Sequence-Aware Recommender Systems

MASSIMO QUADRANA, ContentWise
PAOLO CREMONESI, Politecnico di Milano
DIETMAR JANNACH, AAU Klagenfurt

Recommender systems are one of the most successful applications of data mining and machine-learning technology in practice. Academic research in the field is historically often based on the matrix completion problem formulation, where for each user-item-pair only one interaction (e.g., a rating) is considered. In many application domains, however, multiple user-item interactions of different types can be recorded over time. And, a number of recent works have shown that this information can be used to build richer individual user models and to discover additional behavioral patterns that can be leveraged in the recommendation process.

In this work, we review existing works that consider information from such sequentially ordered user-item interaction logs in the recommendation process. Based on this review, we propose a categorization of the corresponding recommendation tasks and goals, summarize existing algorithmic solutions, discuss methodological approaches when benchmarking what we call *sequence-aware recommender systems*, and outline open challenges in the area.

CCS Concepts: • **Information systems → Recommender systems**; *Collaborative filtering*; • **Computing methodologies** → *Learning from implicit feedback*;

Additional Key Words and Phrases: Sequence, session, trend, algorithms, dataset, evaluation, recommendation

ACM Reference format:
Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. 2018. Sequence-Aware Recommender Systems. *ACM Comput. Surv.* 51, 4, Article 66 (July 2018), 36 pages.
https://doi.org/10.1145/3190616

1 INTRODUCTION

Recommender Systems (RS) are software applications that support users in finding items of interest within larger collections of objects, often in a personalized way. Today, such systems are used in a variety of application domains, including, for example, e-commerce or media streaming, and receiving automated recommendations of different forms has become a part of our daily online user experience. Internally, such systems analyze the past behavior of individual users or of a user community as a whole to detect patterns in the data. On typical online sites, various *types* of relevant actions of a user can be recorded, for example, that a user views an item or makes a purchase,

This work was mainly carried out when Massimo Quadrana was a Ph.D. student at Politecnico di Milano.

Authors' addresses: M. Quadrana, Contentwise, Via Schiaffino 11, 20158 Milano, Italy; email: massimo.quadrana@contentwise.com; P. Cremonesi, Politecnico di Milano, Dipartimento di Elettronica, Informatica e Bioingegneria (DEIB), Via Ponzio 34/5, 20133 Milano, Italy; email: paolo.cremonesi@polimi.it; D. Jannach, Institute of Applied Informatics, AAU Klagenfurt, Universitätsstraße 65-67, 9020 Klagenfurt am Wörthersee, Austria; email: dietmar.jannach@aau.at.
Current address: M. Quadrana, Pandora Media, 2101 Webster St, Oakland, CA 94612, USA.

https://doi.org/10.1145/3190616

and several of the actions of a single user may relate to the same item. These recorded actions and the detected patterns are then used to compute recommendations that match the preference profiles of individual users.

In academic environments, the predominant problem abstraction is that of matrix completion where we are given a user-item rating matrix and the goal is to predict the missing values. This abstraction is generally well-suited to train machine-learning models that aim to capture longer-term user preference profiles. The corresponding algorithms, however, typically implement no specific means to take the users' short-term behavior or intents into account in their recommendations; nor are they designed to use the rich information that is contained in the sequentially ordered user interaction logs that are often available in practical applications.

In practice, however, there are many application scenarios where considering short-term user interests and longer-term sequential patterns can be central to the success of a recommender. A typical example problem setting is that of *session-based recommendation* [45, 56], where no longer-term user histories are available. Instead, we have to adapt the recommendations according to the assumed short-term interests of an anonymous user. The goal in such scenarios usually is to recommend objects that match a given sequence of user actions.

Typical algorithmic approaches in that context learn to predict the best next item from sequential user interaction logs. Considering such sequences is, however, not only relevant for the short-term adaptation of the recommendations. The sequential logs can also be used to derive longer-term behavior patterns, e.g., to detect *interest drifts* of individual users over time [85], to identify short-term *popularity trends* in the community that can be exploited by recommendation algorithms [54, 62], or to reason about the best point in time to *remind* users of certain items they have seen or purchased before [70]. Finally, there are application domains where the recommendation of one item (e.g., an accessory) only makes sense after some other object was purchased. Such weak or strict ordering constraints might correspondingly be learned from the data and considered by a sequence-aware recommender.

Overall, sequence-aware recommendation scenarios are highly relevant in practice and a number of relevant works were proposed in the recent past. Research in the field is, however, comparably scattered and no common understanding of the different facets of the problem exists. In this survey work, we therefore (i) categorize the various scenarios of sequence-aware recommendation approaches in the academic literature, (ii) we review the various algorithmic approaches that were proposed to extract and leverage patterns from interaction logs, and (iii) we finally discuss specific issues when benchmarking different recommendation methods. One of the major goals of the review in that context is to lay the path for more standardized and better reproducible research works in the field.

The article is organized as follows. In Section 2 and Section 3, we characterize and categorize different types of sequence-aware recommendation problems that can be found in the literature. Section 4 reviews existing algorithms and Section 5 discusses methodological questions regarding their evaluation and comparison. Section 6 finally gives a brief outlook on future research directions.

2 CHARACTERIZING SEQUENCE-AWARE RECOMMENDER SYSTEMS

Sequence-aware recommendation problems are different from the traditional matrix-completion setup in a number of ways. Figure 1 gives a high-level overview of the problem, its inputs, outputs, and specific computational tasks. In general, the ordering of the objects can be relevant both with respect to the inputs and to the outputs. We will discuss these aspects in more detail next.

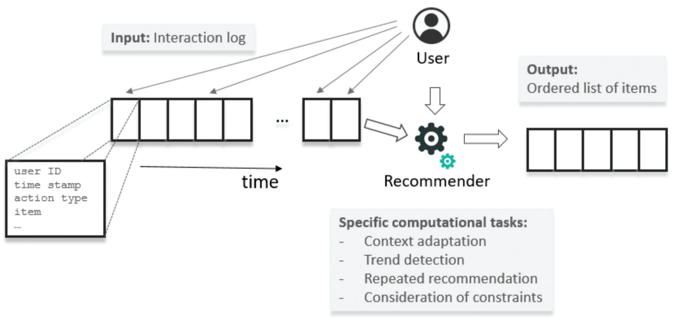
Input: Interaction log
User
Output: Ordered list of items
user ID
time stamp
action type
item
…
time
Recommender
Specific computational tasks:
- Context adaptation
- Trend detection
- Repeated recommendation
- Consideration of constraints

Fig. 1. High-level overview of sequence-aware recommendation problems.

2.1 Inputs, Outputs, Computational Tasks, and Abstract Problem Characterization

Inputs. The main input to sequence-aware recommendation problems is an ordered and often timestamped list of past user actions. Users can be already known by the system or anonymous ones. Each action can be associated with one of the recommendable items. Finally, each action can be one of several pre-defined *types* and each action, user, and item may have a number of additional attributes. Overall, the inputs can be considered as a sort of enriched clickstream data.

In the traditional matrix completion setup, all ratings are attached to one of the known users and items. We do not require this to be the case for sequence-aware recommenders. Anonymous user actions are not uncommon, for example, in the e-commerce domain, where users are often not logged in. Nonetheless, relevant information can be extracted from past anonymous sessions. We also do not require that each action is related to an item, since, for example, relevant information can be extracted from the users' search or navigation behavior as well [53]. Finally, in most application scenarios, each action will have an assigned action type (e.g., item-view, item-purchase, add-to-cart). And, depending on the domain, additional information might be available that describes further details of an action (e.g., whether an item was discounted when the action took place), the users (e.g., demographics), or the items (e.g., metadata features).

Generally, such forms of input data are available in many practical applications, for example, in the form of application or web server logs. Usually, we do not, however, assume to have larger quantities of *explicit* ratings available in sequence-aware recommender systems.

Outputs. The output of a sequence-aware recommender are ordered lists of items as will be described more formally below. In this general form, the outputs are similar to those of a traditional "item-ranking" recommendation setup. However, in some sequence-aware recommendation scenarios, the ordering of the objects in the recommendation list can be relevant as well. Instead of considering the list of recommendations as a set of *alternatives* for the user, there are scenarios where the user should consider *all* recommendations and do this in the provided order. Typical examples include the recommendation of a sequence of tracks in music recommendation or the recommendation of a series of learning courses. We will describe such application scenarios in more detail later in the article.

Computational Tasks. Different computational tasks of sequence-aware recommenders can be identified in the literature. Most commonly, a task that is not present in traditional matrix completion setups is the identification of sequence-related patterns in the recorded user actions. These can be *sequential* patterns, where the order of the actions is relevant, or they can be *co-occurrence* patterns, where it is only important that two actions happened together, for example, within the same session. In some cases, also *distance* patterns can be relevant, for example, when the problem is to compute a good point in time to remind the user of something through a recommendation. Note that the corresponding patterns do not have to be made explicit, as often done, e.g., in sequential pattern mining [79], but can be implicitly encoded in complex machine-learning models as well.

Besides the identification of such patterns that are subsequently used in the recommendation task, another computational task of a sequence-aware recommender can be to reason about *order constraints*. Such constraints can be either prescribed and given for an application domain as strict constraints (e.g., in terms of a given curriculum for the learning course recommendation problem), given as heuristics (e.g., in terms of track transition rules for next-track music recommendation), or be implicitly derived from the given input data as a sort of weak constraints.

Finally, the patterns (or more generally, the learned models) that were identified in the data and the constraints have to be related with the point in time for which the recommendation is sought for. In a session-based recommender, one might consider the last few user actions and then look for past sessions that were similar to the current one. On the other hand, when a recommender is used as a reminder for the repeated purchase of consumables, the distance (in time) to the last purchase action of the user to the present time might be relevant.

Abstract Characterization. Adopting the formalisms of [3], we can describe the problem at a more formal, abstract level as follows. Let C be a set of users and I a set of recommendable items. In contrast to matrix-completion problems, we are not interested in predicting a utility value for each $i \in I$ and for each $c \in C$, but in computing an ordered list of objects L of length k for each user, where each element of $l \in L$ corresponds to an element of $i \in I$. Technically, each sequence L is an element of the set of all permutations up to length k of the powerset of I, that is, $L \in S_k(\mathcal{P}(I))$. We denote this latter set of possible lists as L^*.

Let u be a function that returns a utility score of a given sequence L for a user c, that is, $u : C \times L^* \rightarrow R$. The sequence-aware recommendation problem then consists of determining the sequence $l'_c \in L^*$ that maximizes the score for the user, i.e.,

$$\forall c \in C, \quad l'_c = \underset{l \in L^*}{\arg\max}\ u(c, l). \tag{1}$$

The main problem in recommender systems is to learn or extrapolate the utility function u from some given data. In the matrix completion problem, which underlies the work in [3], the input is a sparse matrix of user-item ratings. In sequence-aware recommender systems, we in contrast assume that the underlying data is a dataset D consisting of sequence[1] of user actions where each user action $A \in D$ has a number of attributes. A sequence dataset D can be considered as an enriched log of actions of a user community, where the attributes of each action A includes some sort of user ID[2] and additional optional attributes like the *action type* (e.g., an item view or click event) or a timestamp.

Overall, our function u is not limited to characterizing utility scores for individual items, but for entire ordered lists of items. This makes it possible to consider additional aspects of utility in sequence-aware recommendation problems, including the diversity of the set as a whole, the quality of the ordering itself, for example, in terms of transitions between objects, or the degree of fulfillment of weak or strict order constraints in L. How these quality factors are considered within existing algorithms will be discussed later in this work in Section 4.

Generally, the design of the utility function u depends on the specific type of value the recommender system should provide to the user, or its *purpose* in the sense of [50]. In the literature

[1]A sequence, as usual, is considered as an ordered set of objects.
[2]In practice, the user ID can either refer to a known user or it is created from a cookie in an ongoing user session.

on recommender systems, researchers often do not explicitly discuss the underlying purpose of the system, which could be information filtering but also discovery support. Instead, they focus on optimizing an abstract computational task like predicting a hidden rating. The situation in the context of sequence-aware recommenders is often similar with the difference that the computational task is mainly to predict the hidden elements of a session given the session beginning. While the performance of an algorithm that predicts the next hidden user action can be assessed with standard measures from information retrieval (such as precision and recall), in other cases specific measures (e.g., diversity metrics) are required in the evaluation process.

2.2 Relation to Other Areas

Implicit-Feedback Recommender Systems. Our characterization of the sequence-aware recommendation problem mainly targets scenarios in which we observe the individual and collective behavior of a user community over time instead of asking for explicit item ratings. A number of research works exist that focus on *implicit* user feedback like purchase events. The problem formulation is, however, often based again on matrix completion, where multiple interactions of one user with an item are not taken into account. *Explicit* item ratings, on the other hand, *can* also be taken into account in a sequence-aware recommender as one of several types of user actions. One potential problem in that context, however, is that the point in time when users provide a rating can be quite different from the point in time when they consumed or purchased an item (e.g., when registering for a movie recommendation service, users initially rate a bunch of movies they have watched in the past). The sequence and timestamp of the ratings might therefore mislead a sequence-aware recommender.

Context-Aware and Time-Aware Recommender Systems. In some of the application scenarios discussed in the next sections, sequence-aware recommender systems represent a special form of context-aware recommender systems. In session-based recommendation, the users' short-term intents, which can be estimated from their very last actions, can represent an important piece of context information to be taken into account when recommending [56].

Time-aware recommender systems (TARS) usually consider time information that is associated with past user actions to adapt the recommendations accordingly (see [17] for an overview). TARS share a number of commonalities with sequence-aware recommenders, for example, in terms of how we can compare different approaches in offline settings. The focus of sequence-aware recommenders is, however, often less on the exact point of time of the past user interactions, but on the sequential order of the events. Furthermore, a number of proposals on time-aware recommenders mainly rely on the matrix completion problem setting when modeling temporal dynamics [68].

Research in Other Related Fields. Some aspects of sequence-aware recommender systems were finally explored also in neighboring fields. Examples are the problem of *query suggestion* in the field of information retrieval or the problem of *interest drift* in the more general field of user modeling. In this article, we concentrate on works where the recommendation problem itself is the main focus, in contrast to works that, for example, aim to develop methods to capture changes in the user preferences over time. When searching for papers to consider in our survey, we therefore used a corresponding search string and selection strategy when we queried a digital library, as will be described in more detail in Section 3.5.

3 A CATEGORIZATION OF SEQUENCE-AWARE RECOMMENDATION TASKS

We identified four main goals in the academic literature that can be achieved with the help of sequence-aware recommender systems in different application scenarios:

(1) Context Adaptation
(2) Trend Detection
(3) Repeated Recommendation
(4) Consideration of Order Constraints and Sequential Patterns

We will discuss these four categories in more detail in the next sections and will then also look at typical application domains for sequence-aware recommenders. Note that all types of problem settings discussed next are based on the same formal problem characterization described in Equation (1), but require specific algorithmic approaches that use the sequence information in the input datasets (see Section 4). The problems are also not mutually exclusive and multiple aspects (e.g., trends and repetitions) can be considered in parallel, as was done, for example, in [62] for the e-commerce domain.

3.1 Context Adaptation

In many domains, the relevance of a recommendable item not only depends on the users' general preferences, but also on their current situation and their short-term intents and interests. Context-aware recommenders take such additional types of information into account. Typical contextual factors in the literature include the user's geographical position, the current weather, or the time of the day [4]. Context factors like these are examples of what is called the *representational context* [28], which is defined by a predefined set of "observable" context variables.

Contextual factors like the user's current shopping intent in an e-commerce setting or their current mood are, however, not directly observable. These types of information, which represent what is called the user's *interactional context*, therefore have to be derived from the users most recent actions and eventually on behavioral patterns of the user and the community as a whole [40, 87]. Considering interactional context factors is particularly important for systems where there are many new or *anonymous* users. Since no historical data is available about their past preferences, it is important to make full use of interactional context information, as representational context information can only help to partition anonymous users into coarse-grained categories, without any real personalization [32]. Overall, understanding the users' situation and goals and making context-adapted recommendations from past interaction data represents a main goal of sequence-aware recommender systems.

Categorization Based on Importance of Long- and Short-Term Interactions. Depending on the availability of historical data for individual users and the importance of focusing on the most recent interactions, we can differentiate between the following *types* of context-adaptation situations.

— *Last-N Interactions-Based Recommendation:* In these scenarios, only the last N user actions are considered. A typical problem setting is that of predicting the next location (or *check-in*) in a location-aware recommender system [21, 72, 76]. The reason to limit oneself to the last actions could be that not many past interactions of that type exist or that the other previous actions of the same type (e.g., check-in events) are not predictive for the next action.
— *Session-Based Recommendation:* In this problem scenario only the last sequence of actions of a user is known and this sequence of actions is limited to a *session*, that is, a limited period of time when the user interacted with the site. Typical application examples include news recommendation [32], e-commerce, video, and classified advertisement recommendation [46].
— *Session-Aware Recommendation:* Finally, there are situations in which we have knowledge both about the users' actions in the last session *and* about their past behavior. This type of problem setting occurs if we have returning customers that can be identified. In this

situation, a sequence-aware recommender system can be based on a combination of long-term and short-term interest models, for example, in e-commerce settings or for app recommendation [9, 40, 56, 91].

Note that our problem definition in Equation (1) covers all three scenarios, that is, the output is a ranked list of items. In the case of a *session-aware* adaptation problem, the underlying sequence dataset D is, however, usually split into two components, where one that contains the older interactions is used for building a long-term model, and the other is used to consider short-term user intents. How the different models are learned or combined then depends on the specific algorithmic approach that is used to maximize the utility function u, which might, for example, return higher scores when recommendations are a mix of familiar and novel items for the user.

On Utility Functions and Recommendation Purposes. Most of the papers on context-adaptation problems in the literature do not make explicit statements about the characteristics of the utility function in the application scenarios they considered. As mentioned above, they in most cases implicitly define the goal through the evaluation procedure and aim to predict hidden elements of a given user session. Thereby, they implicitly assume that this next action is in some sense the best recommendation for the given purpose.

The task of recommenders in context-adaptation scenarios can most often be characterized as "find matching items" for a given session beginning, without any further explicit specification of what represents a good recommendation. In some works—and in practical environments—a number of more specific purposes can be identified (see also [50]). The task of a recommender can be, for example, to create a list of *alternatives* for the currently inspected items (similar items). In other applications, in contrast, the task can be to determine *complements*, for example, accessories to a main shopping item in e-commerce. In yet other application domains, the recommendations should represent suitable or logical *continuations* of either the current session (e.g., next-track music recommendations) or the user's longer-term behavior (e.g., next-basket recommendations). Finally, we can differentiate if the user is assumed to pick one of the recommendations (e.g., one alternative in e-commerce scenarios), or consider all of them together (e.g., playlist recommendation for audio and video streaming). This latter scenario was recently addressed in [101] for the news domain. In their work, the authors model the user's expected utility of an item during the course of a session and try to diversify the recommended content within a session accordingly.

3.2 Trend Detection

The detection of trends in a given sequence dataset is another potential, but less explored, goal that can be accomplished by sequence-aware recommenders. We can distinguish between the following types of information that can be extracted from sequential log information to be used in the recommendation process.

— *Community Trends.* Considering the popularity of items within a user community can be important for successful recommendations in practice, for example, in streaming media recommendation [35]. Since the popularity of items can change over time in different domains, sequence-aware recommenders can aim to detect and utilize popularity patterns in the interaction logs to improve the recommendations. Such trends can be long-term (e.g., things becoming outdated or out-of-fashion over time), seasonal, or reflect short-term and one-time popularity peaks. In the fashion domain, for example, considering the community trends of the last few days can represent a successful strategy when selecting items for recommendation [54].

— *Individual Trends.* Changes in the interest in certain items can also happen at an individual level. These interest changes can be caused when there is a "natural" interest drift, for example, when users grow up, or when their preferences change over time, for example, due to the influence of other people, due to exceptional events, or when they discover something new. In the news domain, for example, individual interests change over time and are influenced by global and local news trends [75]. Another example problem is the task of modeling the dynamics of the musical taste of users [85].

3.3 Repeated Recommendation

In some application domains, recommending items that the user already knows or has purchased in the past can be meaningful. Such scenarios are not considered at all in the traditional matrix completion setup. We can identify the following categories of repeated recommendation scenarios.

— *Identifying Repeated User Behavior Patterns.* Past interaction logs can be used by sequence-aware recommenders to identify patterns of repeated user behavior. A typical application example could be the repeated purchase of consumables, like printer ink. Such patterns can be both mined from the behavior of individual users, as in [109, 122, 123], or the community as a whole. Repeated user actions are particularly relevant for app recommendation problems. In this context, patterns of repeated user behavior can be used to provide shortcuts to applications that are frequently launched in a certain sequence by the user. An example is to suggest to launch the "e-mail" or "calls" app after opening the "contacts" app. The general goal here is to enhance the user experience with the device [9, 78, 87].
— *Repeated Recommendations as Reminders.* In a different scenario, repeated recommendations can help to *remind* users of things they found interesting in the past. Depending on the domain, these reminders could relate to objects that the user has potentially forgotten (e.g., an artist that she or he liked in the past), or to objects that the user has recently interacted with [70]. The latter scenario is particularly relevant in e-commerce, and the recommendation of recently inspected items is common on platforms like Amazon.com.

Note that from the viewpoint of our problem characterization in Section 2, the two scenarios are identical. In the first case, however, there is often an underlying "logical" reason why an item should be recommended again, which is not the case for scenarios of the latter class, that is, the recommendation of assumedly "forgotten" or recently relevant items.

In both mentioned scenarios, besides the selection of items to repeatedly recommend, a sequence-aware recommender has to reason about the *timing* of the recommendations. In the reminding scenario in e-commerce, the time frame to remind users of previously seen items might be narrow and objects may become obsolete soon, for example, if they were not purchased after a few view events. Nonetheless, always reminding users of items they have inspected in the last session might be inappropriate if the user's current shopping intent does not match that of the previous session. In the context of the recommendation of consumables, items can be repeatedly recommended after longer periods of time, for example, weeks or even months. Proper timing can, however, still be important and such types of repeated recommendations share similar problems as *proactive* recommenders [26], for example, that their recommendations might interrupt the user at the wrong time.

3.4 Consideration of Order Constraints and Observed Sequential Patterns

In Section 3.1 on short-term context adaptation, we have discussed different tasks of sequence-aware recommender systems in which the order of objects—either in the logs, in the current session, or in the recommendations—can play a role. Considering such orderings can be central to sequence-aware recommendation tasks also outside the context of the user's last session, which is why we discuss this aspect in some more depth in this section.

Specifically, there are two types of information about sequentiality one can additionally consider when determining a suitable order of the recommendations.

(a) First, there can be "external" domain knowledge in the form of strict or weak ordering constraints that prescribes the ordering. In the domain of recommending a sequence of learning courses, for example, there might be strict requirements regarding the order of different courses that have to be considered by the recommender (e.g., when one cannot attend one course before another one was completed [88, 113]). In the domain of movie recommendation, in contrast, it might be reasonable to recommend a sequel to a movie only after a user has watched the preceding episode. Such a constraint is, however, not necessarily strict.

(b) Second, one can try to identify such sequential consumption patterns from the user behavior, and, for example, automatically infer that users who watched a certain movie later on watched its sequel. From a technical perspective, sequential pattern mining techniques have, for example, been applied in different application domains of recommenders (e.g., for predicting the next navigation actions of users on websites [83, 86] or to find next tracks to play in music recommendation problems [15]).

When considering these factors that can influence the ordering, a number of application-specific variations might have to be considered, among them the following.

— *Importance of the Order of the Recommendations.* Depending on the specifics of the application scenario and the goals of the recommendation service, the order of the recommended items can be relevant or not, even in the same domain. In the context of next-track music recommendation, one can try to determine a playlist continuation where all elements are generally a good fit for the current listening session [57]. In contrast, one could, however, also try to make sure that also the transitions between the tracks are smooth or that the resulting playlist has certain characteristics, for example, a continuous increase of the tempo [80].
— *Importance of the Exact Order of the Past Events.* Similarly, the exact order of the past events may or may not be relevant for the recommendation task. Again in the domain of the music recommendation, one can try to look for sequential patterns in past sessions as in [40] or simply consider track co-occurrence patterns, as done in the neighborhood-based approach in [15]. In cases where the order is relevant, one might additionally consider the age of the individual events in the log and reduce the importance of older events.
— *Existence of Implicit Order Constraints.* When recommending complements (e.g., accessories for a given item), there might be implicit constraints of what can be reasonably recommended and these constraints can depend on the specific domain or product category. One can recommend a memory card as an accessory to a camera, but not the other way around. For other categories, however, recommendations in both directions can be plausible.

In general, besides the identification of sequential patterns from past data to select and rank items, an additional task for the recommender is to ensure that such strict and weak order constraints are respected in the resulting lists.

3.5 Categorization of Existing Works

In this section, we will classify existing research works according to our categorization scheme in order to obtain a better understanding of which aspects of sequence-aware recommenders are comparably well explored and which areas need further investigation.

We selected the papers to be considered in this categorization as follows. We issued a query to the ACM Digital Library using a set of relevant search terms,[3] sorted the first 1,000 search results by relevance, and manually inspected the abstracts of the first 100 papers. If a paper made explicit use of sequences of user actions, it was considered relevant and its referenced papers were scanned as further potential candidates to consider. Table 1 shows a summary of the papers that were finally considered relevant for this work.

In the table, the first column shows the context-adaptation type of each approach according to our categorization. The second column indicates which form of ordering constraints were considered; the third column mentions the main application domain addressed in the papers.

3.5.1 Categorization Based on Context-Adaptation Type. We differentiate between session-aware, session-based, and last-interaction adaptation approaches. Figure 2 shows the number of research works that fall into the different categories.

We can see that a large fraction of the papers focuses only on the very last visited object when determining the next object to be recommended. A slightly smaller number of papers consider session-based recommendation scenarios. However, we can observe increased interest in such session-based recommendation problems in the recent years, which is fueled both by the industry[4] and by the emergence of new deep learning–based sequence-learning techniques, as we will discuss in Section 4. In fact, more than half of the papers on session-based and session-aware recommendation were published starting from 2014. Finally, even fewer papers also take the long-term preferences of the user into account. While short-term user intents might in many cases be more relevant than long-term preferences, some works—like the ones by [56, 91]—show that considering long-term preferences can be important to achieve more accurate recommendations.

The limited number of works on session-aware systems might partially be due to the lack of publicly available benchmark datasets. Overall, however, the results shown in Figure 2 indicate more research is required to better understand the interplay between long-term and short-term preferences in sequence-aware recommenders.

3.5.2 Order Constraints. As the column labeled "Ord." in Table 1 shows, most existing works in sequence-aware recommendation rely on implicitly derived (and correspondingly weak) ordering constraints and only a few works address scenarios where explicit and strong order constraints have to be considered. This can again be caused by the focus of the research community on certain application domains like media (movie) recommendation [55], where ordering constraints exist—consider movie sequels or follow-up news stories—but are often not considered in the corresponding algorithmic approaches.

3.5.3 Categorization Based on Domain. Figure 3 finally shows in which application domain sequence-aware recommenders were investigated. In the following, we give examples of typical research works in each domain.

E-commerce is the most often investigated domain, which is not surprising as session-based recommendation and session-aware recommendation, as defined above, are common tasks in e-

[3]The exact query was "sequence sequential trend next basket bundles repeat session order + recommend."
[4]For example, a major e-commerce platform provided a new dataset in the context of the 2015 ACM RecSys Challenge about session-based recommendation.

Table 1. Categorization of Works Regarding Context-Adaptation Type, Order Constraints, and Domain

Paper	**Context**	**Ord.**	**Domain**	**Paper**	**Context**	**Ord.**	**Domain**
Baeza-Yates et al. [9]	SA	I	APP	Parameswaran et al. [88]	–	S	CS
Chen et al. [19]	LI	WI	MUS	Pauws et al. [89]	–	S	MUS
Chen et al. [20]	LI	WI	MUS	Quadrana et al. [91]	SA	I	EC,VID
Cheng et al. [21]	LI	I	POI	Reddy et al. [92]	–	W	CS
Chou et al. [22]	LI	I	MUS	Rendle et al. [94]	LI	I	EC
Feng et al. [30]	LI	I	POI	Rudin et al. [95]	LI	I	EC
Garcin et al. [32]	SB	I	NEWS	Shani et al. [97]	LI	I	EC,WWW
Grbovic et al. [37]	LI	I	EC	Soh et al. [98]	SB	I	OTH
Greenstein-Messica et al. [38]	SB	I	EC	Song et al. [99]	SB	I	EC,VID
Hariri et al. [40]	SB	WI	MUS	Song et al. [100]	SB	I	NEWS
He et al. [42]	SB	I	QRY	Sordoni et al. [102]	SB	I	QRY
He and McAuley [43]	LI	I	EC,POI	Tagami et al. [103]	LI	I	ADS
He et al. [41]	LI	I	POI	Tavakol and Brefeld [104]	SB	I	EC
Hidasi et al. [46]	SB	I	EC,VID	Trevisiol et al. [105]	SB	I	WWW,NEWS
Hidasi et al. [45]	SB	I	EC	Turrin et al. [106]	SB	I	MUS
Hosseinzadeh Aghdam et al. [47]	LI	I	MUS	Twardowski [107]	SB	I	EC
Hsueh et al. [48]	LI	I	EC	Vasile et al. [108]	SB	WI	MUS
Jannach et al. [56]	SA	I	EC	Wang and Zhang [109]	LI	I	EC
Jannach et al. [57]	SA	WI	MUS	Wang et al. [110]	LI	I	EC
Jannach and Ludewig [54]	SA	–	EC	Wu et al. [111]	SB	WI	MUS
Lerche et al. [70]	SA	I	EC	Xiang et al. [112]	SA	I	OTH
Letham et al. [71]	LI	I	EC	Xu et al. [113]	–	S	CS
Lian et al. [72]	LI	I	POI	Yan et al. [114]	SB	I	QRY
Lim et al. [73]	LI	S	POI	Yap et al. [115]	LI	I	WWW
Liu et al. [76]	LI	SI	POI	Yu and Riedl [117]	LI	S	OTH
Liu et al. [77]	LI	I	POI	Yu et al. [116]	LI	I	EC
Lu et al. [78]	SB	I	APP	Zang et al. [118]	LI	I	EC
Maillet et al. [80]	–	W	MUS	Zhang et al. [121]	LI	I	ADS
McFee and Lanckriet [81]	LI	WI	MUS	Zhang et al. [120]	LI	I	POI
Mobasher et al. [83]	SB	I	WWW	Zhao et al. [122]	LI	I	EC
Moling et al. [84]	SB	–	MUS	Zhao et al. [123]	LI	I	EC
Moore et al. [85]	LI	WI	MUS	Zheleva et al. [124]	SA	–	MUS
Nakagawa and Mobasher [86]	SB	I	WWW	Zhou et al. [125]	LI	I	WWW
Natarajan et al. [87]	SA	I	MUS,APP	Zimdars et al. [126]	LI	I	WWW

Context: SA: Session-aware, SB: Session-based, LI: Last-N interactions; **Order Constraints:** W: Weak, I: Inferred, S: Strong; **Application Domain:** APP: App recommendation, MUS: Music domain, QRY: Query recommendation, EC: E-Commerce domain, CS: Learning courses recommendations, AD: Advertisements, WWW: Web navigation recommendation, OTH: Others.

commerce scenarios and investigated, for example, in [45, 46] or [56]. Other problem settings in the e-commerce domain include next-basket recommendation [94, 110, 116] and the problem of *reminding* users, for example, in [122, 123] and [70].

Music recommendation is another "natural" application domain of sequence-aware recommenders. The consumption of music is often session-based and the listener's interest can change strongly from one session to another. The user experience can furthermore be influenced by the

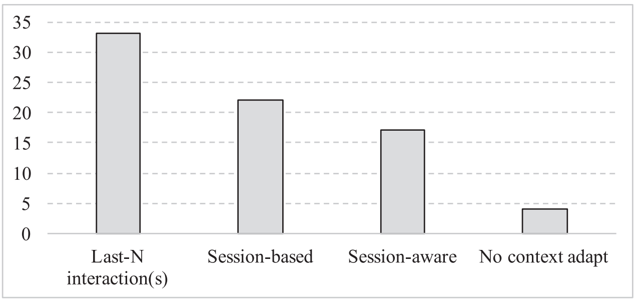

Fig. 2. Distribution of types of context adaptation in the analyzed research works.

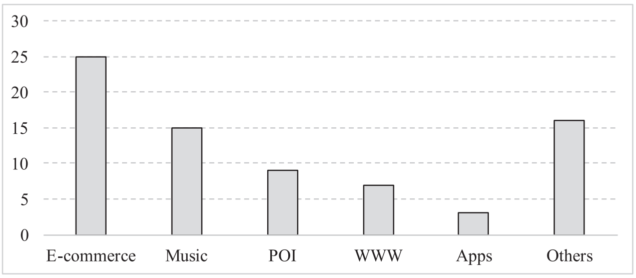

Fig. 3. Distribution of research works per application domain.

order in which the tracks are played, that is, weak ordering constraints can exist between the tracks. Such constraints can either be explicitly given by the user [89] or can be inferred from listening logs of a user community as done, for example, in [19] and [57]. Constraints can be inferred on the basis of the transitions between pairs of songs [19, 20], or based on metadata and other attributes of tracks that are usually "played together" [57, 108]. This knowledge can then be leveraged by sequence-aware recommenders to generate playlists or plausible continuations of listening sessions (see, e.g., [57] or [106]).

POI recommendations make predictions on the user's next location or make recommendations for the next place to visit. Moving between places is a sequential process, where the user's movements are usually limited by distance, time, or budget constraints. Sequence-aware recommenders have been applied in this context in different ways to predict the next user location (e.g., based on the user's current location [21, 43]). Considering several past user locations has shown to be helpful, for example, for travel planning [73], or when predicting which place the user will most probably visit at a specific time [77].

Web navigation prediction is an early application area of sequence-aware recommenders [83, 86, 125]. Web browsing is usually a sequential process and next-page visit predictions can help to recommend users interesting links that fit their current browsing session or to pre-load web pages.

App recommendation or app usage prediction is a more recent application field of sequence-aware recommenders, where considering the user's current context is crucial. A typical example of a research work is described in [9], where repeated usage patterns are mined from activity logs to pre-fetch applications or to make contextual suggestions on which app to use.

Others. Finally, there are a number of other application domains of sequence-aware recommenders which have, however, only been explored by a few research works. Examples include *advertisement* recommendation [121], news recommendation [32], job posting recommendation [91], the recommendation of videos in streaming platforms [46, 91], query recommendation [8, 18, 42, 102], or the recommendation of next activities to add during workflow modeling [59].

Table 2. Categorization of Works Regarding Repeated Recommendation and Trend Detection

Repeated Recommendation	
• Find items for repeated recommendation	[54, 70]
• Find timing for repeated recommendation	[76, 77, 109, 122, 123]
Trend detection	
• Detect individual trends	[85, 109]
• Detect community trend	[100, 109]
• Detect seasonal trends	[109]

3.5.4 Specific Tasks of ASequence-aware Recommendation. Table 2 lists works that considered the problems of repeated recommendations and trend detection.

Repeated recommendations, while useful in practical applications [70], were rarely the focus of researchers in the context of sequence-aware recommenders. Similarly, the detection of trends has also not been investigated to a large extent, even though a recent work suggests that taking, in particular, short-term trends can be important in the e-commerce domain [51]. A few works consider community trends and some investigate seasonal aspects. Interestingly, only two works analyzed here considered individual consumer trends and potentially outdated user tastes [85, 109].

Finally, there are only few works that explicitly mention that the goal is to recommend *similar* items [38, 57, 85] or *complements* [104]. A large number of papers in the literature focus explicitly on *next-item* recommendation [9, 42, 46, 78, 83, 86, 87, 99, 102, 121, 125] and *list continuation* [40, 57, 80, 117]. The problem of *next-basket* recommendation in e-commerce was also the focus of only a few works [72, 94, 110, 116].

4 ALGORITHMS FOR SEQUENCE-AWARE RECOMMENDER SYSTEMS

We can identify three main classes of algorithms in the literature that are used for the extraction of patterns from the sequential log of user actions: *sequence learning*, *sequence-aware matrix-factorization*, and *hybrids*. Each of these classes can be further decomposed into subcategories, leading to the *taxonomy of algorithms* in Table 3. A few comparably uncommon technical approaches are listed under the category "Others."

The majority of the reviewed works rely on some form of sequence-learning methods. Such approaches are a natural choice for the given problem, and frequent pattern mining techniques, for example, have been applied for a long time (e.g., for the prediction of user navigation patterns on websites [83, 86]). Other approaches, in particular the ones based on recurrent neural networks and distributed item representations, were only recently successfully transferred from domains like Natural Language Processing to sequential recommendation problems [37, 46]. We discuss the different approaches in more detail in the next sections.

4.1 Sequence Learning

Sequence-learning methods are useful in application domains where the data to be analyzed has an inherent sequential nature, like in Natural Language Processing, time-series prediction, DNA modeling, and, as the focus of this work, sequence-aware recommendation.

4.1.1 Frequent Pattern Mining.

Methods. Frequent Pattern Mining (FPM) techniques were originally developed to discover user consumption patterns within large transaction databases. Early Association Rule Mining approaches [5] focused on identifying items that frequently co-occur in the same transaction,

Table 3. Taxonomy of Algorithms for Sequence-Aware Recommender Systems

Class	**Sub-class**	**Family**	**Examples of research works**
Sequence learning	Frequent pattern Mining	Frequent itemsets	[48, 56, 83, 86, 106, 118]
		Frequent sequences	[78, 83, 86, 95, 115, 125]
	Sequence modeling	Markov models	[32, 42, 47, 81]
		Reinforcement Learning	[84, 97, 104]
		Recurrent neural networks	[29, 45, 46, 77, 98, 100, 102, 107, 116, 121]
	Distributed item representations	Latent Markov embeddings	[19, 20, 30, 111]
		Distributional embeddings	[9, 27, 37, 92, 103, 108, 124]
	Supervised models w/sliding window		[9, 110, 126]
Matrix factorization			[107, 117, 122, 123]
Hybrid methods	Factorized Markov chains		[21, 41, 43, 72, 94]
	LDA/Clustering w/ sequence learning		[40, 87, 99]
Others	Graph-based		[77, 105, 112, 120]
	Discrete optimization		[57, 73, 89, 113]

regardless of the order of their appearance. Later on, Sequential Pattern Mining [6] techniques were developed that considered item co-occurrences only as a pattern when the items appeared in the same order. An extreme case finally is Contiguous Sequential Patterns, which require that the co-occurring items are adjacent in the sequence of actions within a transaction.

Technically, in all approaches the patterns are mined in an offline process and usually translated into a set of association rules or another compact form of knowledge representation (see, e.g., [83, 115]). Usually, the resulting rules have values attached (e.g., *confidence* and *support*) that express their strength. In the prediction phase, we are given a partial transaction (e.g., an item that a user has recently bought) for which we seek additional items. These items are then determined by scanning the database for matching rules and by applying them on our partial transaction. In recommendation scenarios, the most simple approach is to determine only pairwise item co-occurrence frequencies in order to implement buying suggestions of the form "Customers who bought … also bought."

Application Examples. In one of the earlier works on sequence-aware recommenders [83, 86], researchers compared Association Rules (AR), Sequential Patterns (SP), and Contiguous Sequential Patterns (CSP) for a web usage mining scenario, where the problem is to predict a user's next navigation action for page prefetching or for *context-adaptation* in *session-based* scenarios. Technically, they used a fixed-sized sliding window of the current user session in the prediction phase. Given the last N user actions in this window, they look up and apply rules of size $N + 1$ and rank the recommendations (i.e., the elements of the rule consequents) based on the confidence value of the firing rules.[5] The obtained results showed that the less constrained patterns (AR and SP) lead

[5]A technically similar approach was later on proposed by Zhou et al. [125].

to better recommendations, whereas the usage of CSPs was more helpful for the page prefetching task. Other kinds of sequential patterns, such as closed patterns and negative patterns, were explored too. See, for example, the works of Zang et al. [118] and Hsueh et al. [48] for *context-adaptation* based solely on the last few user actions.

When using frequent pattern approaches, personalization is obtained by matching the activity of the user with the pre-extracted patterns. In a more recent work, Yap et al. [115] propose to further personalize the method and to weight the patterns according to their estimated relevance for the individual user. They designed three schemes to determine personalized pattern relevance scores and compared it to a popularity-based method. Their empirical evaluation indicates that applying personalized rule scoring schemes yields more accurate personalized *next-item recommendations* for the target users.

As an example of a comparably recent application domain, Lu et al. [78] present the MASP (*Mobile Application Sequential Patterns*) mining method for the problem of providing *context-adaptation* to smartphones by predicting the user's next used app. In their approach, transactions are composed of sequences of app usages which are annotated with the location of the user at each time. The MASP-mine algorithm takes into account both the user movements and app launches to discover the sequential patterns. At prediction time, the single pattern with the maximal support value that matches the recent user movements and activity is used to predict the app that is launched next.

Discussion. Frequent pattern mining techniques are well-explored and also easy to implement and interpret. The main drawbacks include the limited scalability of some of the approaches and, probably more importantly, the problem of finding suitable threshold values for the offline mining task. Usually, the main parameter is the minimum support value. If it is set too low, too many (often noisy) patterns are identified; if it is set too high, only rules for the most frequently occurring items will be found. To deal with the problem, one can start to search for the highest-quality rules in a database that was created with a fixed minimum-support threshold, and iteratively relax the quality constraints until a matching rule is found. Alternatively, different algorithm variants were proposed that use multiple minimum support values or "adjusted" confidence scores [95].

Another common challenge when using frequent pattern mining techniques is to decide between the different variants (AR, SP, CSP). Determining sequential patterns (SP, CSP) is often not only computationally more expensive given the more strict type of rules to be mined,[6] but it can also easily lead to much smaller rule bases. This in turn can result in situations at prediction time where no matching rule is found. Which method works best can furthermore depend on the application domain. The ordering of the events can be very important, for example, in the context of query or app recommendation [78, 125]. In other problem settings, for example, web page recommendation or next-track music recommendation, considering the item orderings might result in too small rule bases or have only small positive effects [14] that probably do not outweigh the additional computational complexity. Finally, in some domains, simple co-occurrence patterns were, despite their simplicity, employed with good success (e.g., in e-commerce and music recommendation [52, 56, 106]).

4.1.2 Sequence Modeling.

Methods. The input to sequence-aware recommenders, that is, the ordered and often timestamped log of past user actions, can be considered as a *time series* with discrete observations. Therefore, in many cases existing and sometimes complex time series prediction methods could,

[6]See [5, 6] for a detailed analysis on the computational complexity of FPM methods.

in principle, be applied. In RS applications, however, the timestamps are often merely used to sort the actions,[7] which allows us to apply "simpler" sequence models, that do not necessarily consider the complex underlying temporal dynamics of the observed sequences of actions.

In general, sequence modeling techniques aim to learn models from past observations to predict future ones, which are in our case user actions. Sequence modeling methods for sequence-aware recommendation mainly belong to three categories: Markov Models, Reinforcement Learning, and Recurrent Neural Networks (see Table 3).

— *Markov Models* consider sequential data as a stochastic process over discrete random variables (or states). The Markov property limits the dependencies of the process to a finite history. For example, in first-order Markov Chains (MCs) the transition probability of every state depends only on the previous state. Higher-order MCs use longer temporal dependencies to model more complex relationships between the states.[8] In sequence-aware recommender systems, the Markov property translates into assuming that the next user actions depend only on a limited number of the most recent preceding actions.
— *Reinforcement Learning* (RL) techniques learn by interacting with the environment, and are sequential in nature. In a recommendation scenario, the interaction consists of a recommendation of an item to the user (the *action*) for which the system then receives a feedback (the *reward*).[9] For instance, in the music domain, the system recommends a song and monitors if the user listens to or skips the recommended song. In this example, we assign a positive reward if the user listens to the song and zero otherwise. The problem is typically formulated as a Markov Decision Process (MDP) and the goal of the system is to maximize the cumulative reward computed over a number of interactions.
— *Recurrent Neural Networks (RNNs)* are distributed real-valued hidden state models with non-linear dynamics.[10] At each timestep, the hidden state of the RNN is computed from the current input in the sequence and the hidden state from the last step. The hidden state is then used to predict the probability of the next items in the sequence. The recurrent feedback mechanism memorizes the influence of each past data sample in the hidden state of the RNN, hence overcoming the fundamental limitation of MCs. RNNs are therefore well suited for modeling the complex dynamics in user action sequences. Variants of RNNs such as Long Short-Term Memory (LSTM) [36] and Gated Recurrent Unit (GRU) [23], by means of their sophisticated hidden dynamics, can model much longer and complex temporal dependencies than other approaches like Hidden Markov Models (HMMs) [34].

Application Examples (Markov Models). MCs in most cases cannot be naively applied to sequence-aware recommendation since data sparsity quickly leads to poor estimates of the transition matrices. Shani et al. [97] therefore enhance their MC-based approach with several heuristics—namely, skipping, clustering, and finite mixture modeling—to mitigate the impacts of data sparsity. In their application, the input data consists of purchase records of an online book store and their experiments show the superiority of sequential models over non-sequential ones in predicting what books to recommend *next* given the sequence of the last few user interactions. In a different application domain, McFee and Lanckriet [81] use MC *mixtures* for the problem of music playlist generation, a typical *list continuation* problem, where a mixture is a weighted ensemble of several MCs

[7]The work of Du et al. [29] represents an exception in that respect.
[8]See [31] for details on Markov Models for pattern recognition.
[9]See [65] for a comprehensive review on Reinforcement Learning.
[10]See [74] for an overview of Recurrent Neural Networks.

(uniform, weighted, and k-nearest neighbors) whose weights are learned via maximum-likelihood optimization on a training dataset of playlists.

Another challenge when applying MCs usually lies in the choice of the order of the model. In the context of a *session-based* next-query recommendation problem, He et al. [42] therefore use a mixture of Variable-order Markov Models (VMMs, sometimes called *context trees*[11]), which use a context-dependent order to capture both large and small Markov dependencies. A different approach is adopted by Garcin et al. [32] in the context of a news recommendation application. They assign one predictor (expert) to each context (node) in the context-tree, where each node is associated with a different order of the Markov model. Each predictor is then trained to predict the article to suggest *next* given the last sequence of user actions. As the sequence of actions of the user grows, deeper nodes in the tree become active and contribute to the final recommendations.

Hidden-state models, in particular HMMs, address some of the limitations of MC models and are, for example, applied in [47] for the *contextual next-track* music recommendation problem. In HMMs, each hidden (or latent) state is a discrete variable associated with a probability distribution over the observed variables. In conformity with the Markov property, every hidden state is conditionally dependent only on the previous one [34]. There already exist a few applications of HMMs to time-aware collaborative filtering [96, 119]. In [47], the hidden states is used to model the (unobserved) context of the user. Discrete-valued hidden states are, however, limited in terms of the contextual information they can store, which to some extent limits their applicability for sequence-aware recommendation.

Application Examples (Reinforcement Learning). Reinforcement learning based on MDPs are used in [97] and [84] for sequence-aware recommendation in online shops. These approaches make it possible to tailor recommendations not only to the recent user activity but also to the expected reward (income) for the shop. Since the state space of MDPs can quickly become unmanageable in realistic scenarios, Tavakol and Brefeld [104] factorize the space over a set of mutually independent item attributes. In their application case of a clothing marketplace, dress characteristics such as category, color, or price can be considered; the sequential relationships between attributes are then independently modeled by an MDP to predict the characteristics of the products the user will likely search or buy next.

Application Examples (RNNs). Zhang et al. [121] use RNNs for click prediction for online advertisements. At each timestep, they train the RNN to predict the next click of the users given their last click and the previous state of the network using a classification loss measure (cross-entropy).

For the domains of e-commerce and media recommendation, Hidasi et al. [45] explore the use of GRUs [23]—a variant of RNNs—for modeling user activity in a session-based scenario. Technically, as in [121], the model is trained to predict the next item in a sequence given the current one, but different loss functions were used, which at the end led to better performance. Later on, Hidasi et al. [46] included item features into the sequence model. The proposed parallel-RNNs model is able to capture the interdependencies between item identifiers and their features, leading to a further improved recommendation accuracy. Quadrana et al. [91] employ *hierarchical* RNNs for modeling user preferences across sessions. The first level of the hierarchy, which represents the state of the user across sessions, is used to initialize and propagate information to the second level of the hierarchy, which is used to generate recommendations within a session. Transferring the information from prior sessions ultimately led to better recommendations. Yu et al. [116] finally employ RNNs for *next-basket* recommendation in e-commerce. In their case, the RNN is fed

[11]See [11] for a comprehensive analysis of algorithms for learning VMMs.

with real-valued representations of the baskets as an input—that is, the sets of items the user has interacted with at each step of the sequence—and trained to rank the items in the next basket.

A number of further technical variants of RNNs and problem encodings were recently proposed for different application domains like e-commerce, online ads, and query or POI recommendation. Twardowski [107], for example, addresses the problem of providing personalized recommendations to *anonymous users* by using RNNs to generate a compact representation of the user's interactional context. Such representation is later combined with item representations to generate *contextual* recommendations. Liu et al. [77] aim at detecting *interest trends* of individual users over time in an RNN-based POI recommender system. Both Sordoni et al. [102] and Soh et al. [98] address the *context adaptation* problem. Sordoni et al. [102] propose a generative model for session-based query recommendation based on Hierarchical Recurrent Encoder-Decoder neural networks. The higher level of the hiercarchy encodes the latent intent of the user in the current search session, while the lower level generates next-query suggestions tailored to this intent. Soh et al. [98] employ RNNs to represent sequences of user interactions to personalize user interfaces.

Song et al. [100] propose a *context adaptation* approach to build a session-aware recommender that can handle long-term (e.g., seasonal) and short-term changes in user preferences in the news domain. In their work, the authors propose the Temporal Deep Semantic Structured Model to combine user and item features with user temporal features into a joint model. Static features are modeled by several feed-forward neural networks, whereas the temporal features are modeled by a set of RNNs.

Du et al. [29] address the *list continuation* problem in a wide range of application domains, from taxi drop-off to financial transactions. Given data with timestamps for the actions, their model combines RNNs with Marked Temporal Point Processes, a mathematical tool that models the distance between subsequent events as a random process. The idea of modeling the distance between subsequent events as a random process is used also by Wang and Zhang [109], who employ a hierarchical Bayesian framework based on hazard models to represent the probability of purchasing a product at a given time, while modeling time-dependent patterns between follow-up purchases at the same time. Both models allow one to compute the utility of recommendations as a function of the sequence of past user actions and of the recommendation timing.

Discussion. Sequence modeling approaches have been successfully applied for a variety of sequence-aware recommendation problems. Our review, however, shows that Markov Models in most cases cannot be directly applied in a naive manner for sequence-aware recommendation problems, due to problems of data sparsity and computational complexity, which is why researchers often rely on specific model variants or embed heuristics into the learning process. Still, it is not always fully clear if such model variants scale to real-world problems.

Deep learning–based techniques, on the other hand, have been increasingly explored in the past few years. Among the factors that contributed to the recent revival of neural networks are the availability of large datasets for training and the increased computational capacities of modern computer hardware.

The computational demands of deep learning approaches can, however, still represent a barrier to their practical use, for example, because of the need for testing and optimizing different (hyper-)parameter configurations. Despite these limitations, researchers should not stop exploring more advanced algorithmic approaches in the future, given the sometimes low-quality impression of today's recommender systems in practice [61] and the fact that some of today's first deep learning–based approaches are sometimes not yet much better than computationally more simple approaches [52]. More research in this area is also particularly important as naive methods can have certain limitations like a bias to recommend mostly popular items, which are not captured by today's problem formulations and common performance measures like precision and recall.

4.1.3 *Distributed Item Representations.*

Methods. Distributed item representations are dense, lower-dimensional representations of the items. The representations are derived from sequences of events and preserve the sequential relationships between the items. Similar to latent factor models, every item is associated with a real-valued *embedding* vector, which represents its projection into a lower-dimensional space in which certain item transition properties are preserved. For example, some representations are based on the co-occurrence of items in similar contexts [37, 108]. Others translate pairwise transition probabilities into distances in a Euclidean space [19, 20]. In sequence-aware recommendation problems, we can recommend the next item(s) given the user's most recent actions by traversing the embedding space in a stochastic fashion [19] or by searching the nearest neighbors to the last item(s) that were explored by the user [37].

Application Examples. Distributed Item Representations have been explored, for example, in the domain of playlist generation. Zheleva et al. [124] build a session-level long-term Latent Dirichlet Allocation (LDA) model based on the community of users and merges it with a short-term LDA model based on the current user session. Chen et al. [19] propose the Latent Markov Embedding (LME) approach, a regularized maximum-likelihood embedding of MCs in the Euclidean space. In their method, every item is projected into a space in a way that the distance between any pair of items in this space is proportional to their transition probability in a first-order MC. The learning procedure directly exploits the *weak ordering* between tracks in playlists. The resulting vector spaces can then be traversed stochastically to generate *new* sequences (in their case, playlists) or *continuations* of existing ones. Later on, Chen et al. [20] extend LME by clustering items and by adding cluster-level embeddings to account for locality in item transitions. Wu et al. [111], on the other hand, propose a Personalized LME version where both items and users are projected into a Euclidean space in a way that the strength of their relationship is reflected by the projection. This allows them to incorporate long-term user preferences into the model and to use them for recommendation. More recently, Reddy et al. [92] employ a similar approach for the recommendation of learning courses. Students, lessons, and assessments are embedded in a common latent skill space, in which the evolution of the student knowledge is formalized by a HMM. The goal of Feng et al. [30] is personalized POI recommendation based on the *last-N* points of interest visited by the user. In their approach, they use a pairwise ranking function similar to Bayesian Personalized Ranking (BPR) [93] to condition the transition probabilities on the personalized ranking preferences of the users.

A different technical approach—in their case to deliver personalized product ads—was chosen by Grbovic et al. [37]. Their work is based on leveraging the so-called "distributional hypothesis," which in the domain of linguistics states that semantically equivalent words occur frequently in the same contexts,[12] for sequence-aware recommendation. The proposed Prod2Vec recommendation method learns distributed item representations from sequences of emails containing purchase receipts. Similar to the LME method, every item is projected into a lower-dimensional space. Specifically, Prod2Vec uses the skip-gram model, which projects items that tend to have similar neighboring items (i.e., items that tend to be "surrounded" by the same set of items) close to each other. Given the sequence of the last few observations for a user, the most similar items in the lower-dimensional space represent the recommendation candidates. Different enhancements of the recommendation scheme of Prod2Vec were proposed, including the usage of time decay factors for older email receipts or the consideration of directed order dependencies between the inter-

[12]The distributional hypothesis forms the basis of the recent Word2Vec [82] and GloVe [90] approaches for Natural Language Processing.

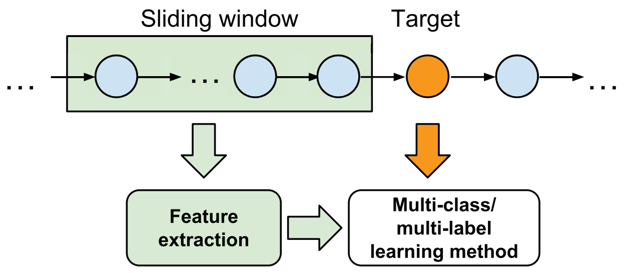

Fig. 4. Sequence-aware recommendation as supervised learning with sliding windows.

actions [27]. Furthermore, Vasile et al. [108] propose Meta-Prod2Vec, an enhanced version of the skip-gram model that conditions the item embeddings also on their metadata. Greenstein-Messica et al. [38] employ Word2Vec and Glove to create item embeddings based on click sequences and item metadata, and use these embeddings to enhance session-based recommendation with RNNs.

A technically different approach to use distributed item representations to learn a model from session-based user data is proposed by Tagami et al. [103]. The authors use the Paragraph Vector (PV-DV) model [69], which learns an additional user vector along with the item representations. In this approach, the user vectors computed from 1 day of interaction data are used as features in a multi-label classification problem to predict the set of ads the user will click on the next day.

Discussion. Distributed representations were successfully applied, in particular, in the domain of Natural Language Processing [82] and, in the context of recommender systems, as an alternative method for representing textual data in content-based approaches [7]. While the applications discussed in this section indicate that using such representations can be helpful for sequence-aware recommendation scenarios as well, some embedding approaches can be computationally demanding and sometimes require extensive parameter tuning to yield good results. Similar to sequence modeling methods, approaches based on distributed representations require a substantial amount of training data to be effective.

While embedding methods exploit the sequential relations between items to learn the item representations, these methods do not make direct use of the sequence of recent user interactions to generate the recommendations. They instead rely on approximate methods like time-decay nearest neighbors [37] or on additional supervised learning layers [9, 103] to predict the next interactions of the user. This in turn can lead to limited effectiveness in domains with strict ordering constraints for which sequence modeling methods can be preferable. Using an indirect approach can also further increase the computational costs of these methods.

For some works, the significance and practicability of some of the proposals in real-world environments is not always fully clear. The playlist generation method of [19], for example, is not only computationally very intensive, it was also evaluated with a specific measure (the average log likelihood), which does not truly inform us about the quality of the resulting playlists [15].

4.1.4 *Supervised Learning with Sliding Windows.*

Methods. Sliding window models convert the next-in-sequence prediction problem into a traditional supervised learning problem that can be solved with any classifier such as decision trees, feed-forward neural networks, and learning-to-rank methods. The general idea of the approach, which resembles autoregressive models, is as follows. A sliding window of size W is moved over each sequence (see Figure 4). At each step, all items within the window are used to derive the feature values of the supervised learning problem and the identifier of the immediately next item is used as a target variable. As a result, the sequence prediction problem is turned into a multi-class classification problem, or into a multi-label classification in case multiple target items are allowed.

Application Examples. In an early approach of that type, Zimdars et al. [126] frame sequential click prediction in a web usage mining scenario as a binary prediction problem. In their approach, clickstream data is first expanded by defining a set of "accumulator" variables (*lagged* and *cache* variables) to represent the contextual and historical activity of the user. Then, a page-level probabilistic decision-tree model is trained and at recommendation time the pages are ranked according to the predicted probability of being the next page.

More recently, Baeza-Yates et al. [9] use contextual features that were extracted from the last 12 hours of the app usage log to predict which (mobile) app will be used *next* in a *session-aware* scenario. Their contextual features include some *basic* features, such as the geolocation and phone usage characteristics, as well as *session* features, which are basically the Word2Vec representations [82] of the actions of the user in the sliding window. Finally, they use a parallelized version of the Tree Augmented Naive Bayes algorithm for the classification (prediction) of the next used app.

In the e-commerce domain, Wang et al. [110] use feed-forward neural networks for *next-basket recommendation.* For the predictions, only the items in the previous basket were used to rank the items for the next one (i.e., the window size is 1). The first layer of the neural network encodes the previous basket as a fixed-size real-valued feature vector that is computed by taking the element-wise maximum (*max-pooling*) or average (*mean-pooling*) of the embeddings of the items of the basket. The second and final layer of the network learns to rank the items in the next basket by taking as input the concatenation of the user's current and previous basket's embeddings.

Discussion. The main advantage of this class of approaches is that it is general, conceptually simple, and that a variety of existing classification methods and libraries can in many cases be applied off-the-shelf. There are, however, also a number of shortcomings. First, as in many classification problems, the effectiveness of the approach depends on the quality of the feature engineering phase. Finding the best features can be challenging and often the features are domain-dependent and cannot be re-used across domains. Second, the obtained results can be highly sensitive on the choice of the window size. Finally, setting up a multi-class (or multi-label) classification problem can be computationally expensive when the set of items grows. Furthermore, the quality of the learned model can be strongly affected by unbalanced distributions of the target variable, which is a common situation in real-life recommendation scenarios.

4.2 Sequence-Aware Matrix Factorization

Method. Sequence-aware recommendation is different in various ways from the traditional matrix-completion problem formulation. However, there are a few cases where sequence information, which is usually derived from timestamps, is also considered within algorithms that are designed for matrix completion.[13]

Application Examples. Zhao et al. [122, 123] focus on an e-commerce recommendation scenario, where the problem is to fill the missing cells of a given user-item purchase matrix. Specifically, they study the problem of finding the *timing for repeated recommendations.* Their proposed method factorizes the matrix using a weighted loss function that maximizes the expected utility for the user. The novel aspect of their approach is that the utility is determined based on the observed time intervals between purchases of each pair of items. The resulting model can predict the personalized

[13]In this section, we focus on *sequence-aware* and time-interval-based algorithms and do not discuss general *time-aware* collaborative filtering techniques as described, for example, in [17] and [68].

relevance of an item depending on the time at which the recommendations are generated and thereby takes the available sequence information into account.

Yu and Riedl [117] analyze the problem of generating interactive personalized story *continuations*. Stories are represented in a prefix graph, in which each node represents a prefix of a story, that is, a possible sequence of point plots. In their problem encoding, they are given a "prefix-rating matrix" to be completed, where the items are replaced by possible story prefixes and users provide ratings only to some of the prefixes. Matrix factorization is used to predict the missing entries. The highest scoring full story that descends from the current "story-so-far" of the user is used to suggest the next plot point. The rating of the user is collected on the next plot point, and the process (factorization, recommendation, rating) continues iteratively until the story reaches an end.

Finally, Twardowski [107] not only proposes an RNN-based method as discussed above but also a matrix factorization approach to *session-based recommendation* for the next-in-sequence recommendation problem in e-commerce. In this approach, session events such as click, add-to-cart, and so forth, as well as the items are encoded by separate latent feature vectors. Sessions are then represented by the time-decayed sum of event vectors associated with the actions performed by the user so far in the current session. The prediction of the next items is then based on the factorization of the observed session-item tuples, which can be obtained with standard ranking loss functions.

Discussion. The advantage of the discussed approaches is that standard matrix factorization algorithms from the literature can often be applied. A main challenge, however, lies in the definition of a suitable and computationally feasible encoding of the given application problem. Considering, for example, the proposal in [117], one not only has to collect user feedback at each step of the sequence, but the given encoding can easily lead to a huge matrix due to the combinatorial explosion of the possible sequences. It also requires continuous updates of the factorization model as users have to provide new ratings for every suggested next step in the sequence.

The approaches discussed in this section can be seen as special cases of matrix factorization algorithms for time-aware recommendation (e.g., [68]). These algorithms mostly focus on tracking changes of user behavior over large time spans, usually through time-evolving user and item latent factors. However, such methods, as discussed in more detail on [17], can however often not be updated in real-time. In some application domains they should therefore be extended or combined with methods that support the short-term adaptation according to the user's current preferences [56].

4.3 Hybrid Methods

Methods. Hybrid models often combine the flexibility of sequence-learning methods with the robustness to data sparsity of factorization-based matrix-completion techniques. Furthermore, such forms of hybridization enable sequence-learning methods to use the power of modern collaborative learning-to-rank models such as BPR [93], which usually cannot be easily embedded in standard sequence-learning approaches.[14]

Application Examples. Among the first proposals for such a hybrid technique is the Factorized Personalized Markov Chain (FPMC) method of Rendle et al. [94]. The method combines matrix factorization with MCs for the problem of *next-item recommendation* given the *last-N interactions* of the user, for example, in e-commerce settings. In the case of first-order MCs, user interactions can be represented as a three-dimensional (*user*, *current item*, *next item*) tensor. Each entry in the tensor corresponds to an observed transition between two items performed by a specific user.

[14]The works presented in [45] and [46] represent exceptions that combine RNNs with BPR's ranking loss criterion.

The proposed method then uses pairwise factorization to predict the unobserved entries in the sparse tensor, that is, to predict personalized transitions between pairs of items. Overall, FPMC can be seen as a first-order MC whose transition matrix is jointly factorized with a standard two-dimensional user-item matrix factorization approach. This joint factorization at the end makes it possible to infer the unobserved transitions in the MC from the transition pairs of other users. At recommendation time, the items are ranked according to their likelihood to be the next item given the last item the user has interacted with.

In addition to e-commerce settings, the FPMC method has been successfully applied to other problem settings, for example, for POI recommendation or check-in prediction in location-based social networks [41, 72]. He and McAuley [43] present a variation of the FPMC method where the matrix factorization technique is replaced by a Factored Item Similarity Model (FISM) [64]. The resulting method, named Factorized Sequential Prediction with Item Similarity Model (Fossil), models users as a combination of the factors of the items they have interacted with, which in turn allows Fossil to provide sequential recommendations also for cold-start users—the FPMC method is limited in that respect since it relies on user-item matrix factorization—as long as the item representations can be estimated accurately. The method was tested in different application domains such as clothes, toy, or electronic devices recommendation.

A number of other hybrid approaches were proposed in the literature. In the context of playlist recommendation, Hariri et al. [40] use LDA [13] to extract latent topics from playlists, where playlists are taken as documents and tracks as words for the LDA step. A sequential pattern mining technique is then applied to find patterns of such latent topics in the playlists. When generating *continuations* from existing sessions or playlists, the frequent patterns are mapped to the topics extracted from the current listening session and used to filter the recommendations that are generated by a classical nearest-neighbor-based recommender.

Song et al. [99] propose the States Transition Pairwise Ranking Model, which combines LDA with first-order MCs to simultaneously model the user's long- and short-term favorites. The user's long-term favorites, for example, in the movie domain, are determined by the topics generated in the LDA step with a user-specific prior. The short-term favorites are captured by an MC transition matrix. The combined model is then basically a HMM whose latent states are controlled by the personalized LDA generative model, which can be trained via Markov Chain Monte Carlo (MCMC) Bayesian inference and then be used to predict the *next-item* given the last few interactions of the user.

Finally, in the context of next-app recommendation, Natarajan et al. [87] propose a method based on behavioral clustering to introduce personalization into *session-aware* models. In the behavioral clustering step, users with similar sequential behavior are grouped by applying *k-means* clustering on the per-user transition matrices of a first-order MC. A personalized PageRank algorithm is then used to build transition models at the cluster level. At recommendation time, the user is mapped to its corresponding cluster and the cluster-level transition model is used to provide sequential recommendations.

Discussion. Hybrid approaches are often used to build recommendation systems due to their capability of overcoming the shortcomings of individual methods, e.g., in terms of limited content discovery support or in the context of user- or item cold start situations [16]. The typical challenges when designing hybrid recommender systems include the problem of finding the best way of combining the predictions of the different recommendation channels and of determining importance weights for each individual method in the final prediction.

Some of the discussed approaches circumvent these issues by combining different competing methods in a unified model, like the combinations of MCs with matrix factorization [94] or with

LDA [99], in a way that the final recommendations are computed by a single algorithm. Other approaches rely on a cascade of algorithms [40], where a sequential model is used to filter the predictions generated by a sequence-agnostic one, or a meta-level approach [87], where a first sequence-aware model is used as the basis for deriving another one.

4.4 Other Methods

Only a handful of the methods proposed in the literature do not fit into the above categories, and most of them are either *graph-based* or based on *discrete optimization* techniques.

Graph-Based Methods. Xiang et al. [112] focus on *context adaption* to detect short-term user intents and interests. The paper describes a graph-based approach which is evaluated based on implicit feedback datasets in the area of social bookmarking. In their approach, long-term and short-term user preferences are fused into a two-sided bipartite graph, the Session-Based Temporal Graph. One side of the graph connects users with the items they have interacted with in the past. The other side of the graph connects session identifiers with the items the user has interacted with in the current session. The edges are weighted to balance the influence of long-term and short-term preferences. The relationships between the items are propagated through the graph via Injected Preference Fusion, and at recommendation time the graph is traversed via a random walk to generate session-aware recommendations.

Liu et al. [77] aim to detect *interest trends* of individual users over time for POI recommendation. Their method integrates static user interests and evolving sequential preferences based on temporal interval assessments. Similar to [122], the time intervals between POI visits are assessed from user check-in sequences in a POI-to-POI transition matrix. Then, a bi-weighted low-rank graph is constructed to learn the individual user's behavioral preferences by identifying a set of common graph bases. As a result, the static interests and evolving sequential preferences of the user are learned simultaneously with the graph. As in [122], recommendations are generated by ranking the POIs by their expected relevance for a given time period.

In the same application domain, Zhang et al. [120] try to detect if there are *order constraints* when visiting POIs. Their method, named LORE, mines sequential patterns from location sequences and represents the patterns as a location-location transition graph. The probability of a user visiting a new location is modeled through an additive n-th order MC, in order to incorporate sequential dependencies between locations.

Finally, Trevisiol et al. [105] apply *contextual adaptation* to address the *new user* problem in news recommendation. In their approach, the users' browsing behavior is represented by two graphs: the BrowseGraph, which collects the behavior of all user sessions, and the ReferrerGraph, a subgraph of the BrowseGraph which is induced by browsing sessions of users coming from the same referrer domain.[15] When the user enters the news portal from the referrer domain, the recommendation candidates are chosen among the neighbors of the article in the Referrer Graph, using the Browse Graph as fallback. The article's neighbors are ranked according to various strategies, such as by random, by popularity, by content similarit, and by edge weight.

Discrete Optimization Methods. Discrete optimization is often employed for sequence-aware recommendation problems when weak or strict ordering constraints between items exist. Typical application examples include travel planning, learning course sequence generation, and playlist generation.

Both Jannach et al. [57] and Pauws et al. [89] address the *list continuation* problem to generate playlists in the music domain. Jannach et al. [57] propose a two-stage method to determine suitable

[15]Examples of referrer domains considered in the paper are *Facebook*, *Twitter*, and *Reddit*.

continuations to a music listening session. In a first step, the recommendable tracks are scored based on a variety of features (including track co-occurrences and musical characteristics). Then, the first elements of the resulting list are re-ranked in a greedy approach in order to optimize the coherence of the track continuations with the recently listened tracks. Pauws et al. [89] include explicit *order constraints*, either global to all users or local to a single user, in their model. The authors first formulate the problem as an integer linear programming (ILP) problem, which is NP-hard in its solution, and then implement a local search heuristic to make the solution scalable.

Similar to [89], Xu et al. [113] address the *list continuation* problem with explicit *order constraints*, but in a different application domain: personalized course sequence recommendation. The specific goal is to find a sequence of courses that (a) will minimize the time-to-graduation of the students and maximizes their GPA, (b) at the same time matches the interests of the user, and (c) respects the sequential constraints between the courses (called the prerequisite graph). Similar to [57], they propose a two-stage algorithm. First, candidate sequences with short time-to-graduation are extracted from the prerequisite graph using a Forward-Search Backward-Induction algorithm. Then, an online regret minimization algorithm based on multi-armed bandits is used to select course sequences that are expected to maximize the students' GPAs.

Generally, most discrete optimization methods for sequence-aware recommendation do not rely on exact or exhaustive search. Instead, due to the computational complexity of the underlying problems, they usually resort to heuristic search or greedy optimization techniques (see also [63]).

4.5 Summary and Pros and Cons of Selected Approaches

The main ideas of the most important families of algorithms along with a selection of their most typical advantages and disadvantages are summarized in Table 4. Note that the entries in the table mainly serve as a rough orientation and that specific pros and cons for individual algorithms within each family can exist.

5 EVALUATION OF SEQUENCE-AWARE RECOMMENDER SYSTEMS

5.1 Common Evaluation Approaches for Recommender Systems

In academic environments, the evaluation of recommender systems is dominated by "offline" (simulation-based) experiments on historical rating or implicit feedback datasets. The common offline evaluation methodology when using the matrix completion abstraction is to split the given preference data into training and test splits, use the training data to learn a model and predict the held-out preferences based on this model. The quality of the outputs of an algorithm can then be assessed with the help of measures such as Root Mean Square Error (RMSE), or Precision and Recall. In addition, one can analyze a number of other quality factors of the recommendations, for example, in terms of the diversity of the list, the general popularity of the recommended items, or the algorithm's catalog coverage.

User (laboratory) studies are an alternative to offline experiments. Such studies are often used to assess the potential impact of recommendations on the behavioral intentions of users, an aspect which cannot be determined based on simulation studies. Finally, field studies (e.g., in the form of A/B tests) are used to analyze the effects of recommenders on their users in real-world environments. This latter form of evaluation is, however, comparably rare in academic environments.

5.2 Offline Evaluation of Sequence-Aware Recommenders

While in principle all three mentioned evaluation forms (offline studies, user studies, field tests) can be applied for sequence-aware recommenders, slightly different evaluation methodologies are used in offline studies. Generally, several aspects of the evaluation protocols used for sequence-

Table 4. Summary of the Main Ideas, Pros, and Cons of the Algorithms for Sequence-Aware Recommendation

Algorithm	**Main idea**	**Pros**	**Cons**
FPM	Discover patterns in user action sequences	• Easy implementation • Explainable results	• Complex configuration • Suffers from data sparsity • Limited scalability
MC	Compute transition probabilities over fixed-length sequences	• Explainable results	• Fixed transition order • Suffers from data sparsity • Limited scalability
VMM	Compute transition probabilities over variable-length sequences	• Variable transition orders • Explainable results	• Suffers from data sparsity
HMM	Model the causal factors in user sequences as transitions between *discrete* hidden states	• Learns from variable-length • inputs Robust to data sparsity	• Limited explainability • Huge number of discrete parameters
RL	Directly maximize the customer and seller reward over time	• Dynamically adapt recommendations to future (unknown) rewards • Under active research	• MDP-based approaches have same issues as MCs • Limited explainability
RNN	Model the causal factors in user sequences with *non-linear* transitions between *continuous* hidden states	• Learns from variable-length inputs • Learns long-term dependencies • Robust to data sparsity • Compact hidden states • Under active research	• Complex configuration • Limited explainability • Benefits not fully clear in some domains
EMB	Embed items into latent spaces that preserves sequential transition properties	• Robust to data sparsity • Visually interpetable embeddings • Under active research	• Need auxiliary methods to make recommendations • Limited explainability
SL	Use supervised learning over features extracted from fixed-size sliding windows over sequences	• Easy implementation • Use off-the-shelf supervised algorithms	• Explainability depends on the chosen supervised method • Feature engineering
MF	Define new inputs and loss functions for MF to handle sequences	• Extensive literature available • Robust to data sparsity	• Non-trivial input and loss design • Concerns regarding scalability

Algorithm: FPM: Frequent Pattern Mining, MC: Markov Chains, VMM: Variable-order Markov Models, HMM: Hidden Markov Models, RL: Reinforcement Learning, RNN: Recurrent Neural Networks, EMB: Distributed Item Representations, SL: Supervised Learning w/ Sliding Windows, MF: Matrix Factorization.

aware recommenders are closely related to those that are used for Time-Aware Recommender Systems (TARS), for which Campos et al. [17] provide a formalization and a detailed analysis. In our subsequent analysis we will therefore take this framework as a reference.

5.2.1 *Evaluation Methodologies.*

Campos et al. [17] discuss evaluation methodologies for TARS along three dimensions: data partitioning, definition of the target items, and cross-validation.

Dataset Partitioning. In standard matrix completion setups, the partitioning of the ratings into training and test sets is often done by sampling ratings randomly. Common variants are to either take samples from the entire dataset or sample ratings per user. For TARS, one can additionally consider a *rating order* criterion and, for example, select the most recent ratings (of the individual user or the community) for the test set. For sequence-aware recommenders, the rating order criterion, that is, in our case the sequence of actions, is already given by definition.

Two forms of splitting the data into training and test set are possible: *event-level* and *session-level* splitting. When splitting the data at the *event-level* we can assign all events before a certain point in time to the training set and the remaining events to the test set, as done in [37]. With this procedure, an event is therefore assigned to one of the sets independent of the user or the session it might belong to. Similarly, one can apply a time-based splitting criterion at the *session-level*, that is, we do not split up sessions but consider the timestamp of the first event of a session to decide if the session goes in the training or test set [45, 106].

Both event-level and session-level partitioning can be applied to either the whole set of users (*community-level* partitioning) or to a subset of the users (*user-level* partitioning). In the latter case, one can select a number of *training users* and put all their data into the training set. Interactions of the remaining *test users* are further split—either at event or session-level—into a *user profile* and *test data* [56]. The user profile includes the less recent interactions and is used as input to the recommender; the test data contains the most recent interactions to be predicted. Finally, mixed approaches are, in principle, possible as well. One can, for example, select a number of users as test users and put all data into the training set except for the most recent sessions of the test users [91].

Besides the question of the splitting criterion, different options exist of how to apply *size conditions* [17]. Placing 80% into the training set and 20% in the test set is a common approach. In sequence-aware recommenders, additional options exist. One can use *time-based* splits and place events (or sessions) prior to a give time into the training set and use the rest for testing [30]. Or, we can use *fixed-size* splits and place the last k events of each sequence into the test set and the remaining into the training set [21].

Our survey shows that no common standard exists in the community regarding data partitioning procedures. In the context of session-based or session-aware recommendation problems, it is, however, advisable to use a session-level partitioning procedure to avoid that individual user sessions are split up. This is, in particular, necessary to mimic the behavior of session-based and session-aware recommender systems that are batch-trained on instances of past sessions for efficiency reasons. In the specific case of session-aware recommendation problems, community-level partitioning has to be applied to ensure that a recommender can also learn longer-term models for individual users. In the case of session-based recommender systems, user-level partitioning has to be preferred to ensure that a recommender is able to provide recommendations to new users (i.e., users with no records of past interactions with the system).

Also with respect to the size conditions, no community standards exist so far. In some works, for example in [45], all sessions except those of the last day are placed into the training set. While the maximum amount of data is provided for learning in such a setting, only one single training-test split can be used. The size of the training data can also be chosen based on domain-specific requirements. In domains such as news recommendation or e-commerce, focusing on the most recent interactions in some cases is sufficient or even advisable, for example, because news can quickly become outdated or because e-commerce shoppers might concentrate on trending or recently added items [52]. In general, when using time-based splitting, the evaluation should be performed with different time splits, which allows us to evaluate the learning rate of the algorithm (i.e., the minimum amount of training data that provide stable recommendations).

Definition of the Target Items. In traditional evaluation setups, we aim to predict ratings for a set of target items or search for an optimal ranking of these items. In sequence-aware recommenders, we are usually interested in predicting future *actions* of a user (or, generally, events), where actions as described above usually have an associated type and item. In addition, the input to the evaluation step is not limited to a user or user-item pair for which a ranked list or rating prediction is sought for, but the input can also include a sequence of events, for example, an entire session from the test set in session-based or session-aware recommendation scenarios. If such a sequence of events is given, the general idea in most evaluation approaches is to hide a subset or all of the events of the given sequence and predict the user's (next) actions. Different variations of the evaluation scheme can be found in the literature.

— *Sequence-agnostic prediction.* In this case, the order of the hidden actions is not relevant and a recommendation is considered successful if it correctly predicts one or more of the hidden actions [122, 123].

— *Given-N next-item prediction.* In this protocol variant, the first N elements of the sequence are "revealed" to the algorithm and the task of the recommender is to predict the immediate next action. Jannach et al. [56] propose to use one specific value for N when evaluating. In the domain of next-track music recommendation, often only the last element of a sequence is hidden (i.e., $N = 1$) [40]. In [45, 46, 91], an approach is taken where N is incrementally increased.
— *Given-N next-item prediction with look-ahead.* This is a combination of the above protocols and used in [37, 45]: the set of revealed events is continuously increased. The order of the hidden actions (which form the look-ahead set) is not relevant.
— *Given-N next-item prediction with look-back.* In addition to the first N items of the current sequence, the idea of this protocol is to also reveal a set of actions that happened *before* the current sequence [56, 70, 91].

Besides the different ways of evaluating the predictions for a given sequence or user session, one can limit the evaluation to actions of certain types, for example, to purchase actions, as done in [56] and in the context of the 2015 ACM RecSys challenge.[16]

The choice of the best evaluation protocol depends both on the specific research question and on the application domain. When evaluating session-based recommender systems, *given-N next-item prediction* (with or without look-ahead) usually is preferred. In case the goal is to assess how quickly a system is able to adapt its recommendations to the user's assumed short-term intents, one might use a *given-N* protocol where N is incrementally increased. Hiding only the last element of a sequence can be done in different application scenarios to assess the recommendation performance when more information is available ("warm start"). Finally, when the goal is to evaluate the value of combining short-term and long-term user preferences or the effectiveness of different *reminding* strategies, the *given-N next-item prediction with look-back* protocol can represent an appropriate choice.

Cross-Validation. Using a randomized cross-validation procedure is not possible in all application areas of sequence-aware recommenders. In the case of next-track music recommendations, where the goal is to predict the next track within a given playlist [40, 57], the given sets of playlists can be randomly split into training and test sets as long as the creation time of the playlists is not relevant.

In many other cases, however, sequence or time dependencies exist, which make it impossible to randomly assign events to the training and test splits. Some works like [45] therefore only use one single training-test split, which can, however, lead to biased results. Therefore, alternative validation approaches like repeated random subsampling procedures should be applied. One can, for example, split the data into several, potentially overlapping time slices of about the same length ("sliding window") and use the events at the end of these time slices as test sets [52, 56]. Alternatively, in the case of large datasets, one can in addition repeatedly sample a subset of the users and consider their last n actions in the test set.

5.2.2 Evaluation Metrics. For many application scenarios of sequence-aware recommenders, standard *classification* and *ranking* metrics can be used for performance evaluation. The set of metrics used in the reviewed papers include Precision, Recall, Mean Average Rank (MAR), Mean Average Precision (MAP), Mean Reciprocal Rank (MRR), Normalized Discounted Cumulative Gain

[16] http://2015.recsyschallenge.com/.

(NDCG), and the F1 metric. Typically, most of the ranking metrics are highly correlated when evaluated with a fixed-size split and their choice does not largely affect the outcomes [25, 44].

For some application domains, such as context adaptation, we can use standard classification metrics (e.g., precision and recall) since the recommendation problem is a classical one and sequence-based techniques are used to generate better recommendations. However, when the application domain requires recommendations to fulfill an explicit or implicit order constraint, ranking metrics such as MAP have to be preferred over classification metrics. *Error* metrics were considered only in [117] and, in general, do not fit sequence-aware recommendation problems well, where in almost all situations it is important to consider entire lists of recommended items.

In some application domains, ranking metrics alone cannot fully inform us about the quality of the recommendations. In playlist recommendation, for example, we may want to assess the capability of the recommender in generating good quality transitions between subsequent songs. Given a current track, there may be many other tracks that are almost equally likely good to be played next, and ranking metrics do not take that aspect into account. This can in turn lead to the effect that more popular tracks are favored by an algorithm [58]. Therefore, alternative metrics derived from natural language processing, like the Average Log-Likelihood, can be adopted instead [19, 20]. However, several concerns on the actual quality of the recommendations and on the interpretability of the results with these metrics have been raised [15].

Generally, when the goal of a recommender is not limited to predicting the next-best action but a sequence or set of events that are related to each other, alternative "multi-metric" evaluation approaches are required that can take multiple quality factors into account in parallel. They can consider, as mentioned, the order of the recommendations or the internal coherence or diversity of the recommended list as a whole. In the music domain, one might also be interested that the set of next tracks is coherent not only in itself but with the last played tracks. In many cases, the different quality factors can lead to tradeoff situations like "accuracy vs. diversity," which have to be balanced by a recommendation algorithm [63].

Differently from conventional recommender systems, many sequence-aware application domains recommend also items that the user already knows or has purchased in the past (see Section 3.3). In these application scenarios, the evaluation protocol has to be adapted in order to consider already known items when computing the metrics.

Finally, sequence learning methods are often computationally more complex than traditional matrix-completion algorithms. Therefore, the evaluation of sequence-aware recommender systems should also discuss the time and space complexity of the methods and report running times for typical training data sets so that the scalability can be assessed.

Overall, a general open issue in the field is the lack of "standard" metrics to assess quality criteria when recommendation lists as a whole should be evaluated, for example, with respect to their diversity or coherence. Usually, a selection of meta-data features is used to measure, for example, *intra-list* diversity or the smoothness of the transitions between the objects in the list. However, to which extent such measures reflect the users' quality perceptions often remains largely open.

5.2.3 Datasets. In recent years, more and more datasets to benchmark different sequence-aware algorithms in a reproducible way have become available to researchers. Table 5 shows the characteristics of a number of public datasets that were used in existing works.

Some researchers rely on traditional rating datasets with timestamps to evaluate sequence-aware algorithms. In reality, however, the point in time when users rate an item might be quite different from the one when the user experienced the item (e.g., watched a movie). The design of corresponding algorithms and the interpretation of the results must therefore be done with care.

Table 5. Publicly Available Datasets for Sequence-Aware Recommendation

Short name	Domain	Users	Items	Events	Sessions	Reference
Amazon	EC	20M	6M	143M	N/A	[43]
Epinions	EC	40k	110k	181k	N/A	[43]
RecSys Chall. 2015	EC	N/A	38k	34M	9.5M	[45]
Ta-feng	EC	32k	24k	829k	N/A	[110, 116]
TMall	EC	1k	10k	5k	N/A	[110, 116]
AVITO	EC	N/A	4k	767k	32k	[107]
Retailrocket	EC	1.4M	235k	2.7M	N/A	[1]
Microsoft	WWW	27k/13k	8k	13k/55k	32k/5k	[125, 126]
MSNBC	WWW	1.3M/87k	1k	476k/180k	N/A	[115, 126]
Delicious	WWW	8.8k	3.3k	60k	45k	[112]
CiteULike	WWW	53k	1.8k	2.1M	40k	[112]
Outbrain	WWW	700M	560*	2B	N/A	[2]
AOL	Query	650k	N/A	17M	2.5M	[102]
Adressa	News	15k	923	2.7M	N/A	[39]
Foursquare 1	POI	31k	N/A	N/A	N/A	[41]
Foursquare 2	POI	225k	N/A	22.5M	N/A	[21, 43]
Gowalla 1	POI	54k	367k	4M	N/A	[21, 43]
Gowalla 2	POI	196k	N/A	6.4M	N/A	[76, 77]
NYC Taxi Dataset	POI	N/A	299	N/A	670k	[29]
Global Terrorism DB	POI	N/A	N/A	N/A	N/A	[76]
Art Of the Mix	Music	N/A	218k	N/A	29k	[40, 57, 81]
30Music	Music	40k	5.6M	31M	2.7M	[106, 108]

EC: E-commerce, WWW: Web browsing, POI: points of interest, *number of sites.

5.3 On User Studies and Field Tests

User studies, which for example aim to evaluate the subjective quality perception of users in sequence-aware recommendation scenarios, are comparably rare. In a recent study in the context of next-track music recommendation [66], researchers compared the quality perception of different algorithms. They designed an online experiment where the main task for the 277 participants was to pick one of the recommendations that were provided by four different algorithms as the best continuation of a given playlist.[17] One of the main results of the study was that considering musical features not only helps to increase the homogeneity of the recommendations in terms of computational (offline) measures, but also to a better quality perception by users. At the same time, focusing on generally popular tracks, which are in many cases already known to the users, showed to be a comparably "safe" strategy, at least in the short term. While the study provided evidence that in this particular domain, the chosen offline measures can be suitable proxies for certain aspects, user studies also have their limitations, for example, in terms of the often artificial setup.

The number of reports on *field tests* for sequence-aware recommendation scenarios is limited as well. The few papers that exist include [33], [37], [62], [84], and [97]. In [62], the authors, for example, report the results of an A/B test, where one strategy was to simply remind users of previously seen items in a session-aware recommendation scenario. In that experiment it turned

[17] A similar study in the music domain was presented in [10].

out that reminding users of recently seen items is not only leading to comparably high accuracy in offline experiments, but also to a certain business value in the real world. Differently from this work, the studies of Garcin et al. [33] in the news domain revealed that the results of an offline experiment were not indicative for the performance of algorithms when deployed in a real system. Specifically, a session-based algorithm showed to be much more effective than a popularity-based strategy that worked best in the offline experiments. Overall, finding good proxy measures to predict the success of a recommender based on offline simulations is similarly challenging for sequence-aware approaches as for other types of recommendation problems.

6 SUMMARY AND FUTURE DIRECTIONS

Sequence-aware recommendation—in particular in the forms of session-based and session-aware recommendation—is a highly relevant problem in practice. Researchers have developed a variety of algorithmic proposals over the past 15 years, and with this survey, our goal is to categorize these approaches and to review the research practice in the field. Throughout the article, we have identified a number of open research questions, among them the following.

Intent Detection. Context-adaptation is one major goal in session-aware recommendation, and the most crucial task here is to estimate the users' context and short-intents. However, these intents cannot always be reliably estimated from the first few interactions of a session. For example, in the music streaming domain, is the user in the mood to discover something new or rather interested in consuming known things [67]? In e-commerce, is the user currently browsing the catalog to understand the range of options, or is she or he interested in re-inspecting a short list of candidates to purchase? To address these questions, future works could, for example, consider existing works in the field of *query intent understanding* from the information retrieval field, using, for example, external knowledge [49]. In addition, research could build on existing theories from other fields like Marketing, where models like AIDA (Attention, Interest, Desire, and Action) are used to describe the different phases of a consumer, from becoming aware of an offering to the actual purchase. As discussed in [101], there are also domains where the user's interest can change even within a session; for example, when a user has read a number of news stories of a certain category and then becomes satiated with the topic. Finally, in the context of intent detection, very little research exists on how to give users the opportunity to correct the system's assumption in case they are wrong. On Amazon.com, some basic mechanisms for user control exist, but it is unclear if they are broadly used by consumers [60].

Combining Short-Term and Longer-Term Profiles. To be able to assess the consumers' state in the decision-making process, information about their behavior during recent sessions and probably also their longer-term behavior has to be taken into account. The most recent works in the field focus on the session-based recommendation problem, where this past information is not available. However, there are many domains where longer-term user profiles exist and more research is required to better leverage this information. Some works [56] show that while the short-term intentions should be predominant in the selection of the recommendations, considering longer-term behavioral patterns and preferences of the individual user (e.g., toward certain brands in an e-commerce scenario) can be important. Existing works like [12] use a simple static weighting of long-term and short-term models or rely on on re-ranking the results of the long-term models based on assumed short-term intents [56, 63]. Better integrated models as used, for example, in [91] are therefore still needed.

Leveraging Additional Data and General Trends. Generally, in the context of user profiling, researchers often rely on one or a few types of specific user interactions like item view events or check-ins at certain locations. In real-world applications, usually much richer types of information are available. Not only are there multiple additional actions related to certain items (e.g., add-to-wishlist, add-to-cart in the e-commerce domain), there are also other relevant user actions like search or category navigation which are not considered to a large extent in today's research. Also, the information about how the users entered the site (e.g., through a search engine) can be a valuable piece of information to assess the user intent from the first few interactions. Going beyond behavioral patterns of the individual, more research is also required in terms of detecting behavioral trends and interest shifts in the entire user community. In different application domains where item recency plays a role, including news, music, and e-commerce, being able to detect and leverage short-term trends in a community is highly important. For the e-commerce domain, works like [52] or [62] show that focusing on items that were trending on the platform in the last few days can be crucial to further improve the recommendation quality.

Toward standardized and more comprehensive evaluations. Finally, our review shows that today a variety of protocol variants and computational metrics are used to compare sequence-aware algorithms, making it difficult to assess how the field progresses. Going beyond the subtle details of the evaluation protocols (e.g., of how to specifically determine the hold-out set), today's research in sequence-aware recommenders is still largely focused on accuracy measures. While this is a known issue also for the matrix-completion formulation of the recommendation problem, in sequence-aware recommendation, and in particular in session-aware recommendation, the usefulness of certain recommendations can largely depend on the contextual situation of the users, for example, if they are exploring the item space, inspecting a smaller choice set, looking for complements to a recently inspected item, and so forth. These aspects call for more purpose-oriented evaluation approaches [24, 50], which take the users' specific contextual situation and goals into account.

REFERENCES

[1] Retrieved January 2018 from https://www.kaggle.com/retailrocket/ecommerce-dataset.

[2] Retrieved January 2018 from https://www.kaggle.com/c/outbrain-click-prediction/data.

[3] Gediminas Adomavicius and Alexander Tuzhilin. 2005. 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 (2005), 734–749.

[4] Gediminas Adomavicius and Alexander Tuzhilin. 2011. Context-aware recommender systems. In *Recommender Systems Handbook*. Springer, 217–253.

[5] Rakesh Agrawal and Ramakrishnan Srikant. 1994. Fast algorithms for mining association rules in large databases. In *VLDB'94*. 487–499.

[6] Rakesh Agrawal and Ramakrishnan Srikant. 1995. Mining sequential patterns. In *ICDE'95*. 3–14.

[7] Amjad Almahairi, Kyle Kastner, Kyunghyun Cho, and Aaron Courville. 2015. Learning distributed representations from reviews for collaborative filtering. In *RecSys'15*. 147–154.

[8] Ricardo Baeza-Yates, Carlos Hurtado, and Marcelo Mendoza. 2004. Query recommendation using query logs in search engines. In *EDBT'04*. 588–596.

[9] Ricardo Baeza-Yates, Di Jiang, Fabrizio Silvestri, and Beverly Harrison. 2015. Predicting the next app that you are going to use. In *WSDM'15*. 285–294.

[10] Luke Barrington, Reid Oda, and Gert R. G. Lanckriet. 2009. Smarter than genius? Human evaluation of music recommender systems. In *ISMIR'09*. 357–362.

[11] Ron Begleiter, Ran El-Yaniv, and Golan Yona. 2004. On prediction using variable order Markov models. *JAIR* 22, 1 (2004), 385–421.

[12] Daniel Billsus, Michael J. Pazzani, and James Chen. 2000. A learning agent for wireless news access. In *IUI'00*. 33–36.

[13] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent Dirichlet allocation, In *JMLR'03*. *JMLR* 3 (2003), 993–1022.

[14] Geoffray Bonnin and Dietmar Jannach. 2013. Evaluating the quality of generated playlists based on hand-crafted samples. In *ISMIR'13*. 263–268.

[15] Geoffray Bonnin and Dietmar Jannach. 2014. Automated generation of music playlists: Survey and experiments. *ACM CSUR* 47, 2 (2014), Article 26, 35 pages.

[16] Robin Burke. 2007. Hybrid web recommender systems. In *The Adaptive Web*, Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl (Eds.). Springer, Chapter: Hybrid Web Recommender Systems, 377–408.

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

[18] Huanhuan Cao, Daxin Jiang, Jian Pei, Qi He, Zhen Liao, Enhong Chen, and Hang Li. 2008. Context-aware query suggestion by mining click-through and session data. In *KDD'08*. 875–883.

[19] Shuo Chen, Josh L. Moore, Douglas Turnbull, and Thorsten Joachims. 2012. Playlist prediction via metric embedding. In *KDD'12*. 714–722.

[20] Shuo Chen, Jiexun Xu, and Thorsten Joachims. 2013. Multi-space probabilistic sequence modeling. In *KDD'13*. 865–873.

[21] Chen Cheng, Haiqin Yang, Michael R. Lyu, and Irwin King. 2013. Where you like to go next: Successive point-of-interest recommendation. In *IJCAI'13*. 2605–2611.

[22] Szu-Yu Chou, Yi-Hsuan Yang, Jyh-Shing Roger Jang, and Yu-Ching Lin. 2016. Addressing cold start for next-song recommendation. In *RecSys'16*. 115–118.

[23] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR* abs/1412.3555.

[24] Paolo Cremonesi, Franca Garzotto, and Roberto Turrin. 2013. User-centric vs. system-centric evaluation of recommender systems. In *IFIP Conference on Human-Computer Interaction*. Springer, 334–351.

[25] Paolo Cremonesi, Yehuda Koren, and Roberto Turrin. 2010. Performance of recommender algorithms on top-n recommendation tasks. In *RecSys'10*. 39–46.

[26] Nofar Dali Betzalel, Bracha Shapira, and Lior Rokach. 2015. "Please, not now!": A model for timing recommendations. In *RecSys'15*. 297–300.

[27] Nemanja Djuric, Vladan Radosavljevic, Mihajlo Grbovic, and Narayan Bhamidipati. 2014. Hidden conditional random fields with deep user embeddings for ad targeting. In *ICDM'14*. 779–784.

[28] Paul Dourish. 2004. What we talk about when we talk about context. *Personal Ubiquitous Comput.* 8, 1 (2004), 19–30.

[29] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. 2016. Recurrent marked temporal point processes: Embedding event history to vector. In *KDD'16*. 1555–1564.

[30] Shanshan Feng, Xutao Li, Yifeng Zeng, Gao Cong, Yeow Meng Chee, and Quan Yuan. 2015. Personalized ranking metric embedding for next new POI recommendation. In *IJCAI'15*. 2069–2075.

[31] Gernot A. Fink. 2014. *Markov Models for Pattern Recognition: From Theory to Applications*. Springer.

[32] Florent Garcin, Christos Dimitrakakis, and Boi Faltings. 2013. Personalized news recommendation with context trees. In *RecSys'13*. 105–112.

[33] Florent Garcin, Boi Faltings, Olivier Donatsch, Ayar Alazzawi, Christophe Bruttin, and Amr Huber. 2014. Offline and online evaluation of news recommender systems at swissinfo.ch. In *RecSys'14*. 169–176.

[34] Zoubin Ghahramani. 2002. An introduction to hidden Markov models and Bayesian networks. In *Hidden Markov Models*. World Scientific Publishing, Chapter: An Introduction to Hidden Markov Models and Bayesian Networks, 9–42.

[35] Carlos A. Gomez-Uribe and Neil Hunt. 2015. The Netflix recommender system: Algorithms, business value, and innovation. *ACM TMIS* 6, 4 (2015), 13:1–13:19.

[36] Alex Graves. 2013. Generating sequences with recurrent neural networks. *CoRR* 1308.0850.

[37] Mihajlo Grbovic, Vladan Radosavljevic, Nemanja Djuric, Narayan Bhamidipati, Jaikit Savla, Varun Bhagwan, and Doug Sharp. 2015. E-commerce in your inbox: Product recommendations at scale. In *KDD'15*. 1809–1818.

[38] Asnat Greenstein-Messica, Lior Rokach, and Michael Friedman. 2017. Session-based recommendations using item embedding. In *IUI'17*. 629–633.

[39] Jon Atle Gulla, Lemei Zhang, Peng Liu, Özlem Özgöbek, and Xiaomeng Su. 2017. The Adressa dataset for news recommendation. In *International Conference on Web Intelligence*. 1042–1048.

[40] Negar Hariri, Bamshad Mobasher, and Robin Burke. 2012. Context-aware music recommendation based on latent topic sequential patterns. In *RecSys'12*. 131–131.

[41] Jing He, Xin Li, Lejian Liao, Dandan Song, and William Cheung. 2016. Inferring a personalized next point-of-interest recommendation model with latent behavior patterns. In *AAAI'16*.

[42] Qi He, Daxin Jiang, Zhen Liao, Steven C. H. Hoi, Kuiyu Chang, Ee-Peng Lim, and Hang Li. 2009. Web query recommendation via sequential query prediction. In *ICDE'09*. 1443–1454.

[43] Ruining He and Julian McAuley. 2016. Fusing similarity models with Markov chains for sparse sequential recommendation. *CoRR* 1609.09152.

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

[45] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In *ICLR'16*.

[46] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. 2016. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *RecSys'16*.

[47] Mehdi Hosseinzadeh Aghdam, Negar Hariri, Bamshad Mobasher, and Robin Burke. 2015. Adapting recommendations to contextual changes using hierarchical hidden Markov models. In *RecSys'15*. 241–244.

[48] Sue-Chen Hsueh, Ming-Yen Lin, and Chien-Liang Chen. 2008. Mining negative sequential patterns for e-commerce recommendations. In *APSCC'08*. 1213–1218.

[49] Jian Hu, Gang Wang, Fred Lochovsky, Jian-tao Sun, and Zheng Chen. 2009. Understanding user's query intent with Wikipedia. In *WWW'09*. 471–480.

[50] Dietmar Jannach and Gedas Adomavicius. 2016. Recommendations with a purpose. In *RecSys'16*. 7–10.

[51] Dietmar Jannach and Malte Ludewig. 2017. Determining characteristics of successful recommendations from log data—A case study. In *SAC'17*.

[52] Dietmar Jannach and Malte Ludewig. 2017. When recurrent neural networks meet the neighborhood for session-based recommendation. In *RecSys'17*. 306–310.

[53] Dietmar Jannach and Malte Ludewig. 2017. Investigating personalized search in e-commerce. In *FLAIRS'17*.

[54] Dietmar Jannach and Malte Ludewig. 2017. Determining characteristics of successful recommendations from log data—A case study. In *SAC'17*.

[55] Dietmar Jannach, Markus Zanker, Mouzhi Ge, and Marian Gröning. 2012. Recommender systems in computer science and information systems—A landscape of research. In *EC-Web'12*. 76–87.

[56] Dietmar Jannach, Lukas Lerche, and Michael Jugovac. 2015. Adaptation and evaluation of recommendations for short-term shopping goals. In *RecSys'15*. 211–218.

[57] Dietmar Jannach, Lukas Lerche, and Iman Kamehkhosh. 2015. Beyond "hitting the hits": Generating coherent music playlist continuations with the right tracks. In *RecSys'15*. 187–194.

[58] Dietmar Jannach, Lukas Lerche, Iman Kamehkhosh, and Michael Jugovac. 2015. What recommenders recommend: An analysis of recommendation biases and possible countermeasures. *User Modeling and User-Adapted Interaction* 25, 5 (2015), 427–491.

[59] Dietmar Jannach, Michael Jugovac, and Lukas Lerche. 2016. Supporting the design of machine learning workflows with a recommendation system. *ACM TiiS* 6, 1 (2016).

[60] Dietmar Jannach, Sidra Naveed, and Michael Jugovac. 2016. User control in recommender systems: Overview and interaction challenges. In *EC-Web 2016*.

[61] Dietmar Jannach, Paul Resnick, Alexander Tuzhilin, and Markus Zanker. 2016. Recommender systems—Beyond matrix completion. *Commun. ACM* 59, 11 (2016), 94–102.

[62] Dietmar Jannach, Malte Ludewig, and Lukas Lerche. 2017. Session-based item recommendation in e-commerce: On short-term intents, reminders, trends, and discounts. *UMUAI* 27, 3–5 (2017), 351–392.

[63] Michael Jugovac, Dietmar Jannach, and Lukas Lerche. 2017. Efficient optimization of multiple recommendation quality factors according to individual user tendencies. *Expert Syst. Appl.* 81 (2017), 321–331.

[64] Santosh Kabbur, Xia Ning, and George Karypis. 2013. FISM: Factored item similarity models for top-N recommender systems. In *KDD'13*. 659–667.

[65] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. 1996. Reinforcement learning: A survey. *J. Artif. Intell. Res* 4 (1996), 237–285.

[66] Iman Kamehkhosh and Dietmar Jannach. 2017. User perception of next-track music recommendations. In *UMAP'17*.

[67] Komal Kapoor, Vikas Kumar, Loren Terveen, Joseph A. Konstan, and Paul Schrater. 2015. "I like to explore sometimes": Adapting to dynamic user novelty preferences. In *RecSys'15*. 19–26.

[68] Yehuda Koren. 2010. Collaborative filtering with temporal dynamics. *Commun. ACM* 53, 4 (2010), 89–97.

[69] Quoc V. Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *ICML'14*. 1188–1196.

[70] Lukas Lerche, Dietmar Jannach, and Malte Ludewig. 2016. On the value of reminders within e-commerce recommendations. In *UMAP'16*. 27–25.

[71] Benjamin Letham, Cynthia Rudin, and David Madigan. 2013. Sequential event prediction. *Mach. Learn.* 93, 2-3 (2013), 357–380.

[72] Defu Lian, Vincent W. Zheng, and Xing Xie. 2013. Collaborative filtering meets next check-in location prediction. In *WWW'13*. 231–232.

[73] Kwan Hui Lim, Jeffrey Chan, Christopher Leckie, and Shanika Karunasekera. 2015. Personalized tour recommendation based on user interests and points of interest visit durations. In *IJCAI'15*. 1778–1784.

[74] Zachary C. Lipton, John Berkowitz, and Charles Elkan. 2015. A critical review of recurrent neural networks for sequence learning. *CoRR* 1506.00019.

[75] Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. 2010. Personalized news recommendation based on click behavior. In *IUI'10*. 31–40.

[76] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. Predicting the next location: A recurrent model with spatial and temporal contexts. In *AAAI'16*.

[77] Yanchi Liu, Chuanren Liu, Bin Liu, Meng Qu, and Hui Xiong. 2016. Unified point-of-interest recommendation with temporal interval assessment. In *KDD'16*. 1015–1024.

[78] Eric Hsueh-Chan Lu, Yi-Wei Lin, and Jing-Bin Ciou. 2014. Mining mobile application sequential patterns for usage prediction. In *GrC'14*. 185–190.

[79] Nizar R. Mabroukeh and C. I. Ezeife. 2010. A taxonomy of sequential pattern mining algorithms. *ACM CSUR* 43, 1 (2010), 3:1–3:41.

[80] François Maillet, Douglas Eck, Guillaume Desjardins, and Paul Lamere. 2009. Steerable playlist generation by learning song similarity from radio station playlists. In *ISMIR'09*.

[81] Brian McFee and Gert Lanckriet. 2011. The natural language of playlists. In *ISMIR'11*. 537–541.

[82] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS'13*. 3111–3119.

[83] Bamshad Mobasher, Honghua Dai, Tao Luo, and Miki Nakagawa. 2002. Using sequential and non-sequential patterns in predictive Web usage mining tasks. In *ICDM'02*. 669–672.

[84] Omar Moling, Linas Baltrunas, and Francesco Ricci. 2012. Optimal radio channel recommendations with explicit and implicit feedback. In *RecSys'12*. 75–82.

[85] Joshua Moore, Shuo Chen, Douglas Turnbull, and Thorsten Joachims. 2013. Taste over time: The temporal dynamics of user preferences. In *ISMIR'13*.

[86] Miki Nakagawa and Bamshad Mobasher. 2003. Impact of site characteristics on recommendation models based on association rules and sequential patterns. In *IJCAI'03*.

[87] Nagarajan Natarajan, Donghyuk Shin, and Inderjit S. Dhillon. 2013. Which app will you use next?: Collaborative filtering with interactional context. In *RecSys'13*. 201–208.

[88] Aditya Parameswaran, Petros Venetis, and Hector Garcia-Molina. 2011. Recommendation systems with complex constraints: A course recommendation perspective. *ACM Trans. Inf. Syst.* 29, 4 (2011), 20:1–20:33.

[89] Steffen Pauws, Wim Verhaegh, and Mark Vossen. 2006. Fast generation of optimal music playlists using local search. In *ISMIR'06*.

[90] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP'14*. 1532–1543.

[91] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *RecSys'17*.

[92] Siddharth Reddy, Igor Labutov, and Thorsten Joachims. 2016. Learning student and content embeddings for personalized lesson sequence recommendation. In *ACM Learning @ Scale'16*. 93–96.

[93] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI'09*. 452–461.

[94] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized Markov chains for next-basket recommendation. In *WWW'10*. 811–820.

[95] Cynthia Rudin, Benjamin Letham, Ansaf Salleb-Aouissi, Eugene Kogan, and David Madigan. 2011. Sequential event prediction with association rules. In *COLT'11*. 615–634.

[96] Nachiketa Sahoo, Param Vir Singh, and Tridas Mukhopadhyay. 2012. A hidden Markov model for collaborative filtering. *MIS Q.* 36, 4 (2012), 1329–1356.

[97] Guy Shani, David Heckerman, and Ronen I. Brafman. 2005. An MDP-based recommender system. *J. Mach. Learn. Res.* 6 (2005), 1265–1295.

[98] Harold Soh, Scott Sanner, Madeleine White, and Greg Jamieson. 2017. Deep sequential recommendation for personalized adaptive user interfaces. In *IUI'17*. 589–593.

[99] Qiang Song, Jian Cheng, Ting Yuan, and Hanqing Lu. 2015. Personalized recommendation meets your next favorite. In *CIKM'15*. 1775–1778.

[100] Yang Song, Ali Mamdouh Elkahky, and Xiaodong He. 2016. Multi-rate deep learning for temporal recommendation. In *SIGIR'16*. 909–912.

[101] Yicheng Song, Nachiketa Sahoo, and Elie Ofek. 2017. When and How to Diversify—A Multi-Category Utility Model of Consumer Response to Content Recommendations. Available at SSRN: https://ssrn.com/abstract=2880779.

[102] Alessandro Sordoni, Yoshua Bengio, Hossein Vahabi, Christina Lioma, Jakob Grue Simonsen, and Jian-Yun Nie. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *CIKM'15*. 553–562.

[103] Yukihiro Tagami, Hayato Kobayashi, Shingo Ono, and Akira Tajima. 2015. Modeling user activities on the web using paragraph vector. In *WWW'15*. 125–126.

[104] Maryam Tavakol and Ulf Brefeld. 2014. Factored MDPs for detecting topics of user sessions. In *RecSys'14*. 33–40.

[105] Michele Trevisiol, Luca Maria Aiello, Rossano Schifanella, and Alejandro Jaimes. 2014. Cold-start news recommendation with domain-dependent browse graph. In *RecSys'14*. 81–88.

[106] Roberto Turrin, Andrea Condorelli, Roberto Pagano, Massimo Quadrana, and Paolo Cremonesi. 2015. Large scale music recommendation. In *LSRS 2015*.

[107] Bartlomiej Twardowski. 2016. Modelling contextual information in session-aware recommender systems with neural networks. In *RecSys'16*. 273–276.

[108] Flavian Vasile, Elena Smirnova, and Alexis Conneau. 2016. Meta-Prod2Vec: Product embeddings using side-information for recommendation. In *RecSys'16*. 225–232.

[109] Jian Wang and Yi Zhang. 2013. Opportunity model for e-commerce recommendation: Right product; right time. In *SIGIR'13*. 303–312.

[110] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2015. Learning hierarchical representation model for nextbasket recommendation. In *SIGIR'15*. 403–412.

[111] Xiang Wu, Qi Liu, Enhong Chen, Liang He, Jingsong Lv, Can Cao, and Guoping Hu. 2013. Personalized next-song recommendation in online karaokes. In *RecSys'13*. 137–140.

[112] Liang Xiang, Quan Yuan, Shiwan Zhao, Li Chen, Xiatian Zhang, Qing Yang, and Jimeng Sun. 2010. Temporal recommendation on graphs via long- and short-term preference fusion. In *KDD'10*. 723–732.

[113] Jie Xu, Tianwei Xing, and Mihaela van der Schaar. 2016. Personalized course sequence recommendations. *IEEE Trans. Signal Process.* 64, 20 (2016), 5340–5352.

[114] Xiaohui Yan, Jiafeng Guo, and Xueqi Cheng. 2011. Context-aware query recommendation by learning high-order relation in query logs. In *CIKM'11*. 2073–2076.

[115] Ghim-Eng Yap, Xiao-Li Li, and Philip S. Yu. 2012. Effective next-items recommendation via personalized sequential pattern mining. In *DASFAA'12*. 48–64.

[116] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. A dynamic recurrent model for next basket recommendation. In *SIGIR'16*. 729–732.

[117] Hong Yu and Mark O. Riedl. 2012. A sequential recommendation approach for interactive personalized story generation. In *AAMAS'12*. 71–78.

[118] Hao Zang, Yue Xu, and Yuefeng Li. 2010. Non-redundant sequential association rule mining and application in recommender systems. In *WI-IAT'10*. 292–295.

[119] H. Zhang, W. Ni, X. Li, and Y. Yang. 2017. Modeling the heterogeneous duration of user interest in time-dependent recommendation: A hidden semi-Markov approach. *Trans. Syst. Man. Cybern. Syst.* 48, 2 (2017), 177–194.

[120] Jia-Dong Zhang, Chi-Yin Chow, and Yanhua Li. 2014. LORE: Exploiting sequential influence for location recommendations. In *SIGSPATIAL'14*. 103–112.

[121] Yuyu Zhang, Hanjun Dai, Chang Xu, Jun Feng, Taifeng Wang, Jiang Bian, Bin Wang, and Tie-Yan Liu. 2014. Sequential click prediction for sponsored search with recurrent neural networks. In *AAAI'14*. 1369–1375.

[122] Gang Zhao, Mong Li Lee, Wynne Hsu, and Wei Chen. 2012. Increasing temporal diversity with purchase intervals. In *SIGIR'12*. 165–174.

[123] Gang Zhao, Mong LI Lee, and Hsu Wynne. 2014. Utilizing purchase intervals in latent clusters for product recommendation. In *SNAKDD'14*. Article 4, 9 pages.

[124] Elena Zheleva, John Guiver, Eduarda Mendes Rodrigues, and Nataša Milić-Frayling. 2010. Statistical models of music-listening sessions in social media. In *WWW'10*. 1019–1028.

[125] Baoyao Zhou, Siu Cheung Hui, and Kuiyu Chang. 2004. An intelligent recommender system using sequential Web access patterns. In *CIS'04*. 393–398.

[126] Andrew Zimdars, David Maxwell Chickering, and Christopher Meek. 2001. Using temporal data for making recommendations. In *UAI'01*. 580–588.

Received July 2017; revised February 2018; accepted February 2018