tue, wed, thu, fri, sat*}, and *timeOfDay* = {*midnight, dawn, morning, AM, noon, PM, evening, night*}, and used all these attributes together for recommendation computation.

Given the flexibility of the categorical context representation, it is easy to incorporate other contextual dimensions beyond time, and use more complex representations of time context. For instance, Panniello et al. (2009b) presented an experimental comparison of contextual pre-filtering and post-filtering approaches using the contextual variable *timeOfTheYear* in a hierarchical structure with two levels: a specific level that considers the values {*summer_holiday*, *summer_no_holiday*, *winter_holiday*, *winter_no_holiday*}, and a generic level in which the time variable takes the values {*summer*, *winter*}. More examples combining different discrete contextual dimensions (including time) can be found in Adomavicius et al. (2005); Gorgoglione and Panniello (2009); Panniello et al. (2013).

3.1.3 Time adaptive heuristics

In this type of TARS, parameters and/or data are dynamically adjusted according to changes of some data characteristics through time. This is an important difference with respect to continuous and categorical time-aware heuristics, as the rating prediction is not targeted for a particular time context. Adaptive heuristics generally penalize older preferences that are presumed to be not/less valid at recommendation time, and usually utilize a continuous time representation. Thus, they could be considered as a particular case of time decay heuristics, but, as noted before, they do not target a specific recommendation time. An example of time adaptive heuristics can be found in Lee et al. (2008, 2009), where implicit purchase information is transformed into explicit ratings by assigning increasing weights to newer ratings. This is modeled as a function $f_t(r)$ that assigns a *weight* to each rating r according to its timestamp:

$$F(u,i) = \underset{v \in N(u)}{\text{aggr}} \; r_{v,i} \cdot f_t\left(r_{v,i}\right)$$

Note that, differently to the time-dependent weight $w_t(\cdot)$ discussed in the previous sections, the function $f_t(r)$ only depends on the ratings, not on the target recommendation time.

These heuristics can also use, for instance, the time an item is incorporated into the system's catalog for the weight computation. We remark again that, under this formulation, the rating prediction is a function of the users and items, but not of the target (recommendation) time.

A particular formulation of $f_t(r)$ is given in Ding et al. (2006), where item ratings are weighted according to their deviation from the target user's most recent ratings on similar items. The underlying assumption is that the user's most recent ratings on a neighborhood of similar items show her current trend on such items. Following the idea of detecting user interest drift, Min and Han (2005) and Cao et al. (2009) developed approaches that derive time series of user ratings, aiming to establish current user interests. An additional form of time adaptive heuristics is described in Lathia et al. (2009), where the number k of neighbors to be used in a kNN approach is dynamically adjusted, looking for values of k that diminish the error on previous predictions. Other approaches performing time adaptive heuristics are Zimdars et al. (2001), where Web logs are coded as time series, and Tang et al. (2003), where the production year of movies is used to reduce dimensionality in a CF system by means of an "old" movie pruning strategy.

3.2 Model-based approaches

These approaches include continuous time-aware models, categorical time-aware models and time adaptive models. They use the rating collection to build models with which rating predictions are computed.

3.2.1 Continuous time-aware models

Approaches in this category build models from rating data, taking the dynamics of such data into account. As in continuous time-aware heuristics, in continuous time-aware models time is represented as a continuous variable, and the target time is explicitly considered for rating prediction.

One of the best known examples of these approaches is the temporal dynamics model proposed by Koren (2009b), which corresponds to a MF model. In general, a MF model is an extension of singular value decomposition (SVD) in which M is iteratively approximated by user and item latent factor matrices P and Q of lower dimension[6] (d in our notation). Using such latent factors the predicted rating computation is formulated as:

$$F(u,i) = \sum_{j=0}^{d} P_{j,u} Q_{j,i} = p_u^T q_i \tag{3}$$

where p_u and q_i are the u-th column of P and i-th column of Q, and represent the latent factor vectors of user u and item i, respectively. Values of P and Q are computed by minimizing an estimation of a rating prediction error, such as the Frobenius norm, $\min ||M - PQ||_F^2$.

In order to take time effects in the MF model into consideration, Koren incorporated into (3) static and dynamic bias terms as follows:

$$F(u,i,t) = \mu + b_u(t) + b_i(t) + p_u^T(t) q_i$$

where μ denotes the overall mean rating, $b_u(t)$ and $b_i(t)$ are user- and item-specific time-dependent biases, and user factors are assumed to change through time, thus becoming time-aware, $p_u(t)$.

Note that this model assigns latent factor vector(s) to a user at each time step $t = s$. In the formulation of Koren (2009b) t is measured at the day level, being able to detect changes in the users' preferences with a daily granularity. An additional factorization-based model including time-variant factors is described in Rendle (2011).

Another example of a continuous time-based model is given in Xiong et al. (2010), where a Bayesian probabilistic tensor factorization (TF) model is proposed. In that work Xiong and colleagues incorporated time as an additional feature vector associated to each time step, instead of to each user and time step as in Koren (2009b). In this way the $|U| \times |I|$ rating matrix M is extended into a three dimensional *tensor* $\mathfrak{M} \in \mathbb{R}^{|U|\times|I|\times|W|}$—note that a tensor extends the two dimensions of the MF model to three or more dimensions. Under this scheme, users, items and time are modeled via probabilistic latent factor vectors that are computed by means of the TF approach. Hence, rating prediction is computed by the scalar product:

$$F(u,i,t) = \sum_{j=0}^{d} P_{j,u} Q_{j,i} W_{j,t}$$

where $P_{\cdot,u}$, $Q_{\cdot,i}$ and $W_{\cdot,t}$ denote the feature vectors of user u, item i, and time step t, respectively. This formulation avoids an expensive increase of time-related factors associated with entities, as in the case of MF models.

A different modeling scheme is presented in Koenigstein et al. (2011), where Koenigstein and colleagues use *session* factors to model specific user behavior in music listening sessions. Such sessions are inferred from time information associated to ratings, in such a way that a session is defined as a set of consecutive ratings with

[6] For the sake of simplicity we consider latent factors vectors of the same size, but they can be of different size.

no more than 5 h of difference. The authors found that users tend to rate songs in a session very similarly.

We note that the main disadvantage of this type of models, from the time perspective, is their inability to extrapolate future temporal dynamics. Authors, however, argue that these models isolate persistent signals from transient noise, thus helping in predicting future user behavior.

3.2.2 Categorical time-aware models

The main difference between continuous time-aware models and categorical time-aware models is given by the domain of the time information they use. In categorical time-aware models, time information is represented as discrete contextual values. These models correspond to algorithms implementing the generic *contextual model* framework proposed in Adomavicius and Tuzhilin (2011) by exploiting time context information.

One of the first approaches on categorical time-aware models is Oku et al. (2006), where several contextual dimensions including time, company and weather were incorporated into a support vector machine model for recommendation.

Karatzoglou et al. (2010) used TF to model n-dimensional contextual information. They called this approach *multiverse* recommendation because of its ability to bridge data pertaining to different contexts (universes of information) into a unified model. In this approach the rating information is represented as an n-dimensional tensor $\mathfrak{M} \in \mathbb{R}^{|U|\times|I|\times|C_1|\times|C_2|\times\cdots\times|C_n|}$, where $C_1, C_2, \ldots, C_n$ represent contextual dimensions of information. By applying the High Order SVD decomposition approach (Lathauwer et al. 2000), M is factorized into factor matrices $P \in \mathbb{R}^{d_P\times|U|}$, $Q \in \mathbb{R}^{d_Q\times|I|}$, $C_k \in \mathbb{R}^{d_{C_k}\times|C_k|}$, and a central tensor $\mathfrak{S} \in \mathbb{R}^{d_P\times d_Q\times d_{C_1}\times\cdots\times d_{C_n}}$, in which d_k denotes the number of latent factors describing each dimension k. In this way the rating prediction formula becomes a function of the target user, item, and context:

$$F(u, i, k_1, k_2, \ldots, k_n) = \mathfrak{S} \times_P P_{\cdot,u} \times_Q Q_{\cdot,i} \times_{C_1} C_{1\cdot,k_1} \times_{C_2} C_{2\cdot,k_2} \times \cdots \times_{C_n} C_{n\cdot,k_n}$$

where $\times_D$ denotes a tensor-matrix multiplication operator, and the subscript shows the direction on the tensor on which to multiply.

Another example of categorical time-aware model is given in Rendle et al. (2011), where factorization machines (FMs) (Rendle 2011) were used to combine continuous and categorical time information.

3.2.3 Time adaptive models

Differently to time-aware models, in time adaptive models the rating prediction does not depend on the target recommendation time. Hence, rating estimations are improved by means of exploiting temporal ordering of ratings rather than temporal closeness and relevance with respect to the target recommendation time.

An example of time adaptive model is described in Karatzoglou (2011), where a temporal order of ratings is incorporated into a MF model, by means of learning differentiated item factors according to the ratings' timestamps, thus extending (3) to:

$$F(u, i) = \sum_{j=1}^{d} P_{j,u} Q^{s}_{j,i} Q^{s-1}_{j,a} Q^{s-2}_{j,b} \cdots Q^{s-N}_{j,N}$$

where $Q^{s}_{\cdot,i}$ is the item factor vector learned for item i with user preference information recorded until time step s, and $a, b, \ldots, N$ denote the items consumed at time step $s - 1$, $s - 2$ and so on. This model is referred to as a multiplicative model. The authors also proposed a summative model in which factor products $p_u^T q_i^s$, $p_u^T q_a^{s-1}$, $p_u^T q_b^{s-2}, \ldots, p_u^T q_N^{s-N}$ are summed up to compute the rating prediction $F(u, i)$.

Another example of time adaptive model can be found in Jahrer et al. (2010), where several time-unaware models are learned. In order to blend such models, training data are split into several bins according to rating times (among others), and different weights are assigned to each time bin. In this way the blending process becomes time-dependent.

4 On the use of evaluation protocols

A deep revision of state of the art on TARS showed us a considerable diversity in the evaluation protocols used to assess time-aware recommendation performance. We shall start our discussion by comparing online and offline evaluation approaches. Then, we shall focus our analysis on protocols used in offline evaluation, due to its primacy in the TARS literature. Aiming to identify and analyze important differences in these protocols, and further define a general methodological framework for TARS evaluation, we shall describe the main steps followed to conduct an offline evaluation of RS in general and TARS in particular. Next we shall introduce various issues regarding evaluation metrics and methodologies that are potential sources of differences between evaluation protocols. Based on such issues, we shall state a number of questions and conditions that have to be considered in any TARS evaluation protocol. These conditions will be addressed by the methodological description framework proposed in Sect. 5.

4.1 Online and offline evaluation of TARS

Two broad types of evaluations can be performed to assess the performance of recommender systems: online evaluation (also called user-based evaluation), and offline evaluation (also called system evaluation) (Herlocker et al. 2004; Gunawardana and Shani 2009; Shani and Gunawardana 2011).

In all evaluation cases, a RS is built with information about the users—such as preferences and demographics—and the items—such as content descriptions and attributes. Then, user responses to received recommendations are tracked and used to compute some metrics related to one or more desired properties of the system, e.g. accuracy, diversity and novelty of rating predictions and user satisfaction.

In online evaluation, users use several settings of the system under evaluation, and may fill questionnaires regarding their experience with the system and the received recommendations. Evaluation results are then obtained by recording and comparing the users' behavior (ratings, activity logs, etc.) over the different system settings (Kohavi et al. 2009), by comparing subjective user perceptions gathered in the questionnaires (Knijnenburg et al. 2012), or by a combination of both. In offline evaluation, datasets containing past user behavior are used to simulate how users would have behaved if they had used the evaluated system. Evaluation results are obtained by comparing actual and predicted ratings for users and items of the dataset (Herlocker et al. 2004; Shani and Gunawardana 2011).

Online evaluation can be considered preferable to offline evaluation, mainly due to its ability to take into account the user's experience (Knijnenburg et al. 2012; Konstan and Riedl 2012). That is, the user's perceptions about the interaction with the system. Moreover, some studies have shown differences between offline metric results and user perceived quality (Cremonesi et al. 2011). Although there is no clear explanation, variations in user interfaces (Cosley et al. 2003), data selection (Cremonesi et al. 2011), and situational and personal characteristics of users (Knijnenburg et al. 2012) may be related with such differences.

Despite its advantages, online evaluation is more difficult and expensive to perform, as it requires counting with (fully) functional implementations of the alternative system settings to evaluate, including its user interface. Additionally, users have to be recruited and possibly paid for testing the system. Offline evaluation, on the other hand, only requires implementing the system's algorithmic settings to be tested. If a dataset is already available, no user recruiting is needed. Thus, a common practice is to test new recommendation algorithms by offline evaluations, especially as a preceding step to online evaluation, in which only the best (offline) performing algorithms would be tested. In this way overall experimentation costs are reduced (Kohavi et al. 2009; Shani and Gunawardana 2011).

In the case of TARS there is little work on online evaluation, probably because the main goal of TARS developments has been to increase the accuracy of recommendations, which can be easily measured in offline evaluations. In fact, some researchers have recruited users for creating datasets with contextual information, but have performed offline evaluation with the created datasets, as e.g. described in Adomavicius et al. (2005) and Park et al. (2007). Examples of TARS evaluated online are the approaches of Weng et al. (2009) and Oku et al. (2006).

Regarding the usage of data, we note that in online evaluation all the data available up to the moment of the recommendation request is used. In the case of offline evaluation, diverse strategies to select the data that will be used to answer to an arbitrary established recommendation request can be used. We deepen into these strategies in the following sections.

4.2 Generic stages of offline evaluation

Figure 1 shows a schematic view of the generic stages of an offline evaluation protocol of RS. In the figure the ratings of matrix M are partitioned into a training set Tr and a test set Te, using a *training-test splitting* process. Tr is used to build a recommen-

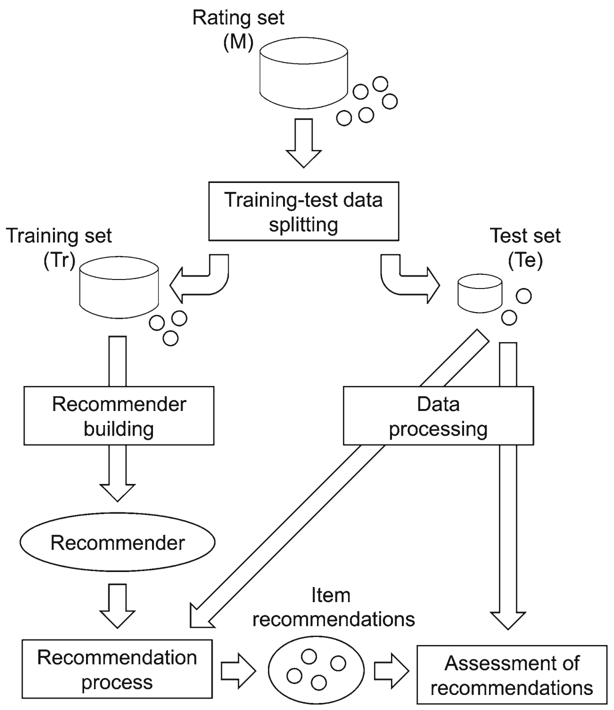

Fig. 1 Schematic view of the stages followed in an offline evaluation protocol for RS

dation model (or equivalently to serve as input for a recommendation heuristics). Te, also referred to as *ground truth*, is used to simulate the users' behavior in response to received recommendations of the model. Once the model (or heuristics) is built, the recommendation process is performed to generate a set of item suggestions for each user. Next, these item suggestions are compared against the ground truth Te using a number of metrics. Additional processing of data may be required depending on the *recommendation task* at hand, namely *rating prediction* and *top-N recommendation* (Herlocker et al. 2004; Gunawardana and Shani 2009). In the former task recommendations correspond to rating predictions, and the above metrics are computed by comparing predicted and actual values of test ratings. In the latter task recommendations consist of a ranked list of items predicted as the most appealing for the user. Here, metrics take into account the ranking positions of relevant and non-relevant test items in the generated list (Cremonesi et al. 2010; Bellogín et al. 2011). In this case the notion of item relevance can take multiple definitions, e.g. by considering as relevant items those whose actual ratings are over certain threshold.

Given a rating matrix M, a recommendation algorithm (model or heuristics) and a recommendation task, we then identify two main components of an offline evaluation protocol: the *metrics*, with which we measure desired properties of generated recommendations, and a *methodology*, with which we create a particular instantiation of the described evaluation stages. We explain these two components next.

4.3 Evaluation metrics

A wide array of metrics has been proposed and used to evaluate and compare recommendation algorithms, attempting to assess different properties of generated recommendations (Gunawardana and Shani 2009). In the literature most of the published evaluations of RS have focused on *rating prediction accuracy* metrics, such as the mean absolute error (MAE) and the root mean squared error (RMSE), which measure how well a RS can predict the ratings of particular items.

Recently it has been argued, however, that *ranking precision* metrics[7] are better suited for recommendation purposes, as RS are typically required to present a limited number of the most appealing items for a user—instead of rating predictions for individual items (Konstan and Riedl 2012). For such purpose, in general, an item ranking is constructed by comparing (and sorting) rating predictions, and the top-N items in the ranking are regarded as recommendations. Then, ranking precision metrics measure to what degree the list of recommendations contains relevant items for the users (Herlocker et al. 2004). These metrics usually correspond to adaptations of metrics used in the information retrieval field, such as precision, recall, and normalized discounted cumulative gain (Baeza-Yates and Ribeiro-Neto 1999), and metrics used in the machine learning field, such as the receiver operating characteristic (ROC) curve, and the area under the ROC curve (Ling et al. 2003).

Note that, in general, rating prediction accuracy metrics are used to assess a rating prediction task, while ranking precision metrics are used to assess a top-N recommendation task.

Apart from prediction accuracy and ranking precision, more recently, other recommendation properties are starting to be investigated, such as novelty and diversity (Vargas and Castells 2011), by means of metrics like Self Information (Zhou et al. 2010), and Intra List Similarity (Ziegler et al. 2005). Novelty metrics aim to capture the degree in which unknown items (for a user in particular or the community in general) are recommended, whereas diversity metrics assess how similar the items in a recommendation list are.

4.4 Evaluation methodologies

Some stages of a RS evaluation can be implemented in different ways, thus leading to methodological differences across studies. An important source of methodological differences is the training-test rating splitting process, especially when rating timestamps are available (Gunawardana and Shani 2009), as is the case in TARS. Hence, for instance, in the literature one can find diverse implementations of the *hold-out* method (Duda et al. 2000), which has been widely used for rating data splitting (Gunawardana and Shani 2009).

A first decision to make when designing an evaluation setting is whether the rating splits should be based on some *rating order* criterion. For instance, we may use a time-

[7] These metrics are also referred to as *usage prediction accuracy* metrics (Gunawardana and Shani 2009) and *classification accuracy* metrics (Herlocker et al. 2004).

dependent ordering in which all ratings are first ordered according to their timestamps. Next, those ratings prior to a particular date are selected for training, whereas the remaining ratings are selected for test, as done in Panniello et al. (2009a). We may, on the other hand, use a random selection of training and test ratings, without considering the ratings' timestamps or any ordering criterion, as done in Stormer (2007). Between these two extreme cases, there are intermediate options. For example, ordering the ratings by timestamp separately for each user, and assigning the most recent ratings of each user to the test set, as done in the Netflix Prize competition (Bennet and Lanning 2007).

Furthermore, a careful analysis of differences among the reviewed evaluation protocols shows that the ordering—and the overall splitting process—can also have different *base sets*. That is, the splits can be created on the base of the whole rating matrix, as done in Karatzoglou (2011), or can be created independently over each user's ratings, as done in Koenigstein et al. (2011).

The *number of ratings* selected for the test set is also chosen differently among TARS-related papers. An example is the typical proportion-based schema (e.g. 80 % of the ratings for the training set, and the remaining 20 % for the test set) used, e.g. in Lee et al. (2010). Other strategies, for instance, select a fixed number of ratings per user, as done in Ding et al. (2006), or set a threshold date and select ratings before (after) that date for the training (test) set, as done in Lathia et al. (2009), to name a few.

Additionally, there are different *cross-validation* strategies aimed to increase the generalization of the evaluation results on independent data. A popular strategy is to use some variant of resampling method, such as *X-fold cross validation*. This technique enables to average results over several test sets extracted from M, by means of repeatedly splitting the data (Adomavicius et al. 2005).

Another important source of methodological differences is related to specific requirements for assessing particular recommendation tasks. For instance, in order to evaluate the top-N recommendation task, we have to establish a set of *target items* a recommender has to rank. As described in Bellogín et al. (2011), several approaches have been used to generate the set of target items. For example, only ranking those items for which the user's relevance can be determined (Adomavicius et al. 2005); or mixing items considered relevant for the user (e.g. highly rated items) with other items considered as non-relevant (e.g. unrated items), as done in Cremonesi et al. (2010).

Also for the top-N recommendation task, the concept of item *relevance* has been interpreted differently. One interpretation is that all items in the target user's test set are relevant. It has been argued, however, that some items in the user's test set should be treated as non-relevant; consider for example a one-time-played song, or a movie rated with 1 in a 1–5 rating scale. Furthermore, as noted by Parra and Amatriain (2011), such information may be treated as a lack of information about the items' relevance.

4.5 Methodological questions and evaluation conditions

Although the diversity in evaluation protocols is not a problem per se, it makes the comparison of RS—and consequently the selection of the appropriate recommendation approach for a particular application or domain—difficult. In order to deal with this situation, we believe that a formal description of decisions on existing alternatives in the evaluation of RS is needed. Hence, from the potential sources of differences between evaluation protocols introduced in Sect. 4.4, we state the following methodological questions regarding the design of a RS evaluation protocol:

- MQ1: What *base set* is used to perform the training-test splitting?
- MQ2: What *rating order* is used to assign ratings to the training and test sets?
- MQ3: How many *ratings* comprise the training and test sets?
- MQ4: What *cross-validation method* is used for increasing the generalization of the evaluation results?
- MQ5: Which items are considered as *target items* (in a top-N recommendation task)?
- MQ6: Which items are considered *relevant* for each user (in a top-N recommendation task)?

In the next section we shall describe the possible ways to address each of these methodological questions by means of a number of *evaluation conditions* which we have extracted from our survey of evaluation protocols used in the TARS literature. These evaluation conditions are: *base set conditions*, addressing MQ1; *rating order conditions*, addressing MQ2; *size conditions*, addressing MQ3; *cross-validation conditions*, addressing MQ4; *target item conditions*, addressing MQ5; and *relevant item conditions*, addressing MQ6. We shall describe and formalize these conditions in the context of a generic methodological description framework, aiming to facilitate a precise communication of the evaluation conditions that drive an evaluation process.

5 A methodological description framework to formalize TARS evaluation conditions

In this section we describe and formalize a number of evaluation conditions, which are aimed to address the methodological questions stated in Sect. 4, and are established from a deep analysis of the evaluation protocols used in the TARS literature. We propose a methodological description framework to facilitate the comprehension of such decisions, and to make the evaluation process fair and reproducible under different circumstances. We first give some general definitions that will be used thereafter in the description of the identified conditions and the proposed framework.

5.1 General definitions

Definition 1 A ***split*** Σ of the rating matrix M is a partition of M into a training set Tr and a test set Te, that is, $\Sigma = \langle Tr, Te \rangle | \, Tr \cap Te = \emptyset$.

Definition 2 A ***splitting procedure*** is an algorithm that takes as input the 4-tuple $\langle M \times \mathscr{b} \times \sigma \times \mathscr{s} \rangle$, where $\mathscr{b} \in \mathfrak{B}$ is a condition in the set $\mathfrak{B}$ of conditions to define a base set, $\sigma \in \mathcal{O}$ is a condition in the set $\mathcal{O}$ of conditions to define an ordered set, and $\mathscr{s} \in \mathcal{S}$ is a condition in the set $\mathcal{S}$ of conditions to define the size of the training and test sets of a split. The output of a splitting procedure is a split Σ.

Definition 3 A ***base set condition*** $\mathscr{b} \in \mathcal{B}$ specifies the set of base datasets $M^{\mathscr{b}} = \{M_1, M_2, \ldots, M_m\}$ generated from M, being $M = \bigcup_k M_k$, $M_k \subseteq M$.

Definition 4 An ***ordering condition*** $\sigma \in \mathcal{O}$ establishes an ordered sequence for a set of ratings. The sequence Seq_k defined by the order condition σ over the ratings in M_k is:

$$Seq_k = \left\{r_{(1)}, r_{(2)}, \ldots, r_{(|M_k|)}\right\} \mid r_{(j)} \in M_k$$

where $r_{(j)}$ denotes the j-th rating in the sequence.

Definition 5 A ***size condition*** $\mathscr{s} \in \mathcal{S}$ sets a criterion for computing the number of ratings from Seq_k that will be assigned to the training and test sets Tr and Te, denoted by $\mathscr{s}^{Tr}\,(Seq_k)$ and $\mathscr{s}^{Te}\,(Seq_k)$.

Algorithm 1 describes the steps of a splitting procedure. First, the base datasets are built (step 1). Then, according to the order condition σ, a sequence of ratings is generated from each base dataset (step 2). After that, according to the size condition $\mathscr{s}$ (and a set of parameter values depending on the value of $\mathscr{s}$), the number of training and test ratings is established (step 3). Taking these sizes into account, the first ratings in each sequence are assigned to the training set, and the remaining ratings are assigned to the test set (step 4).

```
Splitting Procedure($M$, $\mathscr{b}$, $\sigma$, $\mathscr{s}$)
Input: Rating matrix $M$, base set condition $\mathscr{b}$, rating order condition $\sigma$,
       size condition $\mathscr{s}$
Output: Split $\Sigma = \langle Tr, Te \rangle$
Step 1: According to $\mathscr{b}$, build the base datasets $M^{\mathscr{b}} = \{M_1, M_2, \cdots, M_m\}$.
Step 2: According to $\sigma$, generate the sequences $Seq_k$ from the datasets $M_k$.
Step 3: According to $\mathscr{s}$, compute the sizes $\mathscr{s}^{Tr}(Seq_k)$ and $\mathscr{s}^{Te}(Seq_k)$.
Step 4: For each ordered sequence $Seq_k$:
        4.1: Find the rating $r_{(p_k)}$ such that the subsequence
             $\{r_{(1)}, r_{(2)}, \cdots, r_{(p_k)}\}, r_{(j)} \in Seq_k$ contains $\mathscr{s}^{Tr}(Seq_k)$ ratings.
        4.2: Assign the first $p_k$ ratings to the training set $Tr$, and the
             remaining ratings to the test set $Te$, forming the split
             $\Sigma = \langle Tr, Te \rangle$.
```

In the following paragraphs we provide specific formulations for conditions on $\mathfrak{B}$, $\mathcal{O}$ and $\mathcal{S}$ found in the revised TARS literature. We also describe additional evaluation conditions—*cross-validation method, target items*, and *relevant items*—that extend the methodological description framework proposed.

5.2 Base set conditions

The base set conditions state whether the rating order and size conditions in steps 2 and 3 of the splitting procedure are applied on the whole set of ratings in M, or on each of the sub-datasets of M independently. Specifically, we consider two conditions, namely the community-centered ($\mathscr{b}_{cc}$) and the user-centered ($\mathscr{b}_{uc}$) base set conditions.

Community-centered base set condition. A single base dataset with all the ratings in M is used:

$$M^{b_{cc}} = M$$

When this base set condition is applied, some users may have all or none of their ratings in the test set. This makes it impossible to assess the RS performance for such users, as in general, CF strategies cannot generate recommendations for users without profiles (i.e. training ratings), and metrics cannot be computed without ground truth data (test ratings). This problem is due to large differences on rating patterns between users, having some users with many more ratings than others, and also different rating distributions across time. A solution for this is to split each user's ratings separately.

User-centered base set condition. A base dataset M_u is built with the ratings of each user u:

$$M^{b_{uc}} = \{M_u \mid u \in U\}, M_u = \left\{r_{u,\cdot} \mid r_{u,\cdot} \in M\right\}$$

By performing the splitting independently on each user's ratings, we can ensure that all users will have ratings in both the training and test sets[8].

5.3 Rating order conditions

The rating order conditions establish the type of ordering to apply in the generation of the rating sequence(s) used to make the training-test set splitting. We have identified two rating orders in the TARS literature: a time-independent (i.e. random) ordering (σ_{ti}), and a time-dependent ordering (σ_{td}).

Time-independent rating order condition. The timestamps associated to ratings are not considered for ordering them. Some other ordering criteria may be used, but in general, the sequence[9] is generated by randomly selecting ratings from the base datasets M_k:

$$M_k \overset{\sigma_{ti}}{\rightarrow} Seq_k^{ti}$$

The main advantage of this rating order condition is its applicability, since it does not require timestamp information. Its main drawback, from a contextual point of view, is that time dependencies between training and test ratings do not hold. This means that the timestamp of some training ratings could be more recent than the timestamp of test ratings. That is, some ratings included in the training set may have been produced after some ratings in the test set. This situation can be interpreted as an evaluation of TARS

[8] In a strict sense, a user-centered split ensures that *most* users will have training and test data, but there may be some users without enough ratings for both training and test sets. This will depend not only on the number of ratings of each user, but also on the definition of other evaluation conditions like the *size* condition.

[9] Note that each user's rating sequence Seq_u^{ti} is generated independently in case of using a user-centered base set condition.

that have knowledge about “future” preferences of the users, which is far away from a real-world setting, and may give TARS unfair advantages in an offline evaluation (Campos et al. 2011b). Using a time-dependent order can help avoiding this problem.

Time-dependent rating order. The rating sequence is ordered according to the rating timestamps by means of a time-dependent rating ordering σ_{td}:

$$M_k \overset{\sigma_{td}}{\rightarrow} Seq_k^{td}$$

being Seq_k^{td} ordered by increasing rating timestamp, i.e., $t\left(r_{(j)}\right) \leq t\left(r_{(j+1)}\right);\ \forall r_{(j)} \in Seq_k^{td}$[10].

This rating order condition aims to maintain time dependencies between ratings, and is only applicable when ratings have associated timestamp information. Thus, when the desired evaluation setting is to mimic real-world conditions, a time-dependent ordering may be preferred.

It is important to note that a strict time-dependent ordering between training and test ratings can be generated only under a community-centered base set condition. In such a case all the test ratings have a timestamp more recent than the timestamp of any training rating, that is, $\forall r_{Tr} \in Tr, r_{Te} \in Te, t\left(r_{Tr}\right) < t\left(r_{Te}\right)$. This combination generates an evaluation setting similar to a real-world setting, where a deployed RS can only be trained with data available up to a particular moment, and its effectiveness is usually evaluated with user feedback provided afterwards. Examples of this approach are presented in Ardissono et al. (2004) and Panniello et al. (2009a). A drawback of this combination—$\mathscr{b}_{cc}$ and σ_{td}—is that, due to different user rating distributions through time, there may be many users without ratings either in the training or in the test set.

When a time-dependent rating order condition is used with a user-centered base set—$\mathscr{b}_{uc}$ and σ_{td}, each user’s ratings are time-sorted *independently from other users’ ratings*. This means that time-dependent order of ratings is maintained for each user independently, and thus cross-user time dependencies are not maintained; some users may have training ratings subsequent to test ratings of other users. On the other hand, by using this combination TARS do not have access to future knowledge of the target user (in contrast to using σ_{ti}), and the problem of leaving many users with only training or test ratings is avoided (in contrast to using $\mathscr{b}_{cc}$).

The user-centered and time-dependent ordering conditions combination has been one of the most used in TARS evaluation, probably because it was used for building the training and test sets of the Netflix Prize competition (Bennet and Lanning 2007). However, it has been argued that TARS which make recommendations to a target user by exploiting knowledge about other users’ “future” preferences (with respect to the target recommendation time) may have unfair advantages in an evaluation (Campos et al. 2011b).

[10] In case of ties, they could be broken by sorting the tied ratings by user_id. If still there are tied ratings, then they could be sorted by item_id (note that in real datasets it is possible to find users with several ratings with the same timestamp due to, e.g. inconsistencies in the log subsystem).

5.4 Size conditions

The size evaluation conditions establish how many ratings from each rating sequence Seq_k are included in the training and test sets, Tr and Te. In the literature it is common to establish and report the size of the test set. There are several ways to establish that size; some of them can be used in combination with any other evaluation conditions, while others can only be used with a particular combination of conditions, as we shall explain below. The covered conditions include proportion-based size, fixed size, and date-based size. Note that, as described in Algorithm 1, the rating ordering is considered when assigning ratings to training and test sets. In general, the first $\mathcal{s}^{Tr}(Seq_k)$ ratings of Seq_k are assigned to Tr, and the remaining $\mathcal{s}^{Te}(Seq_k)$ ratings are assigned to Te.

Proportion-based size. Denoted by $\mathcal{s}_{prop}$, this condition establishes that a proportion q_{prop} of the ratings in each Seq_k is used as test data, and the remaining ratings are used as training data, that is, $\mathcal{s}_{prop}^{Te}(Seq_k) = q_{prop} \cdot |Seq_k|$ and $\mathcal{s}_{prop}^{Tr}(Seq_k) = \left(1 - q_{prop}\right) \cdot |Seq_k|$, with $q_{prop} \in [0, 1]$[11].

When a user-centered base set condition ($\mathcal{b}_{uc}$) is used—i.e. a rating sequence is built for each user—a proportion-based size ensures that different users have a similar proportion of their ratings in the training and test sets. This combination is used in Zheng and Li (2011).

Fixed size. Denoted by $\mathcal{s}_{fix}$, this condition establishes that a fixed number q_{fix} of the ratings in each Seq_k are used as test data, that is, $\mathcal{s}_{fix}^{Te}(Seq_k) = q_{fix}$ and $\mathcal{s}_{fix}^{Tr}(Seq_k) = |Seq_k| - q_{fix}$.

When using $\mathcal{b}_{uc}$ a fixed size condition ensures that the same number of ratings is assigned to the users' test sets, regardless the number of training ratings of each user. Given that some users may have less than q_{fix} ratings, the size condition can be changed for such users. For instance, in the Netflix Prize competition dataset, a fixed number of $q_{fix} = 9$ ratings of each user was selected for building the test sets, but in cases where $|Seq_u| < 18$ only half of a user's ratings were selected as test data. That is, there was a mechanism switching from the fixed size into a proportion-based size condition with $q_{prop} = 0.5$ (Bennet and Lanning 2007).

If the fixed size condition refers to the number of training ratings, all the users have the same number of training ratings q_{fix}^{Tr}, leaving the remaining ratings for the test set (whose size would vary from user to user). This latter case has been referred to as *given N* (Ding and Li 2005), where $N = q_{fix}^{Tr}$.

Time-based size. Denoted by $\mathcal{s}_{time}$, this size condition can only be used with a time-dependent rating order condition. It establishes a threshold time q_{time} that is used to assign the ratings of each Seq_k into training and test sets. In this case the ratings with a timestamp after q_{time} are assigned to Te, and the ratings with a timestamp before q_{time} are assigned to Tr. Hence, given the last index p_k in Seq_k that satisfies $t\left(r_{(p_k)}\right) \le q_{time}$, the sizes $\mathcal{s}_{time}^{Tr}(Seq_k) = \left|\{r_{(1)}, r_{(2)}, \ldots, r_{(p_k)}\}\right|$ and $\mathcal{s}_{time}^{Te}(Seq_k) = |Seq_k| - \mathcal{s}_{time}^{Tr}(Seq_k)$ are established, as done in Lu et al. (2009).

[11] In case of a non-integer size value, it could be rounded to the nearest integer value.

Using the time-based size condition in combination with either a community-centered or a user-centered base set condition yields equivalent training and test sets if the threshold q_{time} is the same for all users. This particular case is similar to the combination of a community-centered base set, a time-dependent rating order, and a proportion-based size conditions (with an appropriate q_{prop} value).

The time-based size condition can be enhanced by incorporating an ending time limit q_{end_time}. In this case only the ratings whose timestamps are between q_{time} and q_{end_time} are assigned to the test set. Hence, given the last index l_k in Seq_k that satisfies $t\left(r_{(l_k)}\right) \leq q_{end\_time}$, the sizes $\mathcal{s}^{Tr}_{time}(Seq_k) = p_k$ and $\mathcal{s}^{Te}_{time}(Seq_k) = l_k - p_k$ are assigned, and the ratings with timestamp subsequent to q_{end_time} are discarded, as done in (Liu et al. 2010b; Pradel et al. 2011). In this case the assumption is that the user's preferences (manifested as ratings) that were produced long after a target recommendation time should not be considered for assessing the quality of recommendations.

An alternative way to specify a time-based size consists in establishing a period of time (or window size) q_{time_window} to the timespan of the test ratings (e.g. $q_{time\_window} = 10$ days). Hence, the ratings assigned to Te are those starting from the last known rating time in each Seq_k minus q_{time_window}. In this case, given the last index p_k in Seq_k that satisfies $t\left(r_{(p_k)}\right) \leq t\left(r_{(|Seq_k|)}\right) - q_{time\_window}$, the sizes $\mathcal{s}^{Tr}_{time\_window}(Seq_k) = p_k$ and $\mathcal{s}^{Te}_{time\_window}(Seq_k) = |Seq_k| - p_k$ are assigned, as done in Zhan et al. (2006).

We note that when used in combination with a $\mathcal{b}_{uc}$ condition, $t\left(r_{(|Seq_k|)}\right)$ is different for each user. In such a case, there will be a different starting date for the ratings in the test set of each user. Because of this, combining the $\mathcal{s}_{time\_window}$ and $\mathcal{b}_{uc}$ conditions has a disadvantage similar to that of the $\mathcal{s}_{fix}$ condition when it is used with user-centered base set and time-dependent rating order conditions, since some users may have all of their ratings within the test timespan.

5.5 Cross-validation conditions

Cross-validation conditions define whether one or more data splits are built with the ratings in M. Research in Statistics (Arlot and Celisse 2010) and Machine Learning (Dietterich 1998) has shown that the variability of evaluation results is diminished by repeating the evaluation process several times using a different data split each time. This procedure is commonly referred to as *cross-validation*. In this section we describe cross-validation methods that have been used in the revised TARS papers. We group them by the way they handle the time dimension. First of all, we introduce the hold-out procedure, as this is the basic building block for the above methods, and moreover, it can deal with the time dimension in different ways according to other evaluation conditions explained in previous sections.

Hold-out splitting. Only one training set and one test set are built, according to the evaluation conditions *base set*, *rating order*, and *size*, and avoiding pairwise (user, item) rating overlap, i.e. $Tr \cap Te = \emptyset$. TARS performance is measured by training the recommendation algorithm with ratings in Tr, and comparing generated recommendations with the ground truth Te, as done, e.g., in Panniello et al. (2009a). The rationale for having separated, non-overlapping training and test sets is that measuring performance of rating predictions on training data may produce an underestimated prediction error (Arlot and Celisse 2010).

5.5.1 Time-independent cross-validation methods

These cross-validation methods use a time-independent rating order condition, and build X different splits $\Sigma_x = (Tr_x, Te_x), x \in \{1, \ldots, X\}$ with the ratings in M, avoiding pairwise (user, item) rating overlap on each split, i.e. $Tr_x \cap Te_x = \emptyset$.

Repeated sampling. This method repeats the hold-out splitting procedure X times, according to some *base set* and *size* conditions. By using a random, time-independent rating order condition (i.e. a different sequence is generated in each repetition, due to the use of a random ordering) it is ensured that each split will be different from the rest. Gordea and Zanker (2007) applied this method.

User resampling. This method randomly samples a subset of users $U_x \subset U$ in each repetition. Then apply the hold-out splitting procedure on each dataset $M_x = \{r_{u,\cdot} | u \in U_x\}$, according to some *base set* and *size* condition. Zheng and Li (2011) used this method.

X-fold cross validation. This is a commonly used method that takes a time-independent ordered sequence of ratings, and splits it into X disjoint sets (called *folds*). Then, X different training and test sets are built, by assigning the ratings in one fold to the test set, and the ratings in the remaining $X - 1$ folds to the training set, as e.g. done in Adomavicius et al. (2005). In general, the folds are equally sized, and thus, the value of X is used to determine the size of training and test sets. Note that this is similar to use a proportion-based size condition, with $q_{prop} = 1/X$. Furthermore, this method can be applied with a community-centered or a user-centered base set conditions. We note that details about the used base sets are rarely reported in the literature; in general, only the usage of an X-fold cross validation method and the value of X are reported.

Leave-one-out. This is a particular X-fold cross validation method in which $X = |M|$. One rating in M is considered as the test set, and the remaining ratings are used for training in each repetition. Although this method has showed the lowest variability in results in generic prediction problems (Arlot and Celisse 2010), its high computational cost (the algorithms must be trained and evaluated $|M|$ times) makes it unfeasible in many situations. Cremonesi and Turrin (2009) applied this method.

Category-based cross validation. This is a particular cross-validation method that has been used to evaluate categorical TARS. In this case, different splits are built according to the value of one or more categorical context variables. The rationale for this is twofold. On the one hand, make independent evaluations of TARS performance in different categorical contexts. And on the other hand, facilitate the computation of a single value of performance across contexts, for a given metric. This cross validation condition can be applied with a time-independent or a time-dependent rating order condition, but requires the availability of categorical context information associated to the ratings. Note that, when the categorical variable correspond to a time context variable, this method let evaluate TARS performance separately on different time contexts (e.g. weekday vs. weekend). But this method does not ensure that time dependencies between ratings in a given training-test set pair hold (unless a time-dependent rating condition is used). The number of different splits that can be generated with this method is limited by the number of different categorical values of the contextual variables. Baltrunas and Amatriain (2009) used this type of cross-validation.

In general, time-independent cross-validation methods present two characteristics that must be handled with care when evaluating TARS. On the one hand, they may produce overlapping training and test sets from the time ordering point of view. On the other hand, they may produce pairwise (user, item) rating overlaps between different training or test sets, which may make the application of statistical tests difficult (Dietterich 1998).

5.5.2 Time-dependent cross-validation methods

These cross-validation methods aim to ensure that time dependencies between ratings in each training-test set pair hold, i.e. $\forall r_{Tr} \in Tr_x, \forall r_{Te} \in Te_x, t(r_{Tr}) < t(r_{Te})$. This is accomplished by using the combination of a community-centered base set and a time-dependent rating order conditions, and some time-based size condition. The time-based size condition can be iteratively updated to form time-evolving training and test sets.

Time-dependent resampling. This is a simple method that selects X different ratings $r^x \in M \mid x \in \{1, 2, \ldots, X\}$, and builds splits Σ_x by using a $\measuredangle_{time}$ condition with the time threshold $q^x_{time} = t(r^x)$. Hermann (2010) used this method.

Time-dependent users resampling. This method is similar to the time-independent user resampling, but uses a time-dependent rating order condition. In this case X different splits are built because the users (and thus the ratings) vary from sample to sample. Cremonesi and Turrin 2010 used this method.

Increasing-time window. This method builds different splits by means of increasing the timespan of training sets. It requires the definition of a training window size $q_{time\_window\_Tr}$ and a test window size $q_{time\_window\_Te}$, measured in some time unit, e.g. *days* or *weeks*. For building the initial training and test sets, this method uses a $\measuredangle_{time}$ condition, with $q_{time} = t(r_{(1)}) + q_{time\_window\_Tr}$ and $q_{end\_time} = q_{time} + q_{time\_window\_Te}$. The method builds subsequent training and test sets by iteratively updating q_{time} and q_{end_time} as follows: $q'_{time} = q_{time} + q_{time\_window\_Te}$ and $q'_{end\_time} = q_{end\_time} + q_{time\_window\_Te}$. Setting $q_{time\_window\_Tr}$ and $q_{time\_window\_Te}$ such that $q_{time\_window\_Tr} + X \cdot q_{time\_window\_Te} = t(r_{(|M|)})$, the method builds X training and test sets, being the timespans in the test sets equally sized. Lathia et al. (2009) used this method.

Fixed-time window. This cross-validation method is a variation of the increasing-time window method. The timespan size of each training set is maintained by means of discarding "old" ratings. In this case the first training and test sets are built as in the increasing-time window method, and the subsequent training sets are pruned by discarding those ratings out of the training time window $q_{time\_window\_Tr} : r \mid t(r) < q_{time} - q_{time\_window\_Tr}$.

Note that this method leads to faster training and evaluation processes compared with those of the increasing-time window method, since the training set sizes do not increase. However, it has the disadvantage of losing part of the training data, which may be valuable for some TARS. Pradel et al. (2011) applied this method.

5.6 Target item conditions

These conditions, which are specific for the top-N recommendation task evaluation, select the target items $Target_u$ to be ranked by the evaluated TARS. We recall that the need to take these conditions into consideration arises from the different nature of rating prediction and top-N recommendation tasks. In rating prediction, a RS is requested to predict the rating a target user would give to a target item. In top-N recommendation, there is no target item, but only a target user; the RS is then requested to estimate the set of top items the target user would prefer.

We note that a broad—and close to a real world setting—target item condition would be ranking all the items except the target user u's training items, which are already known by u, i.e. setting $Target_u = I \setminus I_{Tr_u}$. In the revised literature, nonetheless, smaller item sets have been used as $Target_u$, letting a faster evaluation, as fewer items have to be ranked by the assessed TARS.

The impact of target item conditions on evaluating recommendation performance has been studied by Bellogín et al. (2011). In the following we describe conditions identified in the revised TARS papers.

User-based target items. The items in the target user's test set Te_u are ranked, i.e. $Target_u = I_{Te_u}$.

The rationale for this condition is to avoid ranking items for which there is no explicit evidence of user preferences. This method was used by Adomavicius et al. (2005) and Ma et al. (2007).

Community-based target items. All the items in the test set Te (i.e. the ratings of the whole community of test users) are ranked, i.e. $Target_u = (\bigcup_{v \in U} I_{Te_v}) \setminus I_{Tr_u}$. A variation that includes more target items consists of ranking all the items in the community training set Tr, i.e. $Target_u = (\bigcup_{v \in U} I_{Tr_v}) \setminus I_{Tr_u}$.

The rationale of this condition is to include in $Target_u$ items interpreted as non-relevant for user u, in order to assess a recommendation algorithm's ability to better rank relevant items. The underlying assumption is that items rated by u are relevant, and unrated items are presumably non-relevant. Pradel et al. (2011) and Zimdars et al. 2001 used this condition.

One-plus random target items. In order to describe this condition, we first define the set of *highly relevant items* for user u as $I_{hrel}u = \{i \in I \mid r_{u,i} \in Te_u, r_{u,i} \geq \tau_{hrel}\}$, where τ_{hrel} is a *high-relevance threshold*, i.e. items in the ground truth of u with high ratings; and the set of *non-relevant items* for user u as $I_{\overline{rel}_u} = \{i \in I \mid r_{u,i} = \emptyset\}$, i.e. items that have not been rated by u.

In this case, several sets $Target_u^k$ for the target user u are built, each of them consisting of one highly relevant item $i^k \in I_{hrel_u}$, plus a set of non-relevant items $J_{\overline{rel}_u} \subseteq I_{\overline{rel}_u} : Target_u^k = i^k \cup J_{\overline{rel}_u}$.

The rationale for this condition is to find out whether a recommendation algorithm is able to consistently rank the selected relevant items above all other non-relevant items (Cremonesi et al. 2010). This condition was used by Stormer (2007).

Other target item conditions. A particular target item condition we identified in our review is based on a *given list of target items*, where a fixed set of items is ranked. This approach has been used in domains where there is a fixed set of possible items to recommend at a particular moment, as in TV show recommendation, in which there is a set of shows being broadcast at a particular time. In such a case $Target_u$ contains the TV listings at recommendation time, as shown by Vildjiounaite et al. (2008).

Another condition we identified is the *one-item target item*, in which $Target_u$ is composed of just one item at a time. In this case a TARS has to decide whether or not to recommend a given item. This condition can be used to measure the ability of an algorithm to recommend only relevant items by repeating the evaluation process with all known items. This particular condition was used with the leave-one-out cross-validation method by Panniello et al. (2009a).

5.7 Relevant item conditions

Relevant item conditions select the items to be interpreted as relevant for the target user. The notion of relevance is central for information retrieval metrics applied to evaluate top-N recommendations. A RS has a set of ratings for some items, and depending on such ratings, the items have to be interpreted as relevant or non-relevant. In this context we identify two main conditions, namely the test-based and the threshold-based relevant item conditions.

Test-based relevant items. The set of relevant items for user u, I_{rel_u}, is formed by the items in u's test set: $I_{rel_u} = I_{Te_u}$. By using this condition, rating/consuming an item is interpreted as indicative of interest for such item. Liu et al. (2010b) used this condition.

Threshold-based relevant items. The items in the user's test set rated/consumed above a threshold value τ_{rel} are considered as relevant, i.e. $I_{rel_u} = \{i \in I | r_{u,i} \in Te_u, r_{u,i} \geq \tau_{rel}\}$. Thus, the test set is pruned from low rated items. This condition was used by Adomavicius et al. (2005) and Vildjiounaite et al. (2008).

Note that the definition of I_{rel_u} is similar to the definition of I_{hrel_u} used in the description of the one-plus random target item condition. The difference between both sets is their threshold value, being $\tau_{hrel} > \tau_{rel}$ in general.

6 Empirical comparison of TARS evaluation conditions

In order to assess the impact of the identified conditions on time-aware recommendation evaluation, we conducted an empirical study comparing the recommendation performance of a number of algorithms when different combinations of the above conditions are used. In this comparison we include two categorical TARS—as they are instances of the most widely used approach for context-aware recommendation—and a well-known heuristic-based continuous TARS. We consider both the rating prediction and the top-N recommendation tasks, and analyze several rating prediction accuracy and ranking precision metrics, as well as novelty and diversity metrics. Moreover, we use three publicly available datasets that belong to different domains—movies and music[12], and have distinct types of ratings—explicit and implicit ratings. In the next sections we present the recommendation algorithms evaluated, the datasets used, and the evaluation metrics and methodologies which were compared. We present and discuss the results obtained, taking advantage of our framework to fairly state the conditions in which the experiments were conducted.

6.1 Recommendation algorithms

For our study, we evaluated two categorical heuristic-based TARS, namely contextual pre-filtering and contextual post-filtering algorithms. These algorithms have been widely used in the CARS-related research literature, and enable an easy incorporation of time context information. We also evaluated a Time Decay algorithm, as an example of continuous heuristic-based TARS.

As baseline algorithm we considered a context-unaware *weighted user-based kNN* algorithm (Herlocker et al. 1999):

$$F(u,i) = \bar{r}_u + \frac{\sum_{vN(u)} \left(r_{v,i} - \bar{r}_v\right) \cdot sim(u,v)}{\sum_{v \in N(u)} sim(u,v)}$$

where $sim(u,v)$ denotes a user similarity function based on the type of ratings used, including weights to penalize user similarities based on little information (understood as a low number of data points). For explicit ratings, the similarity function is the weighted Pearson correlation, defined as:

$$sim(u,v) = \frac{n}{w} \cdot \frac{\sum_{i \in I_v \cap I_u} \left(r_{v,i} - \bar{r}_v\right) \cdot \left(r_{u,i} - \bar{r}_u\right)}{\sum_{i \in I_v \cap I_u} \left(r_{v,i} - \bar{r}_v\right)^2 \cdot \sum_{i \in I_v \cap I_u} \left(r_{u,i} - \bar{r}_u\right)^2}$$

where n is the number of items rated by both users u and v, and w is a constant. For implicit ratings, the similarity function is the weighted cosine similarity, defined as:

$$sim(u,v) = \frac{n}{w} \cdot \frac{\sum_{i \in I_v \cap I_u} r_{v,i} \cdot r_{u,i}}{\sqrt{\sum_{i \in I_v \cap I_u} \left(r_{v,i}\right)^2} \cdot \sqrt{\sum_{i \in I_v \cap I_u} \left(r_{u,i}\right)^2}}$$

where $r_{u,i}$ denotes the number of times the user u consumed item i. We set $w = 50$ and $k = 200$ in all our experiments, as they provided good results and tendencies similar to other tested values. In cases where kNN was unable to compute a recommendation (e.g. because the target user/item did not appear in a training set), we used the user's/item's/global mean rating as the default prediction value. In case of implicit ratings, the default prediction value was set to 0, as there is not a meaningful mean rating value, but rather a long-tailed item consumption rate.

[12] In this work we use a pure collaborative filtering approach for the music recommendation domain. Content-based approaches—exploiting special characteristics of music, such as chord, melody, lyrics, and composer—could be used instead, but they fall out of the scope of this study.

The first evaluated recommendation algorithm is an implementation of the *contextual pre-filtering (PRF)* approach presented in Adomavicius and Tuzhilin (2011). This algorithm selects ratings relevant to the target context, and using the selected ratings, it computes rating predictions with a context-unaware recommendation strategy. Specifically, we used the *time Of The Week* $=$ {*workday,weekend*} categorical variable as time context, and the kNN approach described above as the underlying rating prediction strategy.

The second evaluated recommendation algorithm is an implementation of the *contextual post-filtering (POF)* approach presented in Adomavicius and Tuzhilin (2011). This algorithm first computes rating predictions, which can be generated by a context-unaware strategy, and then rating predictions are contextualized according to the target context. We used the same categorical time variable and kNN rating prediction strategy as the one used in the PRF approach. The contextualization of rating predictions was performed by a filtering strategy presented in Panniello et al. (2009b), which penalizes the recommendation of items that are not relevant in the target context as follows. The relevance of an item i for the target user u in a particular context c is approximated by the probability $P_c(u,i,c) = \frac{|U_{u,i,c}|}{k}$, where $U_{u,i,c} = \left\{v \in N(u) \,|r_{v,i,c} \neq \emptyset\right\}$, that is, the user's neighbors v who have rated/consumed item i in context c. The item relevance is determined by a threshold value τ_{P_c} (set to 0.1 in our experiments) that is used to contextualize the ratings as follows:

$$F(u,i,c) = \begin{cases} F(u,i) & if \quad P_c(u,i,c) \geq \tau_{P_c} \\ \min(R) & if \quad P_c(u,i,c) < \tau_{P_c} \end{cases}$$

where $\min(R)$ returns the minimum rating value for the domain at hand.

We note that this particular implementation of POF is better suited for a top-N recommendation task, as rating predictions may be heavily penalized in some cases (due to replacement of the predicted rating value by $\min(R)$ when $P_c(u,i,c) < \tau_{P_c}$), thus affecting rating prediction accuracy metrics.

The third evaluated recommendation algorithm is an implementation of the *Time Decay (TD)* approach, proposed in Ding and Li (2005):

$$F(u,i,t) = \bar{r}_u + \frac{\sum_{vN(u)} \left(r_{v,i} - \bar{r}_v\right) \cdot sim(u,v) \cdot e^{-\lambda \cdot (t - t(r_{v,i}))}}{\sum_{v \in N(u)} sim(u,v)}$$

with $\lambda = 1/200$ and t being a continuous time variable. Note that in this implementation we use time values with day granularity as done by Ding and Li (2005).

6.2 Datasets

In our study we used four datasets with timestamp information obtained from MovieLens[13] (Herlocker et al. 1999), Netflix[14] (Bennet and Lanning 2007), and Last.fm[15]

[13] MovieLens movie recommendations, http://movielens.umn.edu

[14] Neflix on-demand video streaming, http://www.netflix.com

[15] Last.fm Internet radio, http://www.lastfm.es

Table 1 Statistics of the used datasets

	MovieLens	MovieLensR	Netflix	Last.fm
Number of users	6,040	60	480,189	992
Number of items	3,706	2,056	17,770	174,091
Number of events	1,000,209	8,979	100,480,507	19,150,868 (898,073 user-item pairs)
Timespan	∼3 years	∼2 years	∼6 years	∼4.5 years
	(2000/04/26–2003/02/28)	(2000/12/27–2003/01/07)	(1999/11/11–2005/12/31)	(2005/02/14–2009/06/19)
Sparsity	0.0447	0.0834	0.0118	0.0052

(Celma 2010). The MovieLens and Netflix datasets have explicit ratings for movies, and the Last.fm dataset contains implicit ratings (listening to records) for music artists. Some basic statistics about the datasets are shown in Table 1. The MovieLensR dataset was built similarly as done by Ding and Li (2005), that is, by selecting the ratings of the first 60 users in the dataset (according to their identifier). We used this dataset to replicate the results reported in that work, where the Time Decay algorithm obtained significant improvements over the *k*NN algorithm.

To make the Netflix dataset more manageable, we divided it into 5 different sub-datasets on which we performed the evaluation. Specifically, we binned the original set of users into 50 equally sized bins, maintaining an increasing size of the user profiles in subsequent bins. Similarly to Lathia et al. (2009), we built each sub-dataset with the ratings of 1,000 randomly sampled users from each bin, plus those ratings generated during the first 500 days in the original dataset (from all users). Each of the new datasets had around 60,000 users, 17,765 items, and 11.7 million ratings, ranging the same timespan as the original dataset.

6.3 Evaluation metrics

Aiming to adequately cover the spectrum of mostly used recommendation quality metrics in offline evaluations, and to obtain an overview of distinct properties of recommendations generated by the tested algorithms under the selected evaluation methodologies, in our experiments, we considered both the rating prediction and the top-N recommendation tasks. We assessed accuracy for the two recommendation tasks, and novelty and diversity for the top-N recommendation task.

Specifically, we used Root Mean Squared Error (RMSE) (Shani and Gunawardana 2011) to assess accuracy in rating prediction, and Precision (P), Recall (R), and normalized Discounted Cumulative Gain (nDCG) (Baeza-Yates and Ribeiro-Neto 1999; Shani and Gunawardana 2011) to assess accuracy (ranking precision) of top-N recommendations. We computed novelty and diversity by means of Self-Information (I) (Zhou et al. 2010) and Intra-list Similarity (ILS) (Zhang and Hurley 2009), respectively. We computed the P, R, I and ILS metrics at cut-off 10, and nDCG on the whole lists of recommended items.

6.4 Evaluation methodologies

Aiming to find and analyze differences on recommendation quality results obtained with distinct combinations of evaluation conditions, we selected four different evaluation methodologies—three of them used a time-dependent order condition, and the other one used a time-independent order condition.

The first methodology (denoted uc_ti_prop) consists of a combination of a user-centered base set ($\mathcal{b}_{uc}$), a time-independent rating order (σ_{ti}), and a proportion-based size ($\mathcal{s}_{prop}$ with $q_{prop} = 0.2$) condition, which is used to generate a splitting $\Sigma_{uc\_ti\_prop}$. According to the framework introduced in Sect. 5, this splitting is represented as:

$$\Sigma_{uc\_ti\_prop} = \langle M, \mathcal{b}_{uc}, \sigma_{ti}, \mathcal{s}_{prop}\left(q_{prop} = 0.2\right)\rangle$$

The second methodology (denoted uc_td_prop) consists of the uc_ti_prop evaluation conditions, but using a time-dependent rating order. Its generated splitting $\Sigma_{uc\_td\_prop}$ is:

$$\Sigma_{uc\_td\_prop} = \langle M, \mathcal{b}_{uc}, \sigma_{td}, \mathcal{s}_{prop}\left(q_{prop} = 0.2\right)\rangle$$

The third methodology (denoted cc_td_prop) is equivalent to the uc_td_prop methodology with a community-centered base set condition. Its generated splitting $\Sigma_{cc\_td\_prop}$ is:

$$\Sigma_{cc\_td\_prop} = \langle M, \mathcal{b}_{cc}, \sigma_{td}, \mathcal{s}_{prop}\left(q_{prop} = 0.2\right)\rangle$$

Finally, the fourth methodology (denoted uc_td_fix) consists of a combination of a user-centered base set, a time-dependent rating order, and a fixed size ($q_{fix} = 9$) condition. Its generated splitting $\Sigma_{uc\_td\_fix}$ is:

$$\Sigma_{uc\_td\_fix} = \langle M, \mathcal{b}_{uc}, \sigma_{td}, \mathcal{s}_{fix}\left(q_{fix} = 9\right)\rangle$$

In case a user has less than 10 ratings, the size condition is switched to the proportion based size condition with $q_{prop} = 0.5$ in order to maintain such user in the training and test sets.

All the combinations use a hold-out procedure, a community-based target item condition $Target_u = \left(\bigcup_{v \in U} I_{Te_v}\right) \backslash I_{Tr_u}$, and a test-based relevant item condition $I_{rel_u} = I_{Te_u}$.

6.5 Results and discussion

Tables 2, 3, 4 and 5 respectively show the average recommendation performance results obtained on the MovieLens, MovieLensR, Netflix and Last.fm datasets. In these tables results are grouped by evaluation methodology, thus facilitating the identification of absolute and relative differences of algorithms within and between evaluation conditions.

Table 2 Performance results on the MovieLens dataset, grouped by evaluation methodology

Methodology	Algorithm	Metric					
		RMSE	P@10	R@10	nDCG	I@10	ILS@10
uc_ti_prop	KNN	0.9066	0.0206	0.0090	0.3424	7.8234	0.0536
	TD	0.9396*	0.0123*	0.0049*	0.3294*	7.5165*	0.0632*
	PRF	0.9136*	0.0167*	0.0077*	0.3468*	8.4496*	0.0980*
	POF	1.4350*	0.1020*	0.0382*	0.4163*	2.7039*	0.1800*
uc_td_prop	KNN	0.9246	0.0067	0.0031	0.3227	9.6791	0.0340
	TD	0.9448*	0.0070	0.0030	0.3196*	9.2671*	0.0390*
	PRF	0.9389*	0.0071	0.0033	0.3249*	8.8760*	0.0607*
	POF	1.6062*	0.0585*	0.0215*	0.3815*	2.7090*	0.1754*
cc_td_prop	KNN	0.9631	0.0322	0.0054	0.4642	8.4924	0.0346
	TD	0.9637*	0.0317	0.0054	0.4640*	8.4813	0.0350
	PRF	0.9709*	0.0196*	0.0035*	0.4570*	8.9337*	0.0633*
	POF	1.9169*	0.1988*	0.0252*	0.5255*	2.5657*	0.1714*
uc_td_fix	KNN	0.9531	0.0081	0.0091	0.2413	5.7619	0.1073
	TD	0.9804*	0.0043*	0.0047*	0.2338*	5.9759*	0.1052*
	PRF	0.9628*	0.0068*	0.0075*	0.2447*	6.7210*	0.1617*
	POF	1.4048*	0.0209*	0.0232*	0.2782*	2.6309*	0.1963*

For each methodology, *darker grey cells* indicate worse values of the corresponding metric, whereas the *white cells* indicate the best values. Statistical significant differences (Wilcoxon $p < 0.05$) of TARS algorithms are indicated with respect to kNN (*)

On the Last.fm dataset, we only tested the top-N recommendation task, since this dataset does not have explicit ratings with which rating prediction comparisons could be done. Thus, RMSE cannot be computed for such dataset. Also, in that dataset, the TD algorithm was not assessed because of multiple events (listening to records) and timestamps related to the same (user, item) pair, which do not let set a unique timestamp to apply the time decay weight.

In the tables we observe that the *performance results provided by each of the assessed metrics for a particular algorithm are very dissimilar when different evaluation methodologies are used*. Figures 2 and 3 show the differences of RMSE and nDCG metrics across methodologies for the four evaluated algorithms, on the MovieLens and MovieLensR datasets respectively. We observe that using the cc_td_prop and uc_td_fix methodologies, kNN, TD and PRF values of RMSE are larger than when using uc_ti_prop and uc_td_prop, on both datasets. Conversely, the uc_td_fix methodology leads to the lowest RMSE values of POF. Moreover, the uc_td_fix methodology leads to the lowest values of nDCG for all the tested algorithms (statistical significant differences with respect to the values obtained with the other methodologies).

The results obtained show that recommendation assessment under different evaluation protocols (metrics and methodologies) is an issue that has to be carefully taken into consideration in order to derive well-founded conclusions about relative performance of recommendation algorithms.

The conducted experiments also reveal that *dissimilar relative rankings of the tested algorithms are obtained, depending on the analyzed dataset, metric, and methodology*. For instance, regarding the rating prediction accuracy measured with the RMSE

Table 3 Performance results on MovieLensR dataset, grouped by evaluation methodology

Methodology	Algorithm	Metric					
		RMSE	P@10	R@10	nDCG	I@10	ILS@10
uc_ti_prop	KNN	1.1320	0.0367	0.0125	0.3784	4.6233	0.0022
	TD	1.1474	0.0367	0.0115	0.3793	4.5967	0.0017
	PRF	1.1756	0.0450	0.0208*	0.3836	4.3867*	0.0292*
	POF	2.2777*	0.0833*	0.0293*	0.4176*	2.5559*	0.2896*
uc_td_prop	KNN	1.1767	0.0317	0.0107	0.3651	4.4579	0.0042
	TD	1.1759	0.0317	0.0107	0.3659*	4.4330	0.0042
	PRF	1.2247	0.0400	0.0157	0.3706	4.3593	0.0934*
	POF	2.1047*	0.0500	0.0238*	0.3869*	2.8733*	0.2511*
cc_td_prop	KNN	1.2370	0.0958	0.0071	0.4535	4.8243	0.0022
	TD	1.2369	0.0958	0.0071	0.4534	4.8094	0.0023
	PRF	1.2387	0.1000	0.0273	0.4668	4.4043*	0.0675*
	POF	2.0521*	0.1292	0.0178*	0.4929*	2.8091*	0.2579*
uc_td_fix	KNN	1.2596	0.0233	0.0259	0.3222	4.0093	0.0048
	TD	1.2463	0.0283	0.0315	0.3270*	4.0130	0.0052
	PRF	1.3133*	0.0450*	0.0500*	0.3434*	3.8496*	0.1600*
	POF	1.9723*	0.0317	0.0352	0.3282*	2.5812*	0.2375*

For each methodology, *darker grey cells* indicate worse values of the corresponding metric, whereas the *white cells* indicate the best values. Statistical significant differences (Wilcoxon $p < 0.05$) of TARS algorithms are indicated with respect to kNN (*)

Table 4 Average performance results on the 5 sub-datasets generated from the Netflix dataset, grouped by evaluation methodology

Methodology	Algorithm	Metric					
		RMSE	P@10	R@10	nDCG	I@10	ILS@10
uc_ti_prop	kNN	0.9208	0.0060	0.0012	0.2694	9.8411	0.0683
	TD	0.9445*	0.0046*	0.0008*	0.2648*	10.1921*	0.0551*
	PRF	0.9210*	0.0055*	0.0021*	0.2682*	10.9453*	0.1756*
	POF	1.7514*	0.0668*	0.0133*	0.3497*	3.8757*	0.2452*
uc_td_prop	kNN	0.9379	0.0023	0.0005	0.2655	13.1286	0.0259
	TD	0.9587*	0.0018	0.0003*	0.2619*	12.9199*	0.0200*
	PRF	0.9454*	0.0017*	0.0004*	0.2624*	12.6859*	0.0729*
	POF	2.0637*	0.0632*	0.0144*	0.3351*	3.8053*	0.2448*
cc_td_prop	kNN	1.0276	0.0032	0.0007	0.2955	13.8497	0.0123
	TD	1.0356*	0.0033*	0.0006*	0.2944*	13.6752*	0.0116*
	PRF	1.0333*	0.0030*	0.0006*	0.2923*	13.1698*	0.0419*
	POF	2.2643*	0.0830*	0.0134*	0.3536*	3.4888*	0.2348*
uc_td_fix	kNN	0.9722	0.0009	0.0010	0.1899	10.2081	0.0702
	TD	0.9838*	0.0007*	0.0009*	0.1867*	8.7635*	0.0909*
	PRF	0.9837*	0.0012*	0.0013*	0.1892*	10.9743*	0.1478*
	POF	1.7705*	0.0116*	0.0131*	0.2346*	3.7158*	0.2464*

For each methodology, *darker grey cells* indicate worse values of the corresponding metric, whereas the white cells indicate the best values. Statistical significant differences (Wilcoxon $p < 0.05$) of TARS algorithms are indicated with respect to kNN (*)

Table 5 Performance results on the Last.fm dataset, grouped by evaluation methodology

Methodology	Algorithm	Metric				
		P@10	**R@10**	**nDCG**	**I@10**	**ILS@10**
uc_ti_prop	kNN	0.0013	0.0001	0.4084	5.4079	0.5886
	PRF	0.0044*	0.0004	0.4143	4.4504	0.5856
	POF	0.1254*	0.0070	0.4354	2.2878	0.3700
uc_td_prop	kNN	0.0005	0.0000	0.3856	5.4116	0.5662
	PRF	0.0022*	0.0001*	0.3910*	4.4670*	0.5402*
	POF	0.0874*	0.0051*	0.4132*	2.3687*	0.2796*
cc_td_prop	kNN	0.0031	0.0002	0.3422	5.1873	0.5540
	PRF	0.0057*	0.0006*	0.3497*	4.3868*	0.5084*
	POF	0.0546*	0.0059*	0.3571*	2.2138*	0.2555*
uc_td_fix	kNN	0.0002	0.0003	0.1839	4.8600	0.5267
	PRF	0.0002	0.0003	0.1846	4.0791*	0.4560*
	POF	0.0032*	0.0061*	0.2108*	2.3819*	0.2474*

For each methodology, *darker grey cells* indicate worse values of the corresponding metric, whereas the *white cells* indicate the best values. Statistical significant differences (Wilcoxon $p < 0.05$) of TARS algorithms are indicated with respect to kNN (*)

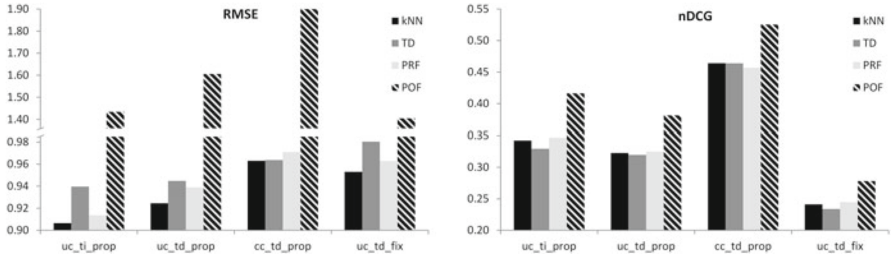

Fig. 2 RMSE and nDCG values of different algorithms across evaluation methodologies, on the MovieLens dataset

metric on the MovieLens dataset, when the cc_td_prop methodology was used TD outperformed PRF, differently to what was obtained when using the other methodologies[16]. A more notorious example can be observed in the MovieLensR dataset, where according to RMSE, and using the uc_ti_prop methodology, the best performance is achieved by kNN. For the same metric using any of the other methodologies, the best performance is achieved by TD, although differences were not statistically significant. In the case of the Netflix dataset, TD was not able to outperform kNN in any of the used methodologies. On the other hand, PRF and POF showed worse performance than kNN in terms of RMSE, regardless of the methodology used.

The relative performance rankings of the algorithms according to the ranking precision metrics also show differences across datasets, metrics, and methodologies. One example is observed when comparing algorithms' ranking using P@10 as performance metric. Using a user-centered base set (uc_td_prop), the algorithms are ranked as POF, PRF, TD and kNN on MovieLens, observing little difference between PRF, TD and

[16] These differences are statistically significant (Wilcoxon $p < 0.05$).

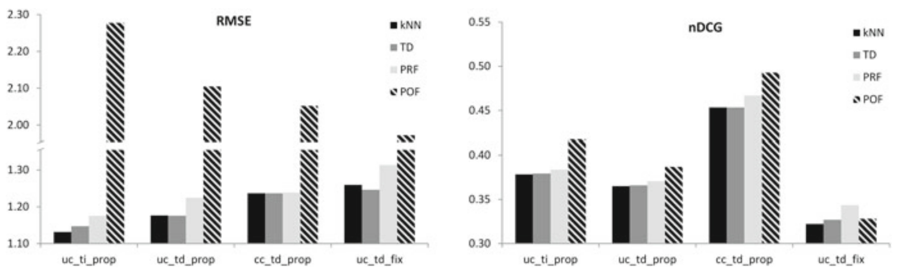

Fig. 3 RMSE and nDCG values of different algorithms across evaluation methodologies, on the MovieLensR dataset

kNN results. Changing into a community-centered base set (cc_td_prop), and using the same dataset, the ranking is POF, kNN, TD and PRF, and the difference between kNN and TD becomes statistically significant. Similar algorithm ranking switches are observed when changing the rating order condition (e.g. by comparing R@10 results on Netflix, using the ud_td_prop methodology instead of uc_ti_prop) and the size condition (e.g. comparing P@10 results on Netflix, using the uc_td_fix methodology instead of uc_td_prop). Moreover, it is notable the contrast in performance between rating prediction accuracy and ranking precision metrics. POF consistently showed a superior performance in terms of P@10, R@10 and nDCG across datasets and methodologies, and an inferior performance according to RMSE. We also remark that the magnitude of metric values may vary considerably from one methodology to another on the same dataset.

Regarding novelty and diversity, we observe even more variations on the relative rankings depending on the datasets, methodology and metric, compared with the ones observed in rating prediction accuracy and ranking precision metrics. It is interesting to note, anyhow, that the relative rankings on Last.fm dataset are stable across methodologies. Additionally, we also observe that in general there is a trade-off between precision-ranking accuracy, and diversity and novelty of TARS. This trade-off was also observed by Panniello et al. (2013) when exploiting other contextual dimensions.

Supported by the fact that they were obtained on several datasets, in different recommendation domains and tasks, and with various types of ratings (explicit and implicit), *the above results show the importance of clearly stating the conditions under which TARS are evaluated*. Differences of absolute metric values obtained by the same algorithm across methodologies confirm the difficulty of comparing results reported by other authors when evaluation conditions are not described precisely. And more importantly, differences on relative rankings of the algorithms across datasets, metrics, and methodologies show the need of selecting a proper evaluation protocol for identifying the improvement capabilities of new TARS correctly.

The evaluation conducted let us detect differences on recommendation results due to the usage of different evaluation protocols, even when the experiments were not exhaustive. We did not test all the described methodologies, and, moreover, alternative cross-validation methods, target items, and relevant item conditions may be used. Additional research is required in order to precisely determine the degree of impact on recommendation results by each evaluation condition.

Another issue that was not addressed in this work is whether the characteristics of individual user profiles or full datasets are related to changes on a metric's values for different evaluation methodologies. Regarding this issue, according to our results, it seems that a larger volume of data in a dataset leads to more stability of the results across methodologies. On the other hand, differences observed between results on the Last.fm and on the other datasets seem to indicate that the type of ratings may play an important role in evaluation. All in all, further experimentation is required in order to reliably establish which evaluation protocols are better suited for users (or datasets) with particular characteristics. The comparison of algorithms performance (and the manner of exploiting temporal context) deserves a deep study on its own, and thus is not considered here.

Finally, regarding the importance of maintaining time dependencies of ratings between training and test sets, we do not observe a consistent pattern. For the categorical time-aware algorithms (PRF and POF), applying a time-dependent ordering on ratings does not seem to have an important impact for evaluation. We believe this may happen when time is treated as a discrete variable, which may not enable tracking user preference evolution through time, but rather may let detect periodicity in the users' preferences. In contrast, for the continuous time-aware algorithm (TD), time-dependent ordering seems to have a more important role (especially on the MovieLensR dataset). From these results, we observe that *methodologies using a time-dependent rating order seem more sensitive to performance differences due to distinct time models used by TARS*. As part of future research, we plan to contrast the results reported herein with others obtained from additional categorical and continuous TARS.

7 Learnt lessons and open questions on TARS evaluation

In this section we summarize the key findings of our study, in order to provide a set of general methodological guidelines for TARS evaluation, and to identify additional research required for a deeper understanding of the evaluation conditions presented in this work. We begin by providing an exhaustive classification of state-of-the-art TARS, based on the evaluation conditions described in our methodological description framework. From this classification we identify the most common evaluation conditions used in the TARS literature.

By analyzing these conditions and the results obtained in our experiments, we formulate a set of methodological guidelines that may help researchers and practitioners interested in TARS evaluation to select proper combinations of evaluation conditions. We must note that these guidelines are based on the insights derived from the works analyzed in our survey, and the experiments we conducted. They thus do not cover all possible combinations of dataset characteristics, or evaluation conditions, metrics, and methodologies.

7.1 Evaluation conditions used in state-of-the-art TARS

Table 6 provides a summary of the revised papers on time-aware recommendation approaches that make use of offline evaluations, by showing the conditions (as defined in Sect. 5) used for evaluating each approach. In the table each row represents a particular combination of conditions, and each column is associated to an evaluation condition; some papers include more than one evaluation and/or condition combination.

In the table we first observe that despite the fact that the papers revised deal with time-aware recommendation approaches, in 24.6 % (14 out of 57) of them, a time-independent ordering of ratings is used in the evaluation protocols. On the other hand, a combination of a community-centered base set and a time-dependent ordering—which provides the evaluation scenario most similar to a real-world setting, maintaining temporal dependencies between training and test ratings, $\forall r_{Te}, r_{Tr}, t\,(r_{Te}) > t\,(r_{Tr})$– is used in 38.6 % of the papers (22).

Regarding cross-validation methods, the basic hold-out procedure (i.e. no cross-validation) is used in 70.2 % of the papers (40). Only in 10 of the 19 papers in which cross-validation was used (17.5 % of the whole list of papers) a time-based cross-validation method was used.

With respect to specific evaluation conditions of the top-N recommendation task, we first note that in 25 papers these conditions are not applicable, since the above task was not evaluated. We observe that in the majority of TARS-related papers addressing the top-N recommendation task (21 out of 32), a test-based relevant item condition was used, but we find an even distribution on the use of different target item conditions.

7.2 Analysis of key findings: General guidelines for TARS evaluation

From the summary given in Table 6 we observe that a considerable number of authors have used a time-independent rating ordering condition ($\mathscr{o}_{ti}$). In the empirical comparison, however, we found that an evaluation methodology with that condition was unable to detect the performance improvements obtained by continuous TARS on some of the used datasets. Thus, our first guideline is:
Guideline 1: *Use a time-dependent rating order condition* ($\mathscr{o}_{td}$) *for TARS evaluation.*

The use of this condition avoids ignoring variations on performance induced by the exploitation of time information by an evaluation methodology.

A second observation from Table 6 is that there is a similar amount of papers using community-centered ($\mathscr{b}_{cc}$) and user-centered ($\mathscr{b}_{uc}$) base set conditions. As discussed before, the combination $\mathscr{b}_{cc},\ \mathscr{o}_{td}$ provides a real world-like evaluation scenario. However, a problem of this combination is that many users may not be evaluated due to a lack of ratings in their training or test sets. Thus, considering the application of Guideline 1, we state a second guideline:
Guideline 2: *If the dataset has an even distribution of data among users, use the community-centered base condition. Otherwise, use a user-centered base condition in order to avoid biases towards profiles with long-term ratings.*

Note that in this guideline we refer to an imprecise notion of "even distribution of data among users". We do not have a specific characterization (e.g. in terms of profile sizes, timespans and sparsity levels) as to precisely define it. Additional research on this issue is required in order to provide a more precise guideline.

Table 6 List of revised papers about TARS, and their used offline evaluation conditions

Paper	General evaluation conditions										Top-N recommendation task evaluation conditions					
	Base set		Rating order		Size			Cross-validation			Target items				Relevant items	
	Community-centered	User-centered	Time-independent	Time-dependent	Proportion-based	Fixed	Time-based	None (Hold-out)	Time-independent CV	Time-dependent CV	User-based	Community-based	One-plus random	Other	Test-based	Threshold-based
Adomavicius et al. (2005)	X		X		X				X		X					X
Ardissono et al. (2004)	X			X			X	X			?	?	?	?	X	
Baltrunas and Amatriain (2009)	X		X		X				X		–	–	–	–	–	–
Bell and Koren (2007)		X		X		X		X			–	–	–	–	–	–
Bell et al. (2008)		X		X		X		X			–	–	–	–	–	–
Brenner et al. (2010)	X			X			X	X				X				X
Campos et al. (2010)	X			X			X	X				X				X
Campos et al. (2011b)	X			X			X			X		X				X
Cao et al. (2009)		X		X	X			X				X			X	
Chen et al. (2011a)		X		X		X		X			–	–	–	–	–	–
Chen et al. (2011b)		X		X		X		X			–	–	–	–	–	–
Cremonesi and Turrin (2009)		X	X			X			X				X		X	
Cremonesi and Turrin (2010)	X			X			X			X			X		X	
Ding and Li (2005)		X		X		X		X			–	–	–	–	–	–
		X	?	?		X		X			–	–	–	–	–	–
Ding et al. (2006)		X		X		X		X			–	–	–	–	–	–
Gantner et al. (2010)	X			X			X	X				X				X
Gordea and Zanker (2007)		X	X			X			X				X		X	
		X		X		X				X			X		X	

Table 6 continued

Paper	General evaluation conditions										Top-N recommendation task evaluation conditions					
	Base set		Rating order		Size			Cross-validation			Target items				Relevant items	
	Community-centered	User-centered	Time-independent	Time-dependent	Proportion-based	Fixed	Time-based	None (Hold-out)	Time-independent CV	Time-dependent CV	User-based	Community-based	One-plus random	Other	Test-based	Threshold-based
Gorgoglione and Panniello (2009)	X			X			X	X						X		X
Hermann (2010)	X			X			X			X	?	?	?	?	X	
Iofciu and Demartini (2009)	X			X			X	X						X	X	
Jahrer and Töscher (2011)		X		X		X		X			–	–	–	–	–	–
Jahrer et al. (2010)		X		X		X		X			–	–	–	–	–	–
Karatzoglou (2011)	X			X	X			X			–	–	–	–	–	–
Karatzoglou et al. (2010)	X		X		X				X		–	–	–	–	–	–
Koenigstein et al. (2011)		X		X		X		X			–	–	–	–	–	–
Koren (2009a)		X		X		X		X			–	–	–	–	–	–
Koren (2009b)		X		X		X		X			–	–	–	–	–	–
Lathia et al. (2009)	X			X			X			X	–	–	–	–	–	–
Lathia et al. (2010)	X			X			X			X		X			X	
Lee et al. (2010)	X		X		X			X					X		X	
Lee et al. (2008)	?	?	?	?	X			X			?	?	?	?	X	
Lee et al. (2009)	?	?	?	?	X			X			?	?	?	?	X	
Li et al. (2011)	X		X		X				X		–	–	–	–	–	–
Lipczak et al. (2009)	X			X			X	X						X	X	
Liu et al. (2010a)	X			X			X	X				X				X
Liu et al. (2010b)	X			X			X			X	?	?	?	?	X	

Table 6 continued

Paper	General evaluation conditions										Top-N recommendation task evaluation conditions					
	Base set		Rating order		Size			Cross-validation			Target items				Relevant items	
	Community-centered	User-centered	Time-independent	Time-dependent	Proportion-based	Fixed	Time-based	None (Hold-out)	Time-independent CV	Time-dependent CV	User-based	Community-based	One-plus random	Other	Test-based	Threshold-based
Lu et al. (2009)	X			X			X	X			–	–	–	–	–	–
Ma et al. (2007)		X	X		X			X			X				?	?
Min and Han (2005)	?	?	?	?	?	?	?	X			–	–	–	–	–	–
Montañés et al. (2009)		X		X			X	X						X	X	
Panniello et al. (2009a)	X		X			X			X					X	X	
	X			X			X	X						X	X	
Panniello et al. (2009b)	X			X			X	X						X		X
Pradel et al. (2011)	X			X			X			X				X	X	
	X			X			X			X		X			X	
Rendle (2011)	X			X			X			X	–	–	–	–	–	–
Rendle et al. (2011)	X		X		X				X		–	–	–	–	–	–
Stormer (2007)	X		X			X			X				X		X	
Tang et al. (2003)		X	X		X			X			X					X
		X		X		X		X			X					X
Töscher and Jahrer (2008)		X		X		X		X			–	–	–	–	–	–
Töscher et al. (2008)		X		X		X		X			–	–	–	–	–	–
Vildjiounaite et al. (2008)		X		X			X	X						X		X

Table 6 continued

Paper	General evaluation conditions										Top-N recommendation task evaluation conditions					
	Base set		Rating order		Size			Cross-validation			Target items				Relevant items	
	Community-centered	User-centered	Time-independent	Time-dependent	Proportion-based	Fixed	Time-based	None (Hold-out)	Time-independent CV	Time-dependent CV	User-based	Community-based	One-plus random	Other	Test-based	Threshold-based
Wu et al. (2011)		X		X		X		X			–	–	–	–	–	–
Xiang and Yang (2009)		X		X		X		X			–	–	–	–	–	–
	?	?	?	?	X			X			–	–	–	–	–	–
Xiang et al. (2010)		X		X		X		X					X		X	
Xiong et al. (2010)	X			X			X	X			–	–	–	–	–	–
		X		X		X		X			–	–	–	–	–	–
		X	X			X			X		-	–	–	–	–	–
Zhan et al. (2006)		X		X			X	X			?	?	?	?	X	
Zheng and Li (2011)		X		X	X					X		X			X	
Zimdars et al. (2001)	X			X			X	X				X			X	
	X		X		X			X				X			X	
Number of papers using the condition (total number of papers: 57)	29	26	14	45	15	22	24	40	10	10	3	10	6	8	21	10

"X" denotes an evaluation condition (at the corresponding column) that is used in a paper (at the corresponding row). Some papers include more than one evaluation and/or condition combination (one at each row). "–" denotes an evaluation condition that is not applicable for a paper. "?" indicates that we could not identify whether an evaluation condition was used or not in a paper. *CV* cross-validation

With respect to the size condition, we noted that when using a $\mathscr{b}_{cc}$, σ_{td} combination, there is an equivalence between proportion-based and time-based size conditions, since it is possible to find a proportion value that defines a splitting point equal to that from a time threshold. On the other hand, when using a $\mathscr{b}_{uc}$, σ_{td} combination, a time-based size definition may suffer from the same general problems of a $\mathscr{b}_{cc}$, σ_{td} combination (leaving some users without training or test data). Likewise, the use of a fixed size with a $\mathscr{b}_{uc}$ condition implies that users with small profiles will have a greater proportion of their profiles held out as test data, which may lead to a cold start situation for such users. In this way, our third guideline is:

Guideline 3: *Use a proportion-based size condition.*

This guideline ensures the appropriate control of the proportion of user profiles held for test purposes in case of using a $\mathscr{b}_{uc}$ condition, and provides an adequate control of training/test proportions when using either a $\mathscr{b}_{cc}$ or a $\mathscr{b}_{uc}$ condition. Our experiments did not cover the effect of using an ending time to limit the size of the test data; this effect may have particular importance on domains with seasonal changes, and further research on this topic is required to assess its impact on measured performance values of TARS.

These first 3 guidelines have been derived from the analysis of our literature review and empirical results, and encompass the methodological questions MQ1, MQ2 and MQ3 stated in section 4.5. Regarding MQ4, MQ5, and MQ6, although we did not perform experiments for assessing their impact in metric results, insights from the analysis of the surveyed work let us formulate two additional guidelines.

Guideline 4: *Apply a cross-validation method consistent with the conditions derived from guidelines 1, 2 and 3.*

Despite we did not test empirically the effect of different cross-validation methods on evaluation metrics, it is known that the use of more than one data split can diminish the variability of results (Dietterich 1998; Arlot and Celisse 2010). In this way, it is highly advisable to use a cross-validation method. Moreover, the selection of the method has to be consistent with the application of guidelines 1-3, i.e. the cross-validation method to use must apply the same rating order, base set, and size condition advised from the guidelines.

In the case of the target item and relevant item conditions, it is important to note that they are required only when assessing a top-N recommendation task. These conditions define which set of items have to be ranked to select the top-ranked items for recommendation, and which items in the test set have to be considered as relevant for the user, respectively.

Guideline 5: *For a top-N recommendation evaluation, use a community-based target item and a threshold-based relevant item condition.*

In the case of target item conditions, the closest condition to a real world setting would be to rank all available items unknown by the user. A small variation, which may let perform faster offline evaluations, consists of considering all items selected for the test set, that is, a community-based target item condition applied over the test set. Bellogín et al. (2011) found that it makes no differences in algorithm relative ranking to apply this condition on the training or the test sets, although they did not test time-aware algorithms. Despite there are several works that apply a user-based target item condition, algorithms' relative ranking may be different from that obtained with a community-based condition (Bellogín et al. 2011). Thus, following the idea of mimicking a real world setting, it is advisable to use a community-based target item condition.

With respect to relevant item conditions, as Parra and Amatriain (2011) noted, low ratings and consumption rates could be treated as a lack of information about the items' relevance. Hence, interpreting low rated/consumed items as relevant results may be counterintuitive. In this context, using a threshold-based condition leads to a more fair evaluation of performance.

By following these guidelines we believe that TARS performance can be assessed more objectively and realistically. Moreover, the guidelines enable an easier and fairer comparison of evaluation results between approaches from different authors, which would ease the development of better TARS.

In a more general perspective, we note that the results of our experimental comparison shown important divergences on the performance of algorithms across measured recommendation properties. Divergences are particularly remarkable between rating prediction accuracy and ranking precision metrics. In fact, the best performing algorithms on RMSE showed poor results on ranking precision metrics, and vice versa. This shows that relying only on rating prediction accuracy metrics for assessing the performance of recommender systems is not advisable, especially when the most valuable recommendation task is distinct from rating prediction. This is an important consideration, given that most work on TARS has been commonly evaluated in terms of rating prediction accuracy, without taking into account other metrics, and different recommendation properties and tasks. More importantly, we stress the value of providing clear and detailed specifications of the evaluation protocols used. Having such specifications at their disposal will enable other researchers and practitioners in the field to compare results fairly, and test whether a new algorithm is able to outperform existing TARS. The proposed methodological description framework, which lets provide rigorous descriptions of evaluation conditions used, may help in this task.

7.3 Open questions

Despite the important findings from this work—finally reflected on the proposed evaluation guidelines, a number of issues require further research in order to fully understand and take advantage of the different evaluation conditions identified so far.

First, more experimentation is required to properly analyze the impact of combinations of conditions not covered in our study. In particular, we did not consider cross-validation conditions, given the combinatorial explosion of conditions that should be tested. We leave the empirical study of those conditions and specific conditions regarding the top-N recommendation task as an interesting and important line of future research. For this purpose, we believe that the proposed evaluation framework provides an important conceptual structure to guide that research.

Another important pending issue is related to the analysis of the relation between characteristics of datasets (and individual user profiles), and evaluation conditions. For instance, the notion of "even distribution of data", stated in guideline 2, is imprecise, and requires further experiments in order to obtain a more precise definition.

Beyond that, the appropriateness of certain evaluation conditions for specific rating distributions through time/users/items, types of events (item ratings, purchases, and consumption), domains, etc. has to be investigated.

The relation between accuracy and novelty/diversity metrics also remains as an open evaluation issue. Given the increasing importance of the latter metrics in the RS field, additional analysis and explanations are required in order to provide time-aware recommendations with adequate levels of those properties.

A final question is whether improvements of TARS performance measured by offline evaluation results are effectively perceivable for real users. As noted, e.g. by Knijnenburg et al. (2012), accuracy improvements are not necessarily observable by users. The lack of online evaluation studies on TARS is a major limitation to address the above question.

8 Conclusions

In this paper we presented a comprehensive survey of state-of-the-art TARS. We found that results and conclusions reported in the literature about how to incorporate and exploit time information within a recommendation process seem to be contradictory in some cases. We hypothesized that the above discrepancies could be caused by meaningful divergences in the evaluation protocols used—metrics and methodologies. Thus, aiming to clarify these discrepancies, we identified a number of key conditions regarding offline evaluation of TARS, which let us state important methodological questions regarding evaluation setting:

- What base set is used to perform the training-test splitting?
- What rating order is used to assign ratings to the training and test sets?
- How many ratings comprise the training and test sets?
- What cross-validation method is used for increasing the generalization of the evaluation results?

In addition to these questions, we also covered specific evaluation conditions for the top-N recommendation task, addressing the following two methodological questions:

- Which items are considered as target items (in a top-N recommendation task)?
- Which items are considered relevant for each user (in a top-N recommendation task)?

Based on the identified conditions we provided a comprehensive classification of the evaluation protocols used in the state of the art on TARS. Furthermore, we developed a methodological description framework aimed to facilitate the comprehension of such conditions, and make the evaluation process fair and reproducible under different circumstances.

Additionally, we conducted an empirical comparison of the impact of several evaluation protocols on measuring relative performances of three widely used TARS approaches and one well-known non-contextual recommendation approach.

From our analysis and experiments, we reported important methodological issues that a robust evaluation of TARS should take into consideration in order to perform a fair evaluation of approaches, and facilitate comparisons among published experiments. In particular, the results obtained showed that the use of different evaluation conditions not only yields remarkable differences between metrics measuring distinct recommendation properties—namely accuracy, precision, novelty, and diversity. They also may affect the relative ranking of approaches for a particular metric. From our survey and analysis of the evaluation protocols used in the TARS literature, and from the results obtained in our experiments, we concluded a set of general guidelines aimed to facilitate the selection of conditions for a proper TARS evaluation. These guidelines recommend making training-test splitting based on a time-dependent rating order over the whole set of ratings in a dataset, applying a proportion-based size definition for training and test sets, using a compatible cross-validation method. In the case of top-N recommendation evaluation, using a real-world like set of items to rank, and a more confident interpretation of item relevance is advised.

We believe that our work raises interesting research questions regarding TARS evaluation. We consider of key importance to study the specific impact of each identified evaluation condition on the assessment of recommendation performance. Moreover, we propose as future research to deepen the analysis of existing relations between dataset characteristics and evaluation conditions, and general effects on less studied novelty and diversity metrics. By having a robust understanding of these effects, it would be possible to select the most appropriate and fair protocol for a given recommendation evaluation task.

Finally, we highlight the need of clearly stating the conditions in which offline experiments are conducted to evaluate RS in general, and TARS in particular. By having fair and consensual evaluation conditions, we will enable the reproducibility of experiments, and ease the comparison of recommendation approaches. In the hope that we can contribute to such purpose, we developed the methodological description framework presented in this paper.

Acknowledgements This work was supported by the Spanish Government (TIN2011-28538-C02), and by Comunidad Autónoma de Madrid and Universidad Autónoma de Madrid (CCG10-UAM/TIC-5877). The authors thank Centro de Computación Científica at UAM for its technical support. The authors also thank the anonymous reviewers for their valuable comments and suggestions.

References

Adomavicius, G., Sankaranarayanan, R., Sen, S., Tuzhilin, A.: Incorporating contextual information in recommender systems using a multidimensional approach. ACM Trans. Inf. Syst. **23**(1), 103–145 (2005)

Adomavicius, G., Tuzhilin, A.: Multidimensional recommender systems: a data warehousing approach. Second International Workshop on Electronic Commerce, pp. 180–192, Heidelberg (2001).

Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. **17**(6), 734–749 (2005)

Adomavicius, G., Tuzhilin, A.: Context-aware recommender systems. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.), Recommender Systems Handbook. ISBN 978-0-387-85820-3 (2011)

Ardissono, L., Gena, C., Torasso, P., Bellifemine, F., Difino, A., Negro, B.: User modeling and recommendation techniques for personalized electronic program guides. Personalized Digital Television. Springer. ISBN 978-1-4020-2164-0 (2004).

Arlot, S., Celisse, A.: A survey of cross-validation procedures for model selection. Stat. Surv. **4**, 40–79 (2010)

Baeza-Yates, R.A., Ribeiro-Neto, B.: Modern Information Retrieval. ACM Press, New York. ISBN 020139829X (1999).

Baltrunas, L.: Context-aware collaborative filtering recommender systems. PhD Dissertation, Free University of Bozen-Bolzano (2011)

Baltrunas, L., Amatriain, X.: Towards time-dependant recommendation based on implicit feedback. RecSys 2009 Workshop on Context-Aware Recommender Systems. New York, NY (2009).

Baltrunas, L., Ricci, F.: Experimental evaluation of context-dependent collaborative filtering using item splitting. User Model. User-Adapt. Interact., Special issue on Context-Aware Recommender Systems (2013)

Bell, R.M., Koren, Y.: Scalable collaborative filtering with fointly derived neighborhood interpolation weights. Seventh IEEE International Conference on Data Mining, pp. 43–52, Omaha (2007)

Bell, R.M., Koren, Y., Volinsky, C.: The Bellkor 2008 solution to the Netflix Prize. Netflix prize report (2008)

Bellogín, A., Castells, P., Cantador, I.: Precision-based evaluation of recommender systems: an algorithmic comparison. Fifth ACM Conference on Recommender Systems, pp. 333–336, Chicago (2011)

Bennet, J., Lanning, S.: The Netflix Prize. KDD Cup and Workshop, San Jose (2007)

Brenner, A., Pradel, B., Usunier, N., Gallinari, P.: Predicting most rated items in weekly recommendation with temporal regression. Workshop on Context-Aware Movie Recommendation, pp. 24–27, Barcelona (2010)

Burke, R.: Hybrid recommender systems: survey and experiments. User Model. User-Adapt. Interact. **12**(4), 331–370 (2002)

Burke, R.: Hybrid web recommender systems. In: Brusilovsky, P., Kobsa, A., Nejdl, W. (eds.) The Adaptive Web, pp. 377–408. Springer, Berlin (2007)

Campos, P.G., Bellogín, A., Díez, F., Chavarriaga, J.E.: Simple time-biased KNN-based recommendations. Workshop on Context-Aware Movie Recommendation, pp. 20–23, Barcelona (2010)

Campos, P.G., Díez, F., Bellogín, A.: Temporal rating habits: a valuable tool for rater differentiation. Second Workshop on Context-Aware Movie Recommendation, pp. 29–35, Chicago (2011a)

Campos, P.G., Díez, F., Sánchez-Montañés, M.: Towards a more realistic evaluation: testing the ability to predict future tastes of matrix factorization-based recommenders. Fifth ACM Conference on Recommender Systems, pp. 309–312, Chicago (2011b)

Cao, H., Chen, E., Yang, J., Xiong, H.: Enhancing recommender systems under volatile user interest drifts. Eighteenth ACM conference on Information and knowledge management, pp. 1257–1266, Hong Kong (2009)

Celma, O.: Music recommendation and discovery in the long tail. PhD Dissertation, Universitat Pompeu Fabra, Spain (2010)

Chen, P., Tsai, C., Chen, Y., Chou, K., Li, C., Tsai, C., Wu, K., Chou, Y., Li, C., Lin, W., Yu, S., Chiu, R., Lin, C., Wang, C., Wang, P., Su, W., Wu, C., Kuo, T., McKenzie, T.G., Chang, Y., Ferng, C., Ni, C., Lin, H., Lin, C., Lin, S.: A linear ensemble of individual and blended models for music rating prediction. KDD-Cup Report, San Diego (2011a)

Chen, T., Zheng, Z., Lu, Q., Jiang, X., Chen, Y., Zhang, W., Chen, K., Yu, Y.: Informative ensemble of multi-resolution dynamic factorization models'. KDD-Cup Report, San Diego (2011b)

Cosley, D., Lam, S.K., Albert, I., Konstan, J.A., Riedl, J.: Is seeing believing? How recommender interfaces affect users' opinions'. SIGCHI Conference on Human Factors in Computing Systems, pp. 585–592, Fort Lauderdale (2003)

Cremonesi, P., Garzotto, F., Negro, S., Papadopoulos, A.V., Turrin, R.: Looking for "Good" recommendations: a comparative evaluation of recommender systems'. Thirteenth IFIP TC 13 International conference on Human-computer interaction, pp. 152–168, Lisbon (2011)

Cremonesi, P., Koren, Y., Turrin, R.: Performance of recommender algorithms on top-n recommendation tasks. Fourth ACM conference on Recommender systems, pp. 39–46, Barcelona (2010)

Cremonesi, P., Turrin, R.: Analysis of cold-start recommendations in IPTV systems. Third ACM Conference on Recommender Systems, pp. 233–236, New York (2009)

Cremonesi, P., Turrin, R.: Time-evolution of IPTV recommender systems. Eighth International Interactive Conference on Interactive TV & Video, pp. 105–114, Tampere (2010)

Dey, A.K.: Understanding and using context. Pers. Ubiquitous Comput. **5**(1), 4–7 (2001)

Dietterich, T.G.: Approximate statistical tests for comparing supervised classification learning algorithms. Neural Comput. **10**(7), 1895–1923 (1998)

Ding, Y., Li, X.: Time weight collaborative filtering. Fourteenth ACM international Conference on Information and Knowledge Management, pp. 485–492, Bremen (2005)

Ding, Y., Li, X., Orlowska, M.E.: Recency-based collaborative filtering. Seventeenth Australasian Database Conference, pp. 99–107, Hobart, Tasmania (2006)

Dourish, P.: What we talk about when we talk about context. Pers. Ubiquitous Comput. **8**(1), 19–30 (2004)

Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. Wiley. ISBN 0471056693 (2000)

Gantner, Z., Rendle, S., Schmidt-Thieme, L.: Factorization models for context-/time-aware movie recommendations. Workshop on Context-Aware Movie Recommendation, pp. 14–19, Barcelona (2010)

Gordea, S., Zanker, M.: Time filtering for better recommendations with small and sparse rating matrices. Eighth International Conference on Web Information Systems Engineering, pp. 171–183, Nancy (2007)

Gorgoglione, M., Panniello, U.: Including context in a transactional recommender system using a pre-filtering approach: two real e-commerce applications. 2009 International Conference on Advanced Information Networking and Applications Workshops, pp. 667–672, Bradford (2009)

Gorgoglione, M., Paniello, U., Tuzhilin, A.: The effect of context-aware recommendations on customer purchasing behavior and trust'. Fifth ACM Conference on Recommender Systems, pp 85–82, Chicago (2011)

Gunawardana, A., Shani, G.: A survey of accuracy evaluation metrics of recommendation tasks. J. Mach. Learn. Res. **10**, 2935–2962 (2009)

Herlocker, J.L., Konstan, J.A., Borchers, A., Riedl, J.: An algorithmic framework for performing collaborative Filtering. Twenty-Second Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 230–237, Berkeley (1999)

Herlocker, J.L., Konstan, J.A., Terveen, L.G., Riedl, J.T.: Evaluating collaborative filtering recommender systems. ACM Trans. Inf. Syst. **22**(1), 5–53 (2004)

Hermann, C.: Time-based recommendations for lecture materials. 2010 World Conference on Educational Multimedia, Hypermedia and Telecommunications, pp. 1028–1033, Toronto (2010)

Hussein, T., Linder, T., Werner, G., Ziegler, J.: Hybreed: A software framework for developing context-aware hybrid recommender systems. User Model. User-Adapt. Interact, Special Issue on Context-aware Recommender Systems (2014)

Iofciu, T., Demartini, G.: Time based tag recommendation using direct and extended users sets'. ECML PKDD discovery Challenge 2009, pp. 99–107, Paris (2009)

Jahrer, M., Töscher, A.: Collaborative filtering ensemble. KDD-Cup Report, San Diego (2011)

Jahrer, M., Töscher, A., Legenstein, R.: Combining predictions for accurate recommender systems. Sixteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 693–702, Washington (2010)

Karatzoglou, A.: Collaborative temporal order modeling. Fifth ACM Conference on Recommender Systems, pp. 313–316, Chicago (2011)

Karatzoglou, A., Amatriain, X., Baltrunas, L., Oliver, N.: Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering. Fourth ACM Conference on Recommender Systems, pp. 79–86, Barcelona (2010)

Knijnenburg, B.P., Willemsen, M.C., Gantner, Z., Soncu, H., Newell, C.: Explaining the user experience of recommender systems. User Model. User-Adapt. Interact. **22**, 441–504 (2012)

Koenigstein, N., Dror, G., Koren, Y.: Yahoo! Music recommendations: modeling music ratings with temporal dynamics and item taxonomy'. Fifth ACM Conference on Recommender Systems, pp. 165–172, Chicago (2011)

Kohavi, R., Longbotham, R., Sommerfield, D., Henne, R.: Controlled experiments on the web: survey and practical guide. Data Mining Knowl. Discov. **18**(1), 140–181 (2009)

Konstan, J.A., Riedl, J.: Recommender systems: from algorithms to user experience. User Model. User-Adapt. Interact. **22**, 101–123 (2012)

Koren, Y.: The BellKor solution to the Netflix grand prize'. Netflix Prize, Report (2009a)

Koren, Y.: Collaborative filtering with temporal dynamics. Fifteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 447–456, Paris (2009b)

Lathauwer, L.D., Moor, B.D., Vandewalle, J.: A multilinear singular value decomposition. SIAM J. Matrix Anal. Appl **21**(4), 1253–1278 (2000)

Lathia, N., Hailes, S., Capra, L.: Temporal collaborative filtering with adaptive neighbourhoods. Thirty-Second International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 796–797, Boston (2009)

Lathia, N., Hailes, S., Capra, L., Amatriain, X.: Temporal diversity in recommender systems. Thirty-Third International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 210–217, Geneva (2010)

Lee, D., Park, S., Kahng, M., Lee, S., Lee, S.: Exploiting contextual information from event logs for personalized recommendation'. In Lee, R. (ed.), Computer and Information Science. ISBN 978-3-642-15404-1 (2010)

Lee, T.Q., Park, Y., Park, Y.: A time-based approach to effective recommender systems using implicit feedback. Expert Syst. Appl. **34**(4), 3055–3062 (2008)

Lee, T.Q., Park, Y., Park, Y.: An empirical study on effectiveness of temporal information as implicit ratings. Expert Syst. Appl. **36**(2), 1315–1321 (2009)

Li, R., Li, B., Jin, C., Xue, X., Zhu, X.: Tracking user-preference varying speed in collaborative filtering. Twenty-Fifth AAAI Conference on Artificial Intelligence, San Francisco (2011)

Ling, C., Huang, J., Zhang, H.: AUC: A better measure than accuracy in comparing learning algorithms. In: Xiang, Y., Chaib-draa B. (eds.). Advances in Artificial Intelligence **2671**, pp. 329–341, LNCS (2003)

Lipczak, M., Hu, Y., Kollet, Y., Milios, E.: Tag sources for recommendation in collaborative tagging systems. ECML PKDD Discovery Challenge 2009, pp. 157–172, Paris (2009)

Liu, N.N., Cao, B., Zhao, M., Yang, Q.: Adapting neighborhood and matrix factorization models for context aware recommendation. Workshop on Context-Aware Movie Recommendation, pp. 7–13, Barcelona (2010a)

Liu, N.N., Zhao, M., Xiang, E., Yang, Q.: Online evolutionary collaborative filtering. Fourth ACM Conference on Recommender Systems, pp. 95–102, Barcelona (2010b)

Lu, Z., Agarwal, D., Dhillon, I.S.: A spatio-temporal approach to collaborative filtering. Third ACM Conference on Recommender Systems, pp. 13–20, New York (2009)

Ma, S., Li, X., Ding, Y., Orlowska, M.E.: A recommender system with interest-drifting. Eighth International Conference on Web Information Systems Engineering, pp. 633–642, Nancy (2007)

Min, S., Han, I.: Detection of the customer time-variant pattern for improving recommender systems. Expert Syst. Appl. **28**(2), 189–199 (2005)

Montañés, E., Quevedo, J.R., Díaz, I., Ranilla, J.: Collaborative tag recommendation system based on logistic regression. ECML PKDD Discovery Challenge 2009, pp. 173–188, Paris (2009)

Oku, K., Nakajima, S., Miyazaki, J., Uemura, S.: Context-aware SVM for context-dependent information recommendation. Seventh International Conference on Mobile Data Management, pp. 109–109, Nara (2006)

Palmisano, C., Tuzhilin, A., Gorgoglione, M.: Using context to improve predictive modeling of customers in personalization applications. IEEE Trans. Knowl. Data Eng. **20**(11), 1535–1549 (2008)

Panniello, U., Gorgoglione, M., Palmisano, C.: Comparing pre-filtering and post-filtering approach in a collaborative contextual recommender system: an application to e-commerce. Tenth International Conference on E-Commerce and Web Technologies, pp. 348–359, Linz (2009a)

Panniello, U., Tuzhilin, A., Gorgoglione, M., Palmisano, C., Pedone, A.: Experimental comparison of pre- vs. post-filtering approaches in context-aware recommender systems. Third ACM Conference on Recommender Systems, pp. 265–268, New York (2009b)

Panniello, U., Tuzhilin, A., Gorgoglione, M.: Comparing context-aware recommender systems in terms of accuracy and diversity: which contextual modeling, pre-filtering and post-filtering methods perform the best. User Model. User-Adapt. Interact, Special Issue on Context-Aware Recommender Systems (2013)

Park, M.-H., Hong, J.-H., Cho, S.-B.: Location-based recommendation system using Bayesian user's preference model in mobile devices. In: Indulska, J., Yang, L., Ungerer, L., Cao, J. (eds.) Proceedings of Ubiquitous Intelligence and Computing, 4611, pp. 1130–1139. Springer, Berlin (2007)

Parra, D., Amatriain, X.: Walk the talk: analyzing the relation between implicit and explicit feedback for preference elicitation. Nineteenth International Conference on User Modeling, Adaption, and Personalization, pp. 255–268, Girona (2011)

Pradel, B., Sean, S., Delporte, J., Guérif, S., Rouveirol, C., Usunier, N., Fogelman-Soulié, F., Dufau-Joel, F.: A case study in a recommender system based on purchase data. Seventeenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 377–385, San Diego (2011)

Rendle, S.: Time-variant factorization models. In Rendle, S. (ed.), Context-aware ranking with factorization models. ISBN 978-3-642-16897-0 (2011)

Rendle, S., Gantner, Z., Freudenthaler, C., Schmidt-Thieme, L.: Fast context-aware recommendations with factorization machines. Thirty-Fourth International ACM SIGIR Conference on Research and Development in Information, pp. 635–644, Beijing (2011)

Sarwar, B., Karypis, G., Konstan, J., Riedl., J.: Analysis of recommendation algorithms for e-commerce. Second ACM Conference on Electronic Commerce, pp. 158–167, Minneapolis (2000)

Shani, G., Gunawardana, A.: Evaluating recommendation systems. In: Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.), Recommender Systems Handbook. ISBN 978-0-387-85820-3 (2011).

Stormer, H.: Improving e-commerce recommender systems by the identification of seasonal products. Twenty-second Conference on Artificial Intelligence, pp. 92–99, Vancouver, British Columbia (2007).

Töscher, A., Jahrer, M.: The bigchaos solution to the Netflix prize 2008. Netflix Prize, Report (2008)

Töscher, A., Jahrer, M., Legenstein, R.: Improved neighborhood-based algorithms for large-scale recommender systems. Second KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition, pp. 4:1–4:6, Las Vegas (2008)

Tang, T.Y., Winoto, P., Chan, K.C.C.: On the temporal analysis for improved hybrid recommendations. 2003 IEEE/WIC International Conference on Web Intelligence, pp. 214–220, Beijing (2003).

Vargas, S., Castells, P.: Rank and relevance in novelty and diversity metrics for recommender systems. Fifth ACM Conference on Recommender Systems, pp. 109–116, Chicago (2011).

Vildjiounaite, E., Kyllönen, V., Hannula, T., Alahuhta, P.: Unobtrusive dynamic modelling of TV program preferences in a household. In: Tscheligi, M., Obrist, M., Lugmayr, A. (eds.), Changing Television Environments. ISBN 978-3-540-69477-9 (2008)

Weng, S.-S., Binshan, L., Chen, W.-T.: Using contextual information and multidimensional approach for recommendation. Expert Syst. Appl. **36**(2009), 1268–1279 (2009)

Whitrow, G. J.: Time in history: views of time from prehistory to the present day. Oxford University Press, Oxford. ISBN 0-19-285211-6 (1988)

Wu, Y., Yan, Q., Bickson, D., Low, Y., Yang, Q.: Efficient multicore collaborative filtering. KDD-Cup Report, San Diego (2011)

Xiang, L., Yang, Q.: Time-dependent models in collaborative filtering based recommender system. 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, pp. 450–457, Milano (2009)

Xiang, L., Yuan, Q., Zhao, S., Chen, L., Zhang, X., Yang, Q., Sun, J.: Temporal recommendation on graphs via long- and short-term preference fusion. Sixteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 723–732, Washington (2010)

Xiong, L., Chen, X., Huang, T., Schneider, J., Carbonell, J.G.: Temporal collaborative filtering with Bayesian probabilistic tensor factorization. SIAM Data Mining **2010**, 211–222 (2010)

Zhan, S., Gao, F., Xing, C., Zhou, L.: Addressing concept drift problem in collaborative filtering systems. ECAI 2006 Workshop on Recommender Systems, pp. 34–39, Riva del Garda (2006)

Zhang, M., Hurley, N.: Novel item recommendation by user profile partitioning. 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, pp. 508–515, Milano (2009)

Zheng, N., Li, Q.: A recommender system based on tag and time information for social tagging systems. Expert Syst. Appl. **38**(4), 4575–4587 (2011)

Zhou, T., Kuscsik, Z., Liu, J.-G., Medo, M., Wakeling, J.R., Zhang, Y.-C.: Solving the apparent diversity-accuracy dilemma of recommender systems. Natl. Acad. Sci. **107**(10), 4511–4515 (2010)

Ziegler, C., McNee, S.M., Konstan, J.A., Lausen, G.: Improving recommendation lists through topic diversification. Fourteenth International Conference on World Wide Web, pp. 22–32, Chiba (2005)

Zimdars, A., Chickering, D.M., Meek, C.: Using temporal data for making recommendations. Seventeenth Conference in Uncertainty in Artificial Intelligence, pp. 580–588, Seattle (2001)

Author Biographies

Pedro G. Campos is an Assistant Professor at the Information Systems Department, Universidad del Bío-Bío, Chile, and a Ph.D. student in the Information Retrieval Group at the Computer Science Department of Universidad Autónoma de Madrid (UAM). He holds a M.Eng. in Software Engineering from the Universidad de Tarapacá, Chile, and a M.Sc. in Computer Science and Telecommunications from UAM. His current research is mainly focused on recommender systems, in particular on the exploitation and evaluation of time information for generating personalized recommendations.

Fernando Díez is an Assistant Professor of Computer Science at Universidad Autónoma de Madrid, Spain, where he received the Ph.D. degree in Computer Engineering. He also holds MSc in Mathematics from the Universidad Complutense de Madrid. During the late 80s and 90s he worked as a researcher in different computer science R&D departments, in companies like Telefónica R&D, and Intelligent Decision Systems. His main research interests currently rely on Information Retrieval, Recommender Systems, e-Learning Systems, and Model Driven Engineering.

Iván Cantador is a Lecturer of Computer Science at Universidad Autónoma de Madrid, where he received the PhD degree in 2008. During his doctoral studies, he was a research consultant at the Knowledge Media Institute, and a research visitor at University of Southampton. After earning his PhD, he worked as a research associate at University of Glasgow, and was a postdoctoral visitor at the Free University of Bozen-Bolzano. He has co-authored over fifty publications in international journals and conferences on information retrieval, recommender systems, and Semantic Web.