Session-based Recommender Systems

Dietmar Jannach, Massimo Quadrana and Paolo Cremonesi

Abstract Session-based recommendation is concerned with the problem of tailoring item suggestions according to the short-term needs and assumed intents of the user. The input in this recommendation scenario consists of an often very short sequence of user interactions that are observed in an ongoing usage session, and in many cases longer-term preferences of the users are not available. Such problems are highly relevant in practice because *(i)* recommendations should often be made also to anonymous and first-time users and because *(ii)* the users' intents can change from session to session. In this chapter, we first elaborate on practical application scenarios for session-based recommender systems, provide a characterization of the problem class, and outline key challenges. Afterwards, we review technical approaches to session-based recommendation and report common practices of evaluating such systems. The chapter ends with a discussion of open challenges and an outlook on future directions in the area.[1]

1 Introduction

In session-based recommendation scenarios, the goal is to tailor the suggestions that are made to the users according to their assumed short-term intents during an ongoing usage session. The main inputs in such settings consists of *(i)* the sequence

Dietmar Jannach
University of Klagenfurt, Austria, e-mail: dietmar.jannach@aau.at

Massimo Quadrana
Pandora Media LLC, USA, e-mail: mquadrana@pandora.com

Paolo Cremonesi
Politecnico di Milano, Italy, e-mail: paolo.cremonesi@polimi.it

[1] Cite as: Dietmar Jannach, Massimo Quadrana, Paolo Cremonesi: Session-based Recommender Systems, in Ricci et al. (eds.), Recommender Systems Handbook, 3rd edition, Springer, 2022, pp. 301–335

of interactions with an individual user that were observed by the system since the session started, *(ii)* information about past sessions by other users. Typical application scenarios can, for example, be found in the e-commerce domain, where the shopping goals of a consumer can change from session to session and where these goals can also relate to entirely different product categories in each session. Furthermore, in many cases, the past interests and preferences of an individual user might not even be known to the system, e.g., because it is a first-time user or a user that is not logged-in and thus anonymous. The challenge in such situations therefore consists in making helpful recommendations based on very little amounts of information about the current user. As an extreme case, consider Amazon's *"Customer's who bought . . . also bought"* recommendations, where the recommendations are tailored—albeit in a non-personalized way—to the one single item that is currently viewed by the user.

Such challenging situations do, however, not only arise in the e-commerce domain, but are common in various application areas of recommender systems (RS) such as music, video, or news recommendation. From a historical perspective, early influential technical approaches for session-based recommendation were proposed in the first half of the 2000s, e.g., [84, 89, 99]. However, despite the high practical relevance of the problem, research in this area remained sparse, in particular when compared to the large number of technical works that focus on making non-contextualized recommendations based on long-term preference profiles. Until about 2015, individual proposals for session-based recommendations were published mostly for the popular domains of e-commerce, music, and news, e.g., [17, 23, 34, 41, 52, 108], or for niche applications like adaptive modeling support for machine learning workflows [50]. In 2015, the data challenge associated with the ACM Conference on Recommender Systems was focusing on specific session-based recommendation problems, and a larger dataset containing anonymous click logs was released. This dataset release and the then-starting boom in deep learning in recommender systems contributed to the largely increased interest in session-based recommendation problems that we observe today.

In this chapter, we review the topic of session-based recommender systems (SBRS) from different perspectives. First in Section 2, we characterize the problem setting and its variants in more detail in the context of Sequence-aware Recommender Systems [87]. We then review technical approaches to build SBRS in Section 3. Section 4 is afterwards devoted to questions regarding the evaluation and comparison of session-based recommendation approaches. In Section 5, finally, we given an outlook on future directions in this area.

2 Problem Characterization

Session-based recommendation problems differ from the traditional "matrix-filling" [93] abstraction in two main ways, see also Table 1.

Table 1 Comparing session-based recommendation with traditional recommendation scenarios.

	Matrix-filling Recommender System	**Session-based Recommendation**
Data	User-item rating matrix	Time-ordered sequence of interactions, organized in sessions
Task	Make time-agnostic recommendations for long-term preferences	Make recommendations for ongoing session

- *Data:* The main input to an SBRS is a time-ordered sequence of logged user interactions.
- *Computational task:* The main task of an SBRS is to make recommendations that are relevant for an ongoing session.

Figure 1 illustrates the problem setting in more detail. The inputs are shown on the left-hand side. First, there are the interactions—for example, purchases or item view actions—that were recorded in the past for an *entire user community*. These can be used to train machine learning models or to apply data mining techniques. SBRS therefore are collaborative filtering (CF) systems [57] or hybrids with a CF component.

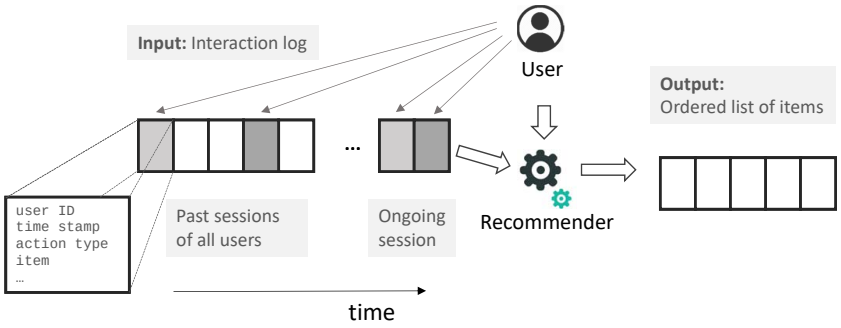

Fig. 1 High-level overview of Sequence-Aware Recommendation, adapted from [87].

Each entry (interaction record) in the log can have additional attributes. The most common attributes are the type of the interaction and which item is affected by the interaction, e.g., a purchase of a certain item. SBRS can in principle also consider interactions that are not item-related, for example, search actions. In Figure 1, the different shades of grey indicate different types of actions.

The log of past interactions is organized in *usage sessions*, which means that each interaction record has some information attached that describes to which session it belongs. This can, for example, be a session ID, which is explicitly assigned by the application that creates the log. If no session IDs are available, one can use information from browser cookies or similar techniques to identify which records

belong together. In some cases, IDs of registered users might assigned to each record, even though the common assumption in SBRS is that users are anonymous.[2]

In cases when there are no explicit session IDs assigned, sessions can be constructed from the log post-hoc with the help of heuristics based, e.g., on cookies. A typical heuristic in some domains is to assume that a new session starts after an extended period of inactivity by the user, e.g., for 30 minutes.

The second input to an SBRS are the interactions that are observed in the *ongoing* usage session. Again, these interactions can have additional attributes, in particular the type of interaction and the affected item. The *computational task* of an SBRS is then to take these currently observed interactions and to make recommendations that are relevant for the ongoing session, using the past interaction data as background information, e.g., in the form of a machine learning model. The *output* of an SBRS, finally, consists of a rank-ordered list of recommended items as usual.

In [87], the following terminology is used to describe two variants of the underlying recommendation problem:

- *Session-based recommendation:* In this scenario, no user IDs are attached to the interactions or sessions. This means that no personalization of the recommendations according to longer-term preferences is possible. These problems are in the focus of this chapter.
- *Session-aware recommendation:* In this case, the interactions or sessions have user IDs attached, which makes it possible to consider long-term preferences when recommending, as done, e.g., in [56, 85, 88] or [105].

The use of terminology in the literature is however not always consistent. Sometimes, the term *session-based* recommendation is also used for situations where long-term preferences are available, e.g., in [88]. Some authors also use the term *sequential recommendation* in the context of session-based recommendation scenarios. Session-based recommendation problems are sequential both in the sense that the goal is to predict the next interactions in a given session and that the available data is time-ordered. However, not all sequential approaches are necessarily session-based. Instead, they may be based on time-stamped and longer-term user preference information, e.g., in the context of next-POI or next-basket recommendations. *Sequence-aware recommendation* is an umbrella term that covers all discussed problem settings: session-based, session-aware, and sequential recommendation. A formal characterization of sequence-aware problems is given in [87], where the authors also discuss additional computational tasks that can be done based on sequential interaction logs, such as trend detection, repeated recommendation, or the consideration of ordering constraints among the recommendations.

Looking at existing categorizations of recommender systems, SBRS[3] are typically based only on *implicit feedback*, i.e., they face the usual problem of correctly

[2] Not connecting click data with individual user accounts may in some cases also be beneficial in terms of user privacy and the General Data Protection Regulation in the European Union.

[3] These discussions apply both for *session-based* and for *session-aware* recommender systems. For the sake of brevity, we will use the term *session-based* in this chapter, and will explicitly state when not both types of problems are meant.

interpreting the various types of user interactions [54]. SBRS are also *context-aware* systems [12]. Context adaptation in SBRS is however usually not based on explicit information about the user's contextual situation, e.g., the weather, but on the observed user interactions in the ongoing session, i.e., on the *interactional context*. Finally, there is also some relation to *time-aware* RS [20] where the user actions are time-stamped or time-ordered. Most time-aware RS in the literature are however not session-based, but are based on the user-item rating matrix abstraction. Still, these systems share the problem with SBRS that specific evaluation protocols are required that consider the ordering of the ratings or interactions.

In sum, the two main challenges of SBRS are the following:

- *Limited preference information:* Since in SBRS no user IDs are available, recommendations have to be made based only on a very small set of observed interactions. This in some ways corresponds to a *user cold-start* problem.
- *Implicit feedback:* There are various types of interactions that can be taken into account and there is some uncertainty attached to the interpretation of the observed user behavior.

3 Technical Approaches

Algorithmic approaches of different types were proposed for session-based recommendation problems over the years. Several of these technical approaches were previously also explored for the more general class of sequence-aware recommendation problems. In [87], the authors propose a categorization of existing approaches to sequence-aware recommendation based on the underling family of algorithms that is used to extract or model (sequential) patterns from the logged user interactions. An alternative approach is followed in [116], where the technical approaches for session-based recommendation are categorized into two main classes. We adopt this latter categorization in this section.

- *Model-free approaches*: In this category, we consider all algorithmic approaches based on data-mining techniques that do not involve the computation of a user or session model. Typical examples in this category are methods that are based on the extraction of frequent ordered and unordered patterns from historical session data [41, 52, 119]. Moreover, we include also all methods that perform inference based on nearest neighbors of the current user session [73] in this category.
- *Model-based approaches*: In this category, we consider all machine-learning approaches that explicitly model the user-item interactions within and across sessions. Examples of methods in this class are Markov Models [34, 47], Recurrent Neural Networks [45, 88] and Reinforcement Learning [99, 108].

In what follows, we only consider pure collaborative approaches, which are solely based on the recorded user-item interactions. In principle, all these approaches can be extended or combined with other techniques in order to consider various types

of side information, such as item meta-data or contextual information. Examples of model-based techniques that leverage side information are [30] or [46].

3.1 Model-free Approaches

Model-free approaches are often conceptually simpler, easy to implement and sometimes also less resource demanding than model-based ones. In many cases, they do not require the computation of complex mathematical models from historical session data, and they generally rely on well optimized data mining procedures to extract patterns from sessions. Another potential advantage is that they can be easier to analyze and debug than complex models. For example, the recommendations generated by frequent-pattern and rule mining techniques can be straightforwardly explained in the terms of the patterns that were extracted from historical session data. Their simplicity can also make them less prone to over-fitting in situations when the training data is sparse. Nonetheless, despite their simplicity, certain types of model-free algorithms—in particular nearest-neighbor techniques—can lead to competitive or even superior performance when compared to more complex model-based variants according to several offline benchmarks [73, 75, 76].

3.1.1 Sequential Pattern Mining

Frequent Pattern Mining (FPM) techniques aim to discover user consumption patterns within large datasets of transactions. These patterns can then be used to predict the most probable actions[4] that the user will take in the session given the sequence of actions they have done so far.

In session-based recommendation scenarios, the most simple approach is to determine only pairwise item co-occurrence frequencies in past sessions (training data) in order to implement buying suggestions of the form "Customers who bought . . . also bought". At run time, recommendations are made by looking up which items often appeared in the same session as those in the ongoing session. A simplified version of association rules mining for SBRS, which only considers rules of size two, is provided in [73].

These kinds of recommendations belong to the family of Association Rules [13], which identify items or actions that frequently co-occur in the same session, regardless of the order of their appearance. Based on our definition of the problem as described in Section 2, FPM-based algorithms are not session-based in the strict sense, but can be used as building blocks for more advanced approaches. Beyond simple co-occurrence patterns, one can consider Sequential Pattern Mining techniques [14], which take the order of items into account. Finally, Contiguous Sequential Patterns require that the co-occurring items are adjacent in the sequence of actions

[4] For the sake of readability, we use the terms "user action (on an item)" and "user-item interaction" interchangeably.

within a session. A simple yet often effective light-weight form of such *Sequential Rules* of size two was also proposed in [73].

3.1.2 Nearest Neighbors

Nearest-neighbor-based methods are among the simplest and most effective approaches to session-based recommendation. We describe here two variants, namely *item-based* and *session-based* nearest-neighbor methods.

Item-based methods for session-based recommendation were introduced in [46], in which the similarity between items is computed based on their co-occurrence in a set of training sessions. Every item is associated to a large sparse binary vector representing whether or not the item appeared in each of the training sessions, and the item similarity is simply computed as the cosine similarity between each pair of vectors. The recommendations are computed as the list of most similar items to the *last item* in the user session. The type of recommendations produced by item-based approaches follow closely the "Customers who bought . . . also bought" logic.

Session-based methods extend this concept beyond the last item in the session by comparing the *entire* user session with the past sessions in the training data. A comprehensive description of this family of model-free approaches is provided in [73]. In summary, given a session s, these methods determine the set N_s of the top-k most similar sessions using a suitable similarity measure $\mathrm{sim}(s_1, s_2)$, such as the Jaccard or cosine similarity over the binary vectors that represent sessions in the item space. The recommendation scores for each item $i \in s$ are computed as:

$$\mathrm{score}_{\mathrm{SKNN}}(i, s) = \sum_{n \in N_s} \mathrm{sim}(s, n) \cdot 1_n(i) \tag{1}$$

where $1_n(i)$ returns 1 if the session n contains i and 0 otherwise. Note that Jaccard and cosine similarity do not take the order of the items in a session into account. In [73] the authors therefore show how variants of this session-based nearest-neighbor method can be built to take into account the order of the user actions. A recent time-aware and sequence-aware variation of this scheme was also discussed in [36].

For both item-based and session-based nearest neighbor methods, the set of neighboring items and sessions can usually be precomputed offline to speed up the processing at prediction time [55].

3.2 Model-based approaches

Model-based approaches exploit advanced machine learning techniques to represent the complex dynamics of the user actions within and across sessions. Most recent model-based approaches to SBRS fall into the category of *sequence learning* methods, which aim to learn models from sequences of past user actions to predict future

ones [78]. Advanced sequence learning methods allow to model complex interactions between user actions in sessions. For example, certain models compute a representation of the user's latent purchase intent in the ongoing session [40, 107]. Other methods explicitly model the temporal dynamics between sessions [112]. In general, model-based algorithms therefore have the potential to identify more complex dependencies and interactions from the logs of historical user sessions.

On the other hand, the greater representational power of these family of techniques often comes at the cost of much larger amounts of training data and computational resources than the model-free counterparts. Moreover, despite the many recent advances in the field of interpretability of machine learning models (see, for example, the work in [122]), most of these models behave as *black-boxes* that are difficult to inspect and to debug when their responses do not conform with our expectations. These issues and the potential unpredictability of the responses require complex systems to constantly monitor their behaviors, which makes their integration into existing production settings non-trivial.

Nevertheless, there exist commercial solutions based, e.g., on Hierarchical Recurrent Neural Networks[5] that simplify the adoption of state-of-the-art session-based recommendation systems in a wide range of real-world applications.

3.2.1 Markov Models

Markov Models treat sequential data as a stochastic process over discrete random variables over a finite number of steps. The simplest of these models are first-order Markov Chains, which model the transition probabilities between two *subsequent* actions in a sequence. An example of using first-order Markov Chains for next-item recommendation in the context of SBRS is given in [73]. The model in this work simply relies on the count of the number of times in which an action on an item m happens immediately after an action on item l for every session in the training dataset S. In summary, given a session $s = (s_1, s_2, \dots s_{|s|})$, where $s_{|s|}$ is the last item on which the last action happened, the recommendation score for item i is computed as:

$$\text{score}_{\text{MC}}(i, s) = \frac{1}{\sum_{p \in S} \sum_{x=1}^{|p|-1} 1_{\text{EQ}}(s_{|s|}, p_x)} \sum_{p \in S} \sum_{x=1}^{|p|-1} 1_{\text{EQ}}(s_{|s|}, p_x) \cdot 1_{\text{EQ}}(i, p_{x+1}) \tag{2}$$

where the function $1_{\text{EQ}}(a, b)$ indicates whether items a and b are the same. The right-hand side counts how frequently item i immediately follows $s_{|s|}$ in the training data, while the normalization coefficient on the left ensures that scores are valid transition probabilities.

In situations with very large item catalogs, the transitions between pairs of items tend to be extremely sparse, which translates into poor estimates of the above tran-

[5] https://docs.aws.amazon.com/personalize/latest/dg/native-recipe-hrnn.html (Date visited: 2020/06/01)

sition probabilities. This issue can be alleviated by using certain heuristics, like skipping, clustering and finite mixture modeling [99], or by factorizing the transition matrix (or tensor) [91]. Markov Chains can be extended beyond first-level interactions, as the user's next action and intent might in fact depend on action previous to the last one. One option are Variable-order Markov Models (VMMs), or context-trees, which allow to consider different sequence lengths in one model [34, 42]. The work in [81] describes the offline and online evaluation of an e-commerce recommender systems based on Bayesian Variable-order Markov Modeling. Suffixes of sessions that were observed in the past are arranged into a tree-shaped structure. Given the current session, the algorithm then computes the probability that a specific item is encountered by aggregating the probabilities from the corresponding nodes in the tree.

Another option are Hidden Markov Models (HMMs), which replace item transition probabilities of plain Markov Chains with the transition probabilities between (consecutive) discrete hidden states, which represent the unobserved hidden state of the user, and a probability distribution from the hidden states to the observed actions. Both probability distributions can be learned from observational data [47].

3.2.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are real-valued hidden-state models that are especially suited for processing data of sequential nature. Similarly to HMMs, they are based on hidden states that are updated for every input in the sequence, and then used to predict the probability of the next item in the sequence. However, RNNs use complex non-linear dynamics, which can largely enhance their learning power. In their simplest variant, the hidden state h_t of the RNN is updated from the current input in the sequence s_t and the previous hidden state as follows[6]:

$$h_t = \sigma(\mathbf{W}^{hs} s_t + \mathbf{W}^{hh} h_{t-1}) \tag{3}$$

where $\mathbf{W}^{hs}$ is the matrix of conventional weights between the input and the hidden layer, $\mathbf{W}^{hh}$ is the matrix of recurrent weights between the hidden layer and itself at adjacent time steps, and $\sigma(\cdot)$ is a non-linear function like the logistic sigmoid function. The updated hidden state is then used to compute the probability of the next item in the sequence simply as follows:

$$y_t = \text{softmax}(\mathbf{W}^{yh} h_t) \tag{4}$$

where $\mathbf{W}^{yh}$ is the matrix of conventional weights between the hidden layer and the output layer. There exist several variants of RNNs, like Long-Short Term Memory networks (LSTM) and Gated Recurrent Units (GRU). A comprehensive review can be found in [69].

[6] Bias terms are omitted to enhance readability.

RNNs are a "natural" option for session-based recommendation given the intrinsic sequential nature of user actions. In fact, various RNN-based models were proposed for SBRS [114]. Many of them are extensions of GRU4Rec [45], the first RNN-based model explicitly devised for session-based recommendation. GRU4Rec models user sessions using Gated Recurrent Units (GRU) [26], which enhance the update Equation 3 by learning when and how much to update the hidden state at each step.

RNNs are generally trained by minimizing the average cross-entropy loss over all sessions $s \in S$:

$$L(S) = -\frac{1}{|S|} \sum_{s \in S} \sum_{s_t \in s} \log y_t$$

in which all actions s_t in the session are assumed to be relevant to the user. In addition to that, GRU4Rec can be trained using ranking loss functions like BPR [90], TOP1, BPR-max and TOP1-max to directly optimize the ranking of the true next action over a sample of negative ones. Training GRU4Rec using ranking losses leads to better recommendation accuracy, as highlighted in [44]. Training RNNs over large item catalogs and long sessions is computationally intensive. GRU4Rec can be trained efficiently on GPUs thanks to mini-batch negative sampling, which samples the negative items used by the loss function among the items in the current mini-batch of sessions that is being processed. GRU4Rec+[44] adds the opportunity to use an additional fixed set of negative items to further improve the output ranking quality.

Several extensions to GRU4Rec were proposed in the literature. PRNNs [46], for example, train multiple GRU4Rec instances in parallel to learn a multi-modal SBRS. Each instance learns a representation of the session from a different mode, such as item identifiers and features like product images and descriptions. PRNNs then employ different strategies to train and merge each GRU4Rec instance. Another extension is HRNN [88, 77], a *session-aware* model that employs two interconnected GRU4Rec instances to model the user behavior both at session- and user-level. After each session, the user-level GRU4Rec instance uses the hidden representation of each session-level GRU4Rec instance at the end of each session to update the user state. The updated user-level representation is then used to initialize the hidden state of the forthcoming session-level GRU4Rec instance, which from there on behaves as a traditional session-based GRU4Rec model.

Even though RNNs are a natural Neural Network architecture for session-based recommendation, they are not the only option available. Researchers have studied several other architectures like Shallow Networks, Convolutional Neural Networks (CNNs), Graph Neural Networks (GNNs) and Variational Auto-Encoders (VAE) for session-based and, more generally, for sequence-aware recommendation. See Table 2 for a non-exhaustive list of such models. As can be seen, many neural models were proposed in recent years. For a few of them, reports regarding their practical use are available as well, see e.g., [62].

Table 2 List of Neural Network based models for SBRS.

Model	Network Type	Task	Short Description
GRU4Rec [45]	RNN	Session-based	Next-item prediction for anonymous user sessions with Gated Recurrent Units, mini-batch negative sampling and ranking loss functions.
GRU4Rec+ [44]	RNN	Session-based	Enhanced GRU4Rec with additional negative sampling and alternative ranking loss functions.
PRNN [46]	RNN	Session-based	A multi-modal session-based model based on multiple GRU4Rec models trained in parallel.
HRNN [88, 77]	RNN	Session-aware	A hierarchical GRU4Rec model that learns from user's long-term and short-term (session-level) interactions.
THRNN [112]	RNN	Session-aware	An enhanced HRNN model with Temporal Point Processes that factors in the temporal distance between consecutive user sessions.
SWIWO [48]	Shallow Network	Session-based	A shallow network with a wide-in-wide-out structure.
NARM [67]	RNN+Attention	Session-based	A model that combines RNN with attention to jointly learn a global session representation and the assumed main purpose of a session.
STAMP [70]	Memory Network	Session-based	Replaces RNN with an attentive memory network that retains the short-term user interests.
CASER [106]	CNN	Sequence-based	A model that transforms session sequences into an "image" of item embeddings, then learns sequential patterns through convolutional filters.
NextItNet [121]	CNN	Sequence-based	A model that learns long-term dependencies over long user sequences through holed convolutions and residual connections.
SVAE [95]	VAE	Sequence-based	A model that encodes sequences of user actions through a recurrent Variational Auto Encoder.
SR-GNN [117]	GNN	Session-based	A model that represents session sequences as graphs. Uses a GNN to capture complex item transitions on the session graph.
TAGNN [120]	GNN	Session-based	Extends SR-GNN by conditioning the session representation on the target item using an Attention network in combination with a GNN.

3.2.3 Reinforcement Learning

All the previously described methods learn a representation of the user or session in order to maximize the chances of guessing the immediate next user action(s) correctly. However they rarely, if never, consider the *final goal* of the user session and *long-term effects* of the recommendations when deciding which action(s) to recommend next. By myopically optimizing for the immediate next actions and *rewards*, the recommender can be easily misled to trade the long-term user satisfaction for immediate engagement, with possible detrimental effects like click-baiting [49]. Moreover, the aforementioned families of recommenders operate deterministically, meaning that the recommender will always suggest the same list of items if fed with the same input sequence of user actions.

In practice, recommenders operate in environments with large and dynamic item spaces, in which items are frequently added and removed from the catalog. Think, for example, of the many newly released albums or singles that are added every day to the catalog of popular music streaming services. User preferences tend also to evolve relatively quickly over time, often influenced by external factors like community trends or by the natural evolution of personal interests.

Recommendation algorithms should hence be flexible enough to adapt their beliefs of the user interests and, consequently, their predictions, to the inherent dynamics of the item and user spaces. One strategy often used in practice is to introduce a certain degree of *exploration* into recommendation algorithms in order to gather feedback from users on the newest items, or to refine the knowledge on the current user's interests by exposing them to less *exploitative* items.

Exploration requires recommenders to take "suboptimal" actions w.r.t. the current estimates of the affinity between users and items, which indeed incurs costs in the short-term that will hopefully be offset in the mid- and long-term. Trading short-term engagement with longer-term success lies at the core of the well-known *exploration-exploitation dilemma* [68], and it is one of the guiding principles behind many Reinforcement Learning techniques.

The main goal of Reinforcement Learning (RL) is that of learning an algorithm capable of taking actions in an environment that are optimized towards a long-term goal. At each round, or *trial*, the RL algorithm, or *agent*, observes a *state* based on which it decides which *action* to take next. Actions are taken by following a *policy*, which is often defined as a stochastic function over actions. For each action taken, the agent receives a *reward* from the environment that it is used to refine the policy in a way that the long-term objective can be achieved under certain guarantees.

In the context of recommender systems, actions are often thought of as the recommendable items, rewards are the users' feedbacks on the recommended items (such as clicks, listening, add-to-cart, etc.). Long-term goals may be the purchase at the end of a session or the conversion of the listener from trial to premium subscription. The fact that policies are no longer deterministic but *stochastic* inherently allows any RL method to explore the action space more or less effectively. Here, we briefly introduce two RL methods that are frequently used in real-world recommenders: Multi-Arm Bandit (MAB) algorithms and Markov Decision Processes (MDP).

Multi-Arm Bandits (MABs) are widely used for sequential decision making. At each round, or trial, the agent selects one of *K arms* for which it receives a reward, which in the simplest case is binary (e.g., the user clicks/plays the recommended item). The goal of the agent is to maximize the sum of the rewards over time. Since the agent receives rewards only for the arms it selects, it needs to *explore* all arms to identify which are the best ones. At the same time, the selection of suboptimal arms leads to lower rewards, which induces the bandit to *exploit* arms with known rewards. A central element of any Bandit algorithm therefore lies in its arm-selection policy, which has to balance exploration and exploitation. Exploration might decrease the *short-term* recommendation quality as many suboptimal arms can be chosen initially, but in turn can improve the *long-term* quality as more information on the items' rewards is gathered. When using MABs for session-based

recommendations, each arm is modeled as an item to be recommended to the user within the session. At each trial, one item is recommended (one arm is chosen) and the algorithm receives as feedback information about whether or not the user interacted with the item. Some examples of arm-selection strategies are ϵ-greedy and Upper Confidence Bounds [16]. An often used variant of Bandit models are Contextual Bandits (CB) [68], which learn personalized arm-selection policies by conditioning the expectations of the rewards on a joint user/item feature vector, called *context*[7]. However, the contextual feature vectors at each round are assumed to be independent from the previous ones, which severely limits the applicability of CB to session-based recommendation scenarios, where the sequence of user actions in the session can be highly important to understand as user's intent or goals.

Markov Decision Processes (MDPs) improve over CB-based approaches by conditioning each action taken by the agent on the sequence of past user actions [22]. An MDP is defined as a four-tuple $(\mathcal{S}, I, \mathbf{P}, R)$, where S is a set of states, I is a set of items, $\mathbf{P} : \mathcal{S} \times I \times \mathcal{S} \rightarrow \mathbb{R}$ is a state transition probability and $R : \mathcal{S} \times I \rightarrow \mathbb{R}$ is the reward function. In an MDP we seek for a policy $\pi(i|s)$ that represents a distribution over the item to recommend $i \in I$ conditioned on the user state $s \in \mathcal{S}$ such that the expected cumulative reward obtained by the recommender systems is maximized [22, 99]. There are different families of methods to solve MDP problems, such as Q-learning [100] and Policy Gradient [66]. In a straightforward application of MDPs for session based recommendations, the set of states $\mathcal{S}$ is defined as the set of all possible sequences of items, and every element in $\mathcal{S}$ can be identified with a (possibly infinitely long) user session in terms of interacted items. At each step, the next item to recommend will be sampled from the policy π, which can be later updated based on the reward received by the user (e.g., the user playing vs. skipping the next recommended song). In SBRS, the MDP can be trained to optimize certain session-based metrics of interest, like the overall time spent listening/watching, or to maximize the chances that the user will perform certain actions throughout the session, such as to purchase one or more products.

The presence of the user state s and of the station-transition function $\mathbf{P}$ allows to condition the actions taken by the MDP-based recommender on historical user actions. For example, the authors of [99] define the user state as the sequence of the last k=*3* user interactions, and the state-transition probabilities are inferred from a k-order Markov Chain with special correction factors. More recently, thanks to the advancements in Deep Reinforcement Learning [83], the authors of [22] proposed an MDP-based recommender model in which the states and state-transition function are parameterized by an RNN with softmax output. In parallel, another RNN learns a user behavior model by predicting which action the user will most likely take next. The user behavior model is used to correct the biases introduced when learning from logged feedback data through importance weighting (see Section 4.5.2 for a further discussion on the topic).

[7] This should not to be confused with the common definition of context in recommender systems, which often refers to the set of contextual variables, such as location, time, etc., that can influence a user's decisions.

In summary, Reinforcement Learning methods can represent a powerful option in situations with rapidly evolving catalogs of items and when the long-term impact of recommendations is of interest, which makes them suitable for both session-based and session-aware recommendation scenarios. However, the design and implementation of these recommenders is non-trivial, as they, for example, can be sensitive to the definition of the state-transition and reward functions. Other challenges include, for example, the existence of large item spaces or small effect sizes [111].

4 Evaluation of Session-based Recommenders

The main types of evaluation approaches in recommender systems are *(i)* offline experiments, *(ii)* user studies, and *(iii)* field tests [98].

In academia, "offline" (simulation-based) experiments on datasets containing historical interactions between the users and the system (e.g., explicit ratings, implicit feedback) are predominant. The common offline evaluation methodology when using the matrix completion abstraction is to split the given interaction data into training and test splits, use the training data to learn a model and predict the held-out preferences based on this model. This process is typically repeated a number of times in a cross-validation procedure in order to compute confidence bounds. The relevance of the recommendations generated by an algorithm can then be assessed with the help of error, classification and ranking metrics from machine learning and information retrieval such as RMSE, Precision, Recall, MAP [43]. In addition to these relevance measures, one can analyze a number of other quality factors of the recommendations, e.g., in terms of the diversity of the list, the general popularity of the recommended items, or the algorithm's catalog coverage [96].

User (laboratory) studies are an alternative to offline experiments. Such studies are often used to assess the potential impact of recommendations on the actual choices 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 [37].

While in principle all three mentioned evaluation forms (offline studies, user studies, field tests) can be applied for SBRS, slightly different evaluation methodologies are used in offline studies.

4.1 Offline Evaluation Protocols

When conducting an offline experiment, different choices have to be made regarding *(i)* the partitioning of the dataset, *(ii)* the definition of the target interactions, and *(iii)* the metrics that are employed. These choices should be guided by the particularities of the targeted application scenario.

4.1.1 Dataset partitioning

The first step for the evaluation of an SBRS is the generation of the training and test datasets. The former is used to train the recommendation algorithm, whereas the second contains held-out data used exclusively during the evaluation. Additionally, a fraction of the data in the training dataset can be held out for tuning the hyperparameters of the recommendation algorithm (the validation set).

In standard matrix completion setups, the partitioning of the interactions into training and test sets is often done by sampling interactions randomly. Placing 80% of the interactions in the training set and 20% in the test set is a common approach. However, in session-based recommendation scenarios, where the interactions are sequentially ordered and grouped in sessions, an alternative approach is needed. For instance, one should consider the temporal ordering of the interactions and select the most recent interactions for the test set.

Two forms of splitting the data into training and test set are possible: *interaction-level* and *session-level* splitting. Session-level splitting partitions the dataset by holding out entire sessions, i.e., without breaking sessions apart. Interaction-level splitting, in contrast, acts over the individual interactions: some or all of the sessions are split into contiguous subsets that are assigned either to the training or the test set.

When splitting the data at the *interaction-level* two additional options exist. We can assign all interactions before a certain point in time to the training set and the remaining interactions to the test set, as done in [39]. With this procedure, an interaction is therefore assigned to one of the sets independent of the user or the session it might belong to. Or, we can use fixed-size *within-session* splits and place the last k interactions of each session into the test set (leave-k-out) and the remaining into the training set, as done in [25]. Similarly, one can apply a time-based splitting criterion at the *session-level*, i.e., we do not split up sessions but consider the timestamp of the first interaction within a session to decide if the session goes in the training or test set [45, 109].

Both interaction-level and session-level partitioning can be applied to either the whole set of users (*hold-out of interactions*) or to a subset of the users (*hold-out of users*). 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 then further split—either at interaction or session-level—into a *user profile* and *test data* [52]. The user profile includes the less recent interactions and is used as input to the recommender while the test data contains the most recent interactions to be predicted.

Mixed approaches are possible and often advised. One can, for example, first adopt a session-level splitting to partition the dataset into training and test sessions. Interactions within each test session are further split with a leave-k-out approach into a *session profile* and *test interactions*.

No common standard exists in the community regarding data partitioning procedures. 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 SBRS that are batch-trained on instances of past sessions

for efficiency reasons. Moreover, user-level partitioning is 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). Finally, it is advisable to use a partitioning procedure that reflects the task at hand. For instance, within-session splits are useful for pure session-based recommender systems. On the contrary, a mixed approach that combines session-level partitioning and within-session split is preferred for session-aware recommender systems.

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, e.g., because news can quickly become outdated or because e-commerce shoppers might concentrate on trending or recently added items [55]. In general, the evaluation should be performed with different time splits, in order to evaluate the learning rate of the algorithm (i.e., the minimum amount of training data that provide stable recommendations).

Using a repeated and randomized dataset partitioning for cross-validation is not always possible with SBRS. Sequence or time dependent interactions exist, which make it impossible to randomly assign interactions to the training and test splits. One possible workaround is to partition the dataset into several, potentially overlapping time slices of about the same length ("sliding window") and use the actions at the end of these time slices as test sets [52, 55].

4.1.2 Definition of target interactions

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 SBRS, we are usually interested in predicting future interactions between a user and the system, where interactions 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 includes a sequence of interactions, e.g., entire sessions from the test set and/or partial sessions from the user or session profile. If such a sequence of interactions is given, the general idea is to hide a subset or all of the interactions of the given session and predict the user's (next) interactions. Two variations of the evaluation scheme can be used. With both variants the first N interactions of the session are used as session profile and "revealed" to the algorithm.

- *Given-N next-item prediction*. With this protocol, the task of the recommender is to predict the immediate next interaction when the first N interactions of the session are revealed to the algorithm [52]. N can be incrementally increased [45, 46, 88]. In same domains, often only the last element of a sequence is hidden (i.e., N = session length-1) [41].
- *Given-N next-item prediction with look-ahead*. With this protocol, the task is to predict one of the remaining hidden interactions (which form the look-ahead set) [39, 45]. The order of the hidden interactions is often not considered relevant

in terms of the quality of the recommendation. Also in this case, N can be incrementally increased.

Additional variations of these protocols exist. One can, for example, limit the evaluation to interactions of certain types (e.g., purchase actions). Another variant is to hide only the last interaction of a session, which is used in application scenarios to assess the recommendation performance when more information is available ("warm start"). For instance, in the domain of next-track music recommendation, sometimes only the last element of a session is hidden [41].

Generally, one limitation of such "hide-and-predict" evaluation approaches lies in the assumption that there exists exactly one good recommendation. Depending on the domain, this might however be too narrow and it might therefore be reasonable to consider items other than the hidden one as good recommendations as well. This was done, for example, in [62] for the e-commerce domain. In their case, items that were nearly identical to the hidden ground truth item were treated as relevant recommendations as they represented possible shopping alternatives.

4.1.3 Evaluation Metrics

The output of an SBRS usually is a ranked list of alternatives, where the items are ordered according to the their likelihood to be the next one that the user will interact with. It is therefore possible to use standard *classification* and *ranking* metrics for performance evaluation. The set of possible metrics include Precision, Recall, Mean Average Rank (MAR), Mean Average Precision (MAP), Mean Reciprocal Rank (MRR), Normalized Discounted Cumulative Gain (NDCG) and the F1 metric. When the application domain requires recommendations to fulfill an explicit or implicit order constraint, ranking metrics (such as MAP, MRR and NDCG) are preferred over classification metrics. 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 [29]. Note that differently from many conventional recommender systems, many SBRS recommend also items that the user already knows or has purchased in the past. In these application scenarios, the evaluation protocol has to be adapted in order to consider already known items when computing the metrics.

In many 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 to generate good 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 to not take that aspect into account. This can in turn lead to the effect that more popular tracks are favored by an algorithm [53]. Therefore, alternative metrics derived from natural language processing, like the Average Log-Likelihood (ALL), can be adopted instead [23, 24]. Regarding the ALL, some concerns however exist regarding the true usefulness this measure [17].

Generally, when the goal of a recommender is not limited to predicting the next-best interaction but a sequence or set of interactions 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 the order of the recommendations or the internal coherence or diversity of the recommended list as a whole. For instance, 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 trade-off situations like "accuracy vs. diversity", which have to be balanced by a recommendation algorithm [58]. Unfortunately, no standards exist yet when it comes to evaluate recommendation lists with respect to, e.g., 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 what extent such measures reflect the users' quality perceptions sometimes remains open.

Finally, SBRS are often computationally more complex than traditional recommenders. Therefore, the evaluation of SBRS 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.

4.2 Evaluation of Reinforcement Learning Approaches

The offline evaluation of recommender systems based on traditional supervised learning algorithms assumes the independence between interactions and recommendations: a user's future interactions with items are not influenced by the recommendations provided to the user, but only by the items in the catalog.

The assumption is not valid anymore when evaluating recommender systems based on reinforcement learning, as the algorithm (agent) is updated based on both the interactions (reward) from the user (environment) at the current time stamp and the recommended items (context, or impressions) at the previous time stamp [115].

Because of the interactive nature of reinforcement learning algorithms, the best way to do an unbiased evaluation is to run the algorithm online on live data, e.g., with A/B testing. However, in practice, this approach is seldom feasible.

Therefore, the offline evaluation of session-based reinforcement learning approaches requires to simulate an online environment in which users dynamically interact with the system whenever a new set of recommendations is presented. However, creating a simulator itself might introduces bias in the results, making the evaluation unreliable [68]. The problem arises because datasets are collected by presenting items to users by using a logging policy (i.e., a recommender system, either personalized or non-personalized,) which is different from the one under test. This leads to the *off-policy policy evaluation problem* in reinforcement learning further discussed in Section 4.5.2.

One solution is to apply the *replayer* method, as described in [97, 115]: each user interaction in the historical dataset is replayed to the algorithm under evaluation. If the item recommended by the algorithm under test is identical to the one in the historical dataset, this item is considered as an impression to the user and the reward can be

computed. The replayer method is unbiased under the assumption that interactions in the dataset were collected by providing users with random recommendations. However, this assumption is unrealistic in most session-based datasets.

Biases in the results obtained with the replayer method can be removed by using off-policy estimators, such as the Inverse Propensity Score used in [37]. However, this comes at the cost of increased variance in the results.

RecoGym, presented recently in [94], is an alternative reinforcement learning simulator that can be used also for session-based recommendations. RecoGym allows to simulate sessions containing both organic interactions (i.e., interactions not affected by recommendations) and *bandit interactions* (inspired by bandit algorithms), thus allowing for a fine control of the trade-off between variance and bias in the evaluation.

4.3 Datasets

Datasets suitable for benchmarking SBRS algorithms must satisfy two requirements: *(i)* interactions must be ordered in time (eventually with an absolute time-stamp) and *(ii)* interactions must be grouped into *usage sessions*. Note that differently from the more traditional top-N recommendation task, datasets for SBRS do not necessarily need to map interactions to users.

In recent years, a number of datasets to benchmark SBRS algorithms have become available to researchers. Table 3 shows the characteristics of a (non-exhaustive) set of public datasets used in existing works.

Table 3 Examples of publicly available datasets for session-based recommendation.

Dataset	Domain	Users	Items	Interactions	Sessions	Reference
Diginetica	EC	-	134M	4,3M	574K	[8]
ACM RecSys Challenge 2015	EC	-	38K	34M	9.5M	[7]
AVITO	EC	-	4K	767K	32K	[6]
RetailRocket	EC	-	417K	2.8M	1.4M	[6]
JD	EC	810K	240K	11M	2.7M	[9]
Microsoft Web Data	Web	40K	8K	68K	37K	[1]
Delicious	Web	8.8K	3.3K	60K	45K	[3]
NYC Taxi Dataset	POI	-	299	-	670K	[4]
Art Of the Mix	Music	-	218K	-	29K	[2]
30Music	Music	40K	5.6M	31M	2.7M	[110, 5]
Spotify Sequential Skip Prediction	Music	-	3.7M	-	150M	[19, 10]

EC: E-commerce, Web: Web browsing; POI: Point of Interest.

Very few datasets (e.g., JD, Microsoft Web Data, Delicious and 30Music) associate sessions with users and, as such, can be used to evaluate both session-aware and session-based recommender systems. Most of the datasets do not associate sessions with users. For these datasets, the assumption is that each session corresponds to a unique user. These datasets therefore cannot be used to evaluate session-aware algo-

rithms. Only a few datasets, such as the Diginetica dataset, provide a partial mapping between sessions and users. Most of the datasets provide multiple types of interactions. For instance, the Diginetica dataset contains views, clicks and purchases; the 30Music datasets contains play events and ratings.

Note that in Table 3, we only list datasets where the interactions are already organized into sessions. We omit datasets in which interactions are time-ordered but not organized into sessions. For these datasets, some pre-processing logic can be applied to create sessions through heuristics, which, however, makes the comparison of different research works more challenging. Furthermore, the chosen heuristics that determine if two interactions belong to the same usage session have to be chosen with care. In the e-commerce domain, for example, we might consider two interactions with a 30-minute gap between them as belonging to different sessions. This gap might, however, be inappropriate in another application. A user of a video-on-demand service might, for example, watch a number of episodes from the same series back to back. In this case, there might be no interactions by the user for more than 30 minutes while the episode is played. Still, the interactions happening after this episode should be part of the same session.

Finally, in some research works, we observe that time-stamped user-item interaction data, like the MovieLens or Netflix datasets, are used to evaluate SBRS. These datasets are however not suitable for the task, e.g., because the time stamps do not correspond to the time of watching a movie but to the point in time when the rating was submitted, which can be hours, weeks, or years after the interaction. The prediction would therefore refer to the question which item a user will rate next, which is typically not in the focus of SBRS research.

4.4 User-centric Evaluation of Session-based Recommendations

Offline evaluations, as discussed so far, have their limitations. Importantly, they cannot inform us about the *quality perception* by users, e.g., if the recommendations were truly considered useful, if they helped to discover something new, reduced the choice difficulty, or increased the users' satisfaction with the system.

User-centric evaluation approaches in the form of controlled randomized experiments are not uncommon in the recommender systems literature in general. Studies like the one in [31] often analyze various quality factors of recommender systems in parallel and, for example, investigate the relationships between quality dimensions such as perceived commendation accuracy, novelty, or diversity. While such user studies usually seek to answer very specific research questions, certain quality dimensions are often common across these dimensions. Pu et al. [86] and Knijnenburg et al. [61] have therefore put forward two alternative general frameworks for user-centric evaluation. These frameworks are not limited to aspects regarding the recommendation algorithms themselves (e.g., regarding accuracy, diversity, and novelty), but also include questions related to the perceived quality of the human-computer interaction and the usefulness of the system as a whole.

In the context of session-based recommendation approaches, the number of user studies and field studies is still very limited. The outcomes of a field study of deploying recommendations of the style "Customers who bought . . . also bought" in the context of a reference item are reported in [64, 65]. The studies revealed, among other aspects, that purchase-based consumption patterns are far more effective than view-based patterns, or that recommender systems of that type can lead to a decrease in sales diversity over time.

Recently, differences between alternative session-based recommendation strategies were explored through user studies in [59] and [74]. In both studies, the application domain was that of music recommendations. The question analyzed in these studies is how playlist continuations, as produced by session-based recommendation algorithms, were perceived by users along different quality dimensions. Moreover, both studies tried to shed light on the question to what extent accuracy results obtained in offline experiments correlate with such quality dimensions.

In [59], 277 study participants were presented with a number of four-item playlists, and for each playlist five alternative continuations were provided. The playlists were previously created and shared by users on different music platforms and originally consisted of at least five tracks. One main task of the participants was to rank the five alternatives in terms of their suitability as a continuation. Four of the alternative continuations were computed with the help of different session-based algorithms, and one was the *true* continuation of the original playlist.

The study led to three main insights. First, the study indicated that it is meaningful to rely on user-created playlists as a "gold standard" for offline evaluations, because the true but hidden continuations were often considered as highly appropriate. Second, it turned out that a hybrid algorithm that considered music metadata was not only favorable in terms of offline accuracy, but was also considered to lead to better continuations according to the users' perceptions. This confirms that offline results were at least to some extent indicative of the users' quality perceptions. Finally, the study also revealed some possible limitations of user-centric evaluations. In particular, it turned out that participants had a tendency to consider tracks as good continuations when they already knew them before the experiment, see also [72] for a discussion of such familiarity effects.

A different study setup for evaluating session-based recommendations was used in [74]. Here, 250 participants were interacting with an online radio service, which was created for the purpose of the study. The participants were first asked to select a start track for the radio, based on which playlists were generated by different algorithms using a between-subjects design. Five different algorithms were used, including three conceptually simple ones, GRU4Rec as a representative of neural techniques, and the recommendations returned from Spotify's API. The participants could then skip or like individual tracks and answered a post-task questionnaire.

The main outcomes can be summarized as follows. First, it was found that conceptually simple methods often led to playlists that the participants liked more than those of more complex approaches. Again, an offline evaluation was also partly indicative of this result. Second, however, it was found that the choice of the optimization goal matters. The tracks played by the simple association rule method, as described above,

received many *like* statements, but the radio station itself was considered to be less attractive than others. Third, the recommendations returned from Spotify's API led to very low performance results in an offline experiment, but the radio station was nonetheless highly rated by the participants. The responses in the questionnaires indicate that the capability of Spotify's algorithm to help users discover interesting tracks contribute to the perceived quality of the recommendations.

Overall, user studies can—as the discussed examples show—help address important research questions that cannot be answered through pure offline evaluation approaches. Clearly, user studies can also have their limitations, e.g., regarding the representativeness of the participants of the study. Nonetheless, they are an important instrument that can help us understand if accuracy improvements in offline experiments really lead to better recommendations as perceived by users.

4.5 Limitations of Current Research

Current research, as mentioned above, is largely based on offline evaluations, and many papers are published each year that propose increasingly more complex methods for the recommendation task. Estimating the true value that would be achieved when using such methods in real-world environments however remains challenging for at least two reasons. First, methodological issues in applied machine learning can easily lead to progress that is actually very limited [33]. Second, the basis for our evaluations, i.e., the datasets, can be "unnatural" in a sense that what we see in the logs is affected by the circumstances and environment in which the data was collected.

4.5.1 Methodological Issues and Limitations of Academic Research

Most published papers on recommendation algorithms are applied machine learning works, where the goal is to demonstrate empirically that a new machine learning model outperforms what is considered the "state-of-the-art" on certain computational measures like Precision or Recall. One problem of this research approach is that there is no exact definition what represents the state-of-the-art (i.e., the baselines in a comparative evaluation). Furthermore, when researchers provide empirical evidence for the superiority of their method, they have some degrees of freedom with respect to the experimental configuration, e.g., regarding the choice of the datasets or the evaluation metric. Finally, another phenomenon observed in the literature is that researchers often invest much effort in fine-tuning their own new model, but do not do the same for the baselines. As a result, it can turn out that the claimed improvements "don't add up" [15], and that newer methods are actually not better than long-known techniques if these are properly tuned. Such problems, which are not new [15], can be found in the recommender systems literature [33, 92], in related areas such as

information retrieval [118], and in more distant fields such as time-series forecasting [79].

In the area of session-based recommendation, related phenomena were observed. In [55] and [73], it was for example shown that nearest neighbor techniques are often competitive or even outperform a widely-used RNN-based technique as well as other sequential recommendation approaches. In subsequent works [75, 76], twelve algorithms—six neural and six non-neural ones—were compared under identical conditions on several datasets. Again, it turned out that in the majority of the examined cases the neural methods were outperformed by conceptually simple methods. Interestingly, many of the more recent papers do not consider such simpler baselines, even though they are not only competitive in terms of performance, but also have other advantages such as the possibility to incrementally update the database.

Like argued in [33, 32] it is therefore important to consider baselines from different algorithm families, and not just methods of a certain type, e.g., deep learning based approaches, and to optimize these baselines in a systematic way. Furthermore, it was observed in [32, 33] that reproducibility of published research in recommender systems is not very high, which is also a factor that hampers progress. Therefore, to ensure progress also in session-based recommendation, it is important that researchers share the data and the software artifacts that they used in their empirical evaluations, including all pieces of code used in the evaluation pipeline, from preprocessing, over data splitting, model learning, and evaluation. A corresponding open-source framework for the evaluation of session-based recommender systems called Session-Rec[8] was made available in [75].

Finally, it is worth pointing out that the offline evaluation practices used in academia represent only a subset of the dimensions and domains on which machine learning algorithms are evaluated in practice. While offline evaluation can be helpful when benchmarking machine learning algorithms under well defined experimental conditions, it can only provide a partial view of their characteristics. In particular, it is not always clear if the obtained results generalize beyond the controlled environment that was used in the evaluation. Session-based recommenders are no exception to this rule. Besides a potential offline-online mismatch discussed in Section 4.4, the different families of session-based recommenders described above exploit certain types of *inductive biases* in the data, like any machine learning algorithm. Examples of inductive biases are the sequentiality of user actions which is exploited by RNN-based models, or the notion of similarity between user sessions in nearest-neighbor techniques. The effectiveness of each method is ultimately strongly based on the presence of such inductive biases in the scenario (dataset) under study [82]. It is therefore advisable to test different families of algorithms in order to truly understand what method works best for the task at hand.

[8] https://github.com/rn5l/session-rec

4.5.2 Potential Biases in Datasets

More and more datasets are nowadays available for conducting offline experiments, as discussed in Section 4. However, often little is known about the environment and circumstances in which the data was collected. The specific circumstances can, however, have a significant impact on what we observe in the logs. For example, when data is collected on an e-commerce site, at least some interactions may be the result of an advertisement campaign or the effect of a temporary discount [113]. Some clicks or purchase actions may also be triggered by an already existing recommendation system. Likewise, in the music domain, the listening logs that are obtained from Last.fm [5], might be the result of what was played by an automated radio station.

An analysis of an e-commerce log dataset in [56] revealed that about every hundredth view action in the logs originated from a click on a recommendation. More importantly, however, such clicks on recommendations very often led to purchases afterwards, and these purchases were often related to items that were currently discounted or generally trending on the store.

As a result, when models are trained on such datasets, they will eventually reflect these biases, and we might for example end up in re-constructing—at least to a certain extent—the logic of a recommender system or effects of a marketing campaign that took place during the data collection period. Therefore, the trained models might be not as effective as expected when they are deployed in practice.

In recent years, different approaches were put forward to deal with such problems. Possible countermeasures include the use of evaluation metrics that take such biases into account or mechanisms that aim to "de-bias" the data [21, 102, 113]. To end up with more realistic offline evaluations, i.e., evaluations that are more predictive of the online success than current evaluation procedures, different proposals were also made in the context of *off-policy learning* in reinforcement learning and bandit-based recommendation approaches that aim at answering to the counterfactual question "How would the system have performed under a different recommender?". The main idea is that of learning a model from the logged feedback and using it to correct the data biases when training the new model. One of the mostly used techniques is Inverse Propensity Scoring (IPS), which assigns a probability to every user action (u, i) under the logging policy π_0, i.e., the unknown policy that generated the data, and re-weights the reward of each action taken by the policy π of the recommender system with weight $w(i|u) = \frac{\pi(i|u)}{\pi_0(i|u)}$. It can be proven that, under reasonable assumptions, the weighting function leads to an unbiased estimator over the biased data [18].

Researchers have extended IPS in several ways, by e.g., computing better estimates of the weights in conditions of high variance through normalization and capping [103], or by applying it to complex recommendation scenarios like slate recommendation [104]. More recently off-policy correction was also employed to correct biases in sequences of user actions [22, 71].

5 Future Directions

Despite the increased research interest in SBRS in recent years, a number of questions and challenges remain open, in particular ones that go beyond algorithmic questions.

5.1 Algorithmic Improvements—Considering More Types of Data

In terms of algorithmic approaches, we identify the following main areas where more research is required: *(i)* the integration of long-term preferences, *(ii)* the inclusion of item meta-data and context, and, *(iii)* the consideration of temporal aspects.

The majority of today's algorithmic proposals focus on situations, where no long-term preference information about the current user is available. While it is often much more important to quickly adapt to the user's *current* interests [52], considering the user's past interests can be beneficial as well. The number of corresponding proposals to build such *session-aware* recommender systems is however still comparably small, and one of the main challenges is to decide how strongly past sessions should be considered in general or in a given session. Some recent model-based and neural approaches are, for example, purely based on collaborative information and the observed user behavior such as purchases or item views [88, 85]; see [63] for a performance comparison. Other approaches in addition try to leverage additional information in the recommendation process. This can both include information about the items [52] and other types of knowledge, e.g., about the social network of the users [51].

The use of *side information* in general appears to be a promising research direction, independent of the question if the long-term preferences are considered or not. In some ways, while numerous model-based approaches were proposed in recent years, the progress that is achieved with pure collaborative models can appear limited [75]. A few methods exist that incorporate side information to build a hybrid system, see e.g., [46, 30] for neural approaches or [108] for an earlier approach based on Markov models. In particular deep learning based techniques seem promising here due to their ability to learn complex relationships and interactions in heterogeneous data. A number of opportunities exist in this context, in particular when new datasets become available that contain detailed item information, e.g., in the news domain [30], or when additional information about the users' behavior can be used, e.g., in terms of their navigation or search activities, see [27].

Explicit knowledge about the user's *context*, e.g., the current time or weather, is another type of information that has not been considered much so far in session-based recommendation. Ample evidence exists that considering the user's current context can be essential to create meaningful recommendations. To our knowledge, [101] is the only method that exploits contextual information for general sequence-based recommendation based on RNNs. Context information can be relevant both in *session-based* and *session-aware* scenarios. In session-based recommendation, where there is little known about the user at the beginning of the session, context

information like the time of the day or what is currently trending in a certain geographical area can be helpful starting points to deal with the extreme cold-start situation. In session-aware scenarios, context information can be helpful to better guess the current user's intent and to correspondingly base the recommendations on past sessions of the user that match this assumed intent. In the domain of music recommendation for example, users might sometimes be open to explore something new, while on other days they might prefer to listen to their favorite tracks from the past [60]. Here, context information therefore might help to decide on which types of items the recommender system should focus on.

To what extent such past interests are relevant in a given situation can also depend on other *temporal aspects* [87, 56]. In some domains, we might, for example, observe an interest drift over time, and users might not be interested in things anymore they liked in the past. In other domains, like for the recommendation of consumables, we can often observe repeat purchases. In this context, recommender systems can act as reminders, and a crucial question in such a setting relates to the best point in time to remind the user. Finally, in certain domains it might be helpful to consider seasonal aspects in particular when long-term preferences are available which are not relevant in the current season and which therefore should not be considered.

5.2 Improving the Evaluation Methodology

Given the mentioned limitations of today's research practice, a change in terms of how we evaluate session-based recommender systems is needed. Such extensions of our research practice should include *(i)* better reproducibility of research results and progress, *(ii)* a stronger focus on the user perception and impact of session-based recommendations, and *(iii)* the consideration of alternative offline procedures that consider the perspective of multiple stakeholders and potential biases in the data.

Regarding *reproducibility and progress* in recommender systems research, some recent works point out in the area of traditional "top-n" reproducibility is actually not high, with less than 40% of papers published at top-level conferences being reproducible "*with reasonable effort*" [33]. Moreover, almost all of these reproducible works were not to be better in terms of prediction accuracy than long-known and much simpler methods. A similar phenomenon can be observed in the literature on session-based recommendation, where many papers report progress over the state-of-the-art, but these improvements do not add up [75]. Future research that is based on offline experiments therefore must follow more rigid rules regarding reproducibility and the choice and optimization of the baselines.

In terms of the evaluation approach, more research is required regarding the *user perception* of session-based recommender systems and the *impact* that different algorithms have on user behavior. User studies as the ones described in Section 4 help us learn about user perceptions and intentions and they can help us contrast offline experimental results with these observations. Ultimately, however, such studies to some extent remain proxies to assess how recommender systems would impact users

"in the wild". There is no doubt that recommender systems in general can generate substantial business value for providers [38, 49], but only limited academic research so far exists that explores the effects of session-based recommendations on user behavior, e.g., in terms of click-through-rates as in [35].

Clearly, it is typically not possible for academic researchers to test their proposals in the real world through A/B tests. The works reported in [28] and [35], as well as other previous studies, indicate that there is a gap between offline and online performance. Often, the algorithms that work well according to computational metrics in an offline setting, do not lead to the best results in terms of the business value in reality. We are therefore in need of improved offline evaluation strategies and metrics, in particular ones that *(i)* are better suited to predict the online success of different algorithms [80], *(ii)* consider the value perspective of multiple stakeholders [11], and *(iii)* look at longitudinal effects of recommender [123].

References

1. Microsoft Anonymous Web Data. http://kdd.ics.uci.edu/databases/msweb/msweb.html, 1998. Last accessed: 15-May-2020.
2. Art of the Mix. http://www.ee.columbia.edu/~dpwe/research/musicsim/, 2004. Last accessed: 15-May-2020.
3. Delicious. http://www.dai-labor.de/en/competence_centers/irml/datasets/, 2008. Last accessed: 15-May-2020.
4. NYC Taxi Trips. http://www.andresmh.com/nyctaxitrips/, 2013. Last accessed: 15-May-2020.
5. 30Music Listening and Playlists Dataset. http://recsys.deib.polimi.it/datasets/, 2015. Last accessed: 15-May-2020.
6. Avito Context Ad Clicks. https://www.kaggle.com/c/avito-context-ad-clicks/, 2015. Last accessed: 15-May-2020.
7. RecSys Challenge 2015. http://2015.recsyschallenge.com/challenge.html, 2015. Last accessed: 15-May-2020.
8. Diginetica CIKM Cup 2016 Dataset. https://competitions.codalab.org/competitions/11161, 2016. Last accessed: 15-June-2020.
9. JD Dataset. https://github.com/alicogintel/SDM, 2019. Last accessed: 15-May-2020.
10. The Music Streaming Sessions Dataset. https://www.aicrowd.com/challenges/spotify-sequential-skip-prediction-challenge-old, 2019. Last accessed: 10-June-2020.
11. H. Abdollahpouri, G. Adomavicius, R. Burke, I. Guy, D. Jannach, T. Kamishima, J. Krasnodebski, and L. Pizzato. Multistakeholder recommendation: Survey and research directions. *User Modeling and User-Adapted Interaction*, 30:127–158, 2020.
12. G. Adomavicius, B. Mobasher, F. Ricci, and A. Tuzhilin. Context-aware recommender systems. *AI Magazine*, 32(3):67–80, 2011.
13. R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, 1993.
14. R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings International Conference on Data Engineering, ICDE '95*, pages 3–14, 1995.
15. T. G. Armstrong, A. Moffat, W. Webber, and J. Zobel. Improvements that don't add up: Ad-hoc retrieval results since 1998. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 601–610, 2009.

16. P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002.
17. G. Bonnin and D. Jannach. Automated generation of music playlists: Survey and experiments. *ACM Computing Surveys*, 47(2):26:1–26:35, 2014.
18. L. Bottou, J. Peters, J. Quiñonero-Candela, D. X. Charles, D. M. Chickering, E. Portugaly, D. Ray, P. Simard, and E. Snelson. Counterfactual reasoning and learning systems: The example of computational advertising. *The Journal of Machine Learning Research*, 14(1):3207–3260, 2013.
19. B. Brost, R. Mehrotra, and T. Jehan. The music streaming sessions dataset. In *Proceedings of the TheWebConf*, page 2594–2600, 2019.
20. P. G. Campos, F. Díez, and I. Cantador. Time-aware recommender systems: A comprehensive survey and analysis of existing evaluation protocols. *User Modeling and User-Adapted Interaction*, 24(1–2):67–119, Feb. 2014.
21. D. Carraro and D. Bridge. Debiased offline evaluation of recommender systems: A weighted-sampling approach (extended abstract). In *Proceedings of the ACM RecSys 2019 Workshop on Reinforcement and Robust Estimators for Recommendation (REVEAL '19)*, 2019.
22. M. Chen, A. Beutel, P. Covington, S. Jain, F. Belletti, and E. H. Chi. Top-k off-policy correction for a reinforce recommender system. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM '19*, pages 456–464, 2019.
23. S. Chen, J. L. Moore, D. Turnbull, and T. Joachims. Playlist prediction via metric embedding. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 714–722, 2012.
24. S. Chen, J. Xu, and T. Joachims. Multi-space probabilistic sequence modeling. In *Proceedings ACM SIGKDD International Conference on Knowledge Discovery, KDD '13*, pages 865–873, 2013.
25. C. Cheng, H. Yang, M. R. Lyu, and I. King. Where you like to go next: Successive point-of-interest recommendation. In *Proceedings International Joint Conference on Artificial Intelligence, IJCAI '13*, pages 2605–2611, 2013.
26. K. Cho, B. van Merriënboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceeedindgs of Empirical Methods in Natural Language Processing, EMNLP '14*, pages 1724–1734, Oct. 2014.
27. CIKM. The CIKM Cup 2016. `https://competitions.codalab.org/competitions/11161`, 2016. Accessed: March 2020.
28. P. Cremonesi, F. Garzotto, and R. Turrin. Investigating the persuasion potential of recommender systems from a quality perspective: An empirical study. *Transactions on Interactive Intelligent Systems*, 2(2):1–41, 2012.
29. P. Cremonesi, Y. Koren, and R. Turrin. Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings ACM Conference on Recommender Systems, RecSys '10*, pages 39–46, 2010.
30. G. de Souza Pereira Moreira, D. Jannach, and A. M. da Cunha. Contextual hybrid session-based news recommendation with recurrent neural networks. *IEEE Access*, 7, 2019.
31. M. D. Ekstrand, F. M. Harper, M. C. Willemsen, and J. A. Konstan. User perception of differences in recommender algorithms. In *Proceedings of the 2014 ACM Conference on Recommender Systems*, RecSys '14, pages 161–168, 2014.
32. M. Ferrari Dacrema, S. Boglio, P. Cremonesi, and D. Jannach. A troubling analysis of reproducibility and progress in recommender systems research. *ACM Transactions on Information Systems*, 39:1–49, 2021.
33. M. Ferrari Dacrema, P. Cremonesi, and D. Jannach. Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, pages 101–109, 2019.
34. F. Garcin, C. Dimitrakakis, and B. Faltings. Personalized news recommendation with context trees. In *Proceedings of the ACM Confererence on Recommender Systems, RecSys '13*, pages 105–112, 2013.

35. F. Garcin, B. Faltings, O. Donatsch, A. Alazzawi, C. Bruttin, and A. Huber. Offline and online evaluation of news recommender systems at swissinfo.ch. In *Proceedings of the ACM Confererence on Recommender Systems, RecSys '14*, pages 169–176, 2014.
36. D. Garg, P. Gupta, P. Malhotra, L. Vig, and G. Shroff. Sequence and time aware neighborhood for session-based recommendations: Stan. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '19, pages 1069–1072, 2019.
37. A. Gilotte, C. Calauzènes, T. Nedelec, A. Abraham, and S. Dollé. Offline a/b testing for recommender systems. In *Proceedings of the ACM International Conference on Web Search and Data Mining, WSDM '18*, pages 198–206, 2018.
38. C. A. Gomez-Uribe and N. Hunt. The Netflix recommender system: Algorithms, business value, and innovation. *Transactions on Management Information Systems*, 6(4):13:1–13:19, 2015.
39. M. Grbovic, V. Radosavljevic, N. Djuric, N. Bhamidipati, J. Savla, V. Bhagwan, and D. Sharp. E-commerce in your inbox: Product recommendations at scale. In *Proceedings ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '15*, pages 1809–1818, 2015.
40. X. Guo, C. Shi, and C. Liu. Intention modeling from ordered and unordered facets for sequential recommendation. In *Proceedings of The Web Conference 2020*, WWW '20, page 1127–1137, New York, NY, USA, 2020.
41. N. Hariri, B. Mobasher, and R. Burke. Context-aware music recommendation based on latent topic sequential patterns. In *Proceedings of the Sixth ACM Conference on Recommender Systems*, RecSys '12, pages 131–131, 2012.
42. Q. He, D. Jiang, Z. Liao, S. C. H. Hoi, K. Chang, E.-P. Lim, and H. Li. Web query recommendation via sequential query prediction. In *Proceedings International Conference on Data Engineering, ICDE '09*, pages 1443–1454, 2009.
43. J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, 2004.
44. B. Hidasi and A. Karatzoglou. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, pages 843–852, 2018.
45. B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk. Session-based recommendations with recurrent neural networks. In *Proceedings International Conference on Learning Representatinos, ICLR '16*, 2016.
46. B. Hidasi, M. Quadrana, A. Karatzoglou, and D. Tikk. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *Proceedings ACM Conference on Recommender Systems, RecSys '16*, pages 241–248, 2016.
47. M. Hosseinzadeh Aghdam, N. Hariri, B. Mobasher, and R. Burke. Adapting recommendations to contextual changes using hierarchical hidden markov models. In *Proceedings of the 9th ACM Conference on Recommender Systems, RecSys '15*, pages 241–244, 2015.
48. L. Hu, L. Cao, S. Wang, G. Xu, J. Cao, and Z. Gu. Diversifying personalized recommendation with user-session context. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 1858–1864, 2017.
49. D. Jannach and M. Jugovac. Measuring the business value of recommender systems. *ACM Transactions on Management Information Systems*, 10(4), 2019.
50. D. Jannach, M. Jugovac, and L. Lerche. Supporting the design of machine learning workflows with a recommendation system. *ACM Transactions on Interactive Intelligent Systems*, 6(1), 2016.
51. D. Jannach, I. Kamehkhosh, and L. Lerche. Leveraging multi-dimensional user models for personalized next-track music recommendation. In *Proceedings of the ACM Symposium on Applied Computing, ACM SAC 2017*, 2017.
52. D. Jannach, L. Lerche, and M. Jugovac. Adaptation and evaluation of recommendations for short-term shopping goals. In *Proceedings of the ACM Conference on Recommender Systems, RecSys '15*, pages 211–218, 2015.

53. D. Jannach, L. Lerche, I. Kamehkhosh, and M. Jugovac. What recommenders recommend: an analysis of recommendation biases and possible countermeasures. *User Modeling and User-Adapted Interaction*, 25(5):427–491, 2015.
54. D. Jannach, L. Lerche, and M. Zanker. Recommending based on implicit feedback. In P. Brusilovsky and D. He, editors, *Social Information Access*. Springer, 2018.
55. D. Jannach and M. Ludewig. When recurrent neural networks meet the neighborhood for session-based recommendation. In *Proceedings of the 11th ACM Conference on Recommender Systems*, RecSys '17, pages 306–310, 2017.
56. D. Jannach, M. Ludewig, and L. Lerche. Session-based item recommendation in e-commerce: On short-term intents, reminders, trends, and discounts. *User-Modeling and User-Adapted Interaction*, 27(3–5):351–392, 2017.
57. D. Jannach and M. Zanker. Collaborative filtering: Matrix completion and session-based recommendation tasks. In S. Berkovsky, I. Cantador, and D. Tikk, editors, *Collaborative Recommendations: Algorithms, Practical Challenges and Applications*, pages 1–38. World Scientific, 2019.
58. M. Jugovac, D. Jannach, and L. Lerche. Efficient optimization of multiple recommendation quality factors according to individual user tendencies. *Expert Systems With Applications*, 81:321–331, 2017.
59. I. Kamehkhosh and D. Jannach. User perception of next-track music recommendations. In *Proceedings of the 2017 Conference on User Modeling Adaptation and Personalization*, UMAP '17, pages 113–121, 2017.
60. K. Kapoor, V. Kumar, L. Terveen, J. A. Konstan, and P. Schrater. "I like to Explore Sometimes": Adapting to Dynamic User Novelty Preferences. In *Proceedings of the 9th ACM Conference on Recommender Systems*, page 19–26, 2015.
61. B. P. Knijnenburg, M. C. Willemsen, Z. Gantner, H. Soncu, and C. Newell. Explaining the user experience of recommender systems. *User Modeling and User-Adapted Interaction*, 22:441–504, 2012.
62. P. Kouki, I. Fountalis, N. Vasiloglou, X. Cui, E. Liberty, and K. Al Jadda. From the lab to production: A case study of session-based recommendations in the home-improvement domain. In *Fourteenth ACM Conference on Recommender Systems*, RecSys '20, page 140–149, 2020.
63. S. Latifi, N. Mauro, and D. Jannach. Session-aware recommendation: A surprising quest for the state-of-the-art. *Information Sciences*, 573:291–315, 2021.
64. D. Lee and K. Hosanagar. Impact of recommender systems on sales volume and diversity. In *Proceedings of the International Conference on Information Systems, ICIS 2014*, 2014.
65. D. Lee and K. Hosanagar. How Do Recommender Systems Affect Sales Diversity? A Cross-Category Investigation via Randomized Field Experiment. *Information Systems Research*, 30(1):239–259, 2019.
66. S. Levine and V. Koltun. Guided policy search. In *International Conference on Machine Learning*, pages 1–9, 2013.
67. J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 1419–1428, 2017.
68. L. Li, W. Chu, J. Langford, and X. Wang. Unbiased offline evaluation of contextual-bandit-based news article recommendation algorithms. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 297–306, 2011.
69. Z. C. Lipton, J. Berkowitz, and C. Elkan. A critical review of recurrent neural networks for sequence learning. *CoRR*, 1506.00019, 2015.
70. Q. Liu, Y. Zeng, R. Mokhosi, and H. Zhang. STAMP: short-term attention/memory priority model for session-based recommendation. In *Proceedings ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '18*, pages 1831–1839, 2018.
71. Y. Liu, A. Swaminathan, A. Agarwal, and E. Brunskill. Off-policy policy gradient with state distribution correction. *arXiv preprint arXiv:1904.08473*, 2019.

72. B. Loepp, T. Donkers, T. Kleemann, and J. Ziegler. Impact of item consumption on assessment of recommendations in user studies. In *Proceedings of the 12th ACM Conference on Recommender Systems, RecSys '18*, pages 49–53, 2018.
73. M. Ludewig and D. Jannach. Evaluation of session-based recommendation algorithms. *User-Modeling and User-Adapted Interaction*, 28(4–5):331–390, 2018.
74. M. Ludewig and D. Jannach. User-centric evaluation of session-based recommendations for an automated radio station. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, 2019.
75. M. Ludewig, N. Mauro, S. Latifi, and D. Jannach. Performance comparison of neural and non-neural approaches to session-based recommendation. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, pages 462–466, 2019.
76. M. Ludewig, N. Mauro, S. Latifi, and D. Jannach. Empirical analysis of session-based recommendation algorithms. *User Modeling and User-Adapted Interaction*, 31, 2021.
77. Y. Ma, B. M. Narayanaswamy, H. Lin, and H. Ding. Temporal-contextual recommendation in real-time. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 2291–2299, 2020.
78. N. R. Mabroukeh and C. I. Ezeife. A taxonomy of sequential pattern mining algorithms. *ACM Computing Surveys (CSUR)*, 43(1):1–41, 2010.
79. S. Makridakis, E. Spiliotis, and V. Assimakopoulos. Statistical and machine learning forecasting methods: Concerns and ways forward. *PloS one*, 13(3), 2018.
80. A. Maksai, F. Garcin, and B. Faltings. Predicting online performance of news recommender systems through richer evaluation metrics. In *Proceedings of the ACM Confererence on Recommender Systems, RecSys '15*, pages 179–186, 2015.
81. S. Martin, B. Faltings, and V. Schickel. Context-tree recommendation vs matrix-factorization: Algorithm selection and live users evaluation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9534–9540, 2019.
82. T. M. Mitchell. *The need for biases in learning generalizations*. Department of Computer Science, Laboratory for Computer Science Research, 1980.
83. V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.
84. B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Using sequential and non-sequential patterns in predictive web usage mining tasks. In *Proceedings of IEEE International Conference on Data Mining*, ICDM '02, pages 669–672, 2002.
85. T. M. Phuong, T. C. Thanh, and N. X. Bach. Neural session-aware recommendation. *IEEE Access*, 7, 2019.
86. P. Pu, L. Chen, and R. Hu. A user-centric evaluation framework for recommender systems. In *Proceedings of the 5th ACM Conference on Recommender Systems*, RecSys '11, pages 157–164, 2011.
87. M. Quadrana, P. Cremonesi, and D. Jannach. Sequence-aware recommender systems. *ACM Computing Surveys*, 54:1–36, 2018.
88. M. Quadrana, A. Karatzoglou, B. Hidasi, and P. Cremonesi. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *Proceedings of the ACM Conference on Recommender Systmes, RecSys '17*, 2017.
89. R. Ragno, C. J. C. Burges, and C. Herley. Inferring similarity between music objects with application to playlist generation. In *Proceedings of the 7th ACM SIGMM International Workshop on Multimedia Information Retrieval*, MIR '05, pages 73–80, 2005.
90. S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence, UAI '09*, pages 452–461, 2009.
91. S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the World Wide Web Conference, WWW '10*, pages 811–820, 2010.
92. S. Rendle, L. Zhang, and Y. Koren. On the difficulty of evaluating baselines: A study on recommender systems. *CoRR*, abs/1905.01395, 2019.

93. P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, CSCW '94, pages 175–186, 1994.
94. D. Rohde, S. Bonner, T. Dunlop, F. Vasile, and A. Karatzoglou. Recogym: A reinforcement learning environment for the problem of product recommendation in online advertising. *arXiv preprint arXiv:1808.00720*, 2018.
95. N. Sachdeva, G. Manco, E. Ritacco, and V. Pudi. Sequential variational autoencoders for collaborative filtering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, WSDM '19, page 600–608, 2019.
96. A. Said, D. Tikk, K. Stumpf, Y. Shi, M. A. Larson, and P. Cremonesi. Recommender Systems Evaluation: A 3D Benchmark. In *RUE Workshop at ACM RecSys 2012*, pages 21–23, 2012.
97. J. Sanz-Cruzado, P. Castells, and E. López. A simple multi-armed nearest-neighbor bandit for interactive recommendation. In *Proceedings of the 13th ACM Conference on Recommender Systems, RecSys '19*, pages 358–362, 2019.
98. G. Shani and A. Gunawardana. Evaluating recommendation systems. In *Recommender Systems Handbook*, pages 257–297. Springer, 2011.
99. G. Shani, D. Heckerman, and R. I. Brafman. An MDP-based recommender system. *The Journal of Machine Learning Research*, 6:1265–1295, 2005.
100. D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.
101. E. Smirnova and F. Vasile. Contextual sequence modeling for recommendation with recurrent neural networks. *CoRR*, abs/1706.07684, 2017.
102. H. Steck. Training and testing of recommender systems on data missing not at random. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, pages 713–722, 2010.
103. A. Swaminathan and T. Joachims. The self-normalized estimator for counterfactual learning. In *Advances in Neural Information Processing Systems, NIPS '15*, pages 3231–3239, 2015.
104. A. Swaminathan, A. Krishnamurthy, A. Agarwal, M. Dudik, J. Langford, D. Jose, and I. Zitouni. Off-policy evaluation for slate recommendation. In *Advances in Neural Information Processing Systems*, pages 3632–3642, 2017.
105. P. Symeonidis, L. Kirjackaja, and M. Zanker. Session-aware news recommendations using random walks on time-evolving heterogeneous information networks. *User Modeling and User-Adapted Interaction*, pages 1–29, 2020.
106. J. Tang and K. Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, page 565–573, 2018.
107. M. M. Tanjim, C. Su, E. Benjamin, D. Hu, L. Hong, and J. McAuley. Attentive sequential models of latent intent for next item recommendation. In *Proceedings of The Web Conference 2020*, WWW '20, page 2528–2534, 2020.
108. M. Tavakol and U. Brefeld. Factored MDPs for detecting topics of user sessions. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, pages 33–40, 2014.
109. R. Turrin, A. Condorelli, R. Pagano, M. Quadrana, and P. Cremonesi. Large scale music recommendation. In *Proceedings of the LSRS workshop at ACM RecSys 2015*, 2015.
110. R. Turrin, M. Quadrana, A. Condorelli, R. Pagano, and P. Cremonesi. 30Music Listening and Playlists Dataset. In *ACM RecSys 2015 Posters*, 2015.
111. F. Vasile, D. Rohde, O. Jeunen, and A. Benhalloum. A gentle introduction to recommendation as counterfactual policy learning. In *Proceedings of the 28th ACM Conference on User Modeling, Adaptation and Personalization*, UMAP '20, page 392–393, 2020.
112. B. Vassøy, M. Ruocco, E. de Souza da Silva, and E. Aune. Time is of the essence: A joint hierarchical rnn and point process model for time and item predictions. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, WSDM '19, page 591–599, 2019.

113. M. Wan, J. Ni, R. Misra, and J. McAuley. Addressing marketing bias in product recommendations. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, WSDM '20, page 618–626, 2020.
114. M. Wang, P. Ren, L. Mei, Z. Chen, J. Ma, and M. de Rijke. A collaborative session-based recommendation approach with parallel memory modules. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '19, pages 345–354, 2019.
115. Q. Wang, C. Zeng, W. Zhou, T. Li, S. S. Iyengar, L. Shwartz, and G. Y. Grabarnik. Online interactive collaborative filtering using multi-armed bandit with dependent arms. *IEEE Transactions on Knowledge and Data Engineering*, 31(8):1569–1580, 2018.
116. S. Wang, L. Cao, and Y. Wang. A survey on session-based recommender systems. *CoRR*, abs/1902.04864, 2019.
117. S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan. Session-based recommendation with graph neural networks. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence, AAAI*, pages 346–353, 2019.
118. W. Yang, K. Lu, P. Yang, and J. Lin. Critically examining the neural hype: Weak baselines and the additivity of effectiveness gains from neural ranking models. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '19, pages 1129–1132, 2019.
119. G.-E. Yap, X.-L. Li, and P. S. Yu. Effective next-items recommendation via personalized sequential pattern mining. In *Proceedings International Conference on Database Systems for Advanced Applications, DASFAA '12*, pages 48–64, 2012.
120. F. Yu, Y. Zhu, Q. Liu, S. Wu, L. Wang, and T. Tan. TAGNN: Target Attentive Graph Neural Networks for Session-based Recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '20*, 2020.
121. F. Yuan, A. Karatzoglou, I. Arapakis, J. M. Jose, and X. He. A simple convolutional generative network for next item recommendation. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining*, WSDM '19, pages 582–590, 2019.
122. M. Zaheer, A. Ahmed, Y. Wang, D. Silva, M. Najork, Y. Wu, S. Sanan, and S. Chatterjee. Uncovering hidden structure in sequence data via threading recurrent models. In *Proceedings ACM International Conference on Web Search and Data Mining, WSDM '19*, page 186–194, 2019.
123. J. Zhang, G. Adomavicius, A. Gupta, and W. Ketter. Consumption and performance: Understanding longitudinal dynamics of recommender systems via an agent-based simulation framework. *Information Systems Research*, 31:76–101, 2020.