Session-aware Recommendation: A Surprising Quest for the State-of-the-art

Sara Latifi[a], Noemi Mauro[b], Dietmar Jannach[a]

[a] University of Klagenfurt, Austria
[b] University of Torino, Italy

Abstract

Recommender systems are designed to help users in situations of information overload. In recent years we observed increased interest in session-based recommendation scenarios, where the problem is to make item suggestions to users based only on interactions observed in an ongoing session, e.g., on an e-commerce site. However, in cases where interactions from previous user sessions are also available, the recommendations can be personalized according to the users' long-term preferences, a process called *session-aware* recommendation. Today, research in this area is scattered, and many works only compare a newly proposed *session-aware* with existing *session-based* models. This makes it challenging to understand what represents the state-of-the-art. To close this research gap, we benchmarked recent session-aware algorithms against each other and against a number of session-based recommendation algorithms along with heuristic extensions thereof. Our comparison, to some surprise, revealed that *(i)* simple techniques based on nearest neighbors consistently outperform recent neural techniques and that *(ii)* session-aware models were mostly not better than approaches that do *not* use long-term preference information. Our work therefore points to potential

Email addresses: sara.latifi@aau.at (Sara Latifi), noemi.mauro@unito.it (Noemi Mauro), dietmar.Jannach@aau.at (Dietmar Jannach)

methodological issues where new methods are compared to weak baselines, and it also indicates that there remains a huge potential for more sophisticated session-aware recommendation algorithms.

Keywords:
Session-aware Recommendation, Evaluation, Reproducibility

1. Introduction

Recommender systems (RS) can nowadays be found on many modern e-commerce and media streaming sites, where they help users find items of interest in situations of information overload. One reason for the success of RS lies in their ability to personalize the item suggestions based on the preferences and observed past behavior of the individual users. Historically, researchers have therefore strongly focused on situations where only information about long-term user preferences is available, e.g., in the form of item ratings. Only in recent years, more focus was put on the problem of *session-based* recommendation, where the system has to deal with anonymous users and therefore can base its recommendations only on a small number of interactions that are observed in the ongoing session.

Due to the practical relevance of this problem, a variety of technical approaches to session-based recommendation were proposed in the past few years, in particular ones based on deep learning (neural) techniques, see [28, 47]. Implicitly, these methods try to make recommendations by guessing the user's short-term intent or situational context only from the currently observed interactions. However, while it is well known that the current intents and context may strongly determine which items are relevant in the given situation [18], information about long-term preferences of users, if available, should not be ignored. In particular, the consideration of such information allows us to make session-based recommendations that are *personalized* according to long-term preferences, a process which is also called *session-aware* recommendation [35].

Session-aware recommendation problems are recently receiving increased interest. Today, the research literature is however still scattered, which makes it difficult to understand what represents the state-of-the-art in this area. One particular problem in that context is that existing works do not use a consistent set of baseline algorithms in their performance comparisons. Some works, for example, mainly compare session-*aware* models with session-*based* ones, i.e., with algorithms that do not consider long-term preference information, e.g., GRU4REC [15]. Several other works use *sequential* recommendation algorithms, e.g., FOSSIL [13], as a baseline. These are algorithms that consider the sequence of the events but are usually designed for settings where the input is a time-stamped user-item rating matrix and not a sequential log of observed interactions. Only in a few works, previous session-aware algorithms are considered in the evaluations. One example is the method by Phuong et al. [34], which uses the HGRU4REC method [36] as a baseline. Finally, almost all works include some trivial baselines, e.g., the recommendation of popular items in a session.

With this work, our goal is to close the research gap regarding the state-of-the-art in session-aware recommendation. For this purpose, we have conducted extensive experimental evaluations in which we compared five recent neural models for session-*aware* recommendation with *(i)* a set of existing neural and non-neural approaches to session-*based* recommendation, and *(ii)* heuristic extensions of the session-based techniques that, e.g., make use of reminders [20] or consider interactions that were observed immediately before the ongoing session. Regarding the baseline techniques, we in particular considered methods based on nearest neighbors techniques, which previously proved to be very competitive in session-based recommendation scenarios [29]. All investigated techniques were compared by extending the evaluation framework shared in [27]. For reproducibility purposes, we share all data and code used in the experiments online[1], including the code for data preprocessing, hyperparameter optimization, and measuring.

The results of our investigations are more than surprising. In the majority of cases, and on all four considered datasets, heuristic extensions of existing *session-based* algorithms were the best-performing techniques. In many cases, even plain session-based techniques, and in particular ones based on nearest-neighbor techniques, outperform recent session-aware models even though they do not consider the available long-term preference information for personalization. With our work, we therefore provide new baselines that should be considered in future works on session-aware recommendation. On a more general level, our observations also point to potential methodological issues, where new models are compared to baselines that are either not properly optimized, that do not leverage all available information, or that are rather weak for the given task. Similar observations were previously made in the field of information retrieval and in other areas of recommender systems

[1] `https://www.dropbox.com/sh/gp1xnm4rsn777fp/AABxfe_Q7aH2qOrPzAbndMCSa?dl=0`

[3, 7, 49].

On a more positive note, our evaluations suggest that there is a huge potential to be tapped by more sophisticated (neural) algorithms that combine short-term and long-term preference signals for session-aware recommendation. An important prerequisite for progress in this area however lies in an increased level of reproducibility of published research. A side observation of our research is that despite some positive developments in recent years, where researchers increasingly share their code on public repositories, it in many cases still remains challenging to reproduce existing works.

The paper is organized as follows. In Section 2, we discuss relevant previous works. Section 3 describes our research methodology in more detail with respect to the compared algorithms, the evaluation protocol, and the performance measures. The results of our experiments are reported in Section 4, and we discuss research implications and future works in Section 5.

2. Previous Work

Historically, recommender systems research focused strongly on the problem of rating prediction given a user-item rating matrix, a setting which is also known as the "matrix completion" problem [39]. In this original collaborative filtering problem setting, the order of the ratings or the time when they were provided were not considered in the algorithms. Soon, however, it turned out that these aspects can matter, leading to the development of *time-aware* recommender systems [5], e.g., in the form of the *timeSVD++* algorithm as used in the Netflix Prize [24].

Ten years after the Netflix Prize, the focus of research has mostly shifted from rating prediction to settings where mainly implicit feedback signals by users (e.g., purchase or item-view events) are available. Moreover, instead of considering the user-item matrix as the main input, recent research more often focuses on settings where the main input to a recommendation algorithm are sequential logs of recorded user interactions. The family of approaches that relies on such types of inputs are referred to as *sequence-aware* recommender systems [35].

Within this class of sequence-aware recommender systems, we can differentiate between three main categories of problem settings and algorithmic approaches.

- *Sequential Recommender Systems:* Unlike the other types of sequence-aware approaches discussed here, these systems are rooted in the tradition of relying on a user-item rating matrix as an input. The particularity of such systems is that the sequence of events is first extracted from the time-stamped rating matrix, and the goal then usually is to predict the immediate next user action (e.g., the next Point-of-Interest that a user will visit), given the entire preference profile of the user.

- *Session-based Recommender Systems:* The input to these systems are time-ordered logs of recorded user interactions, where the interactions are grouped into *anonymous sessions*. Such a session could, for example, correspond to a listening session on a music service, or a shopping session on an e-commerce site. One particularity of such approaches is that users are not tracked across sessions, which is a common problem on websites that deal with first-time users or users that are not logged in. The prediction task in this setting is to predict the next user action,

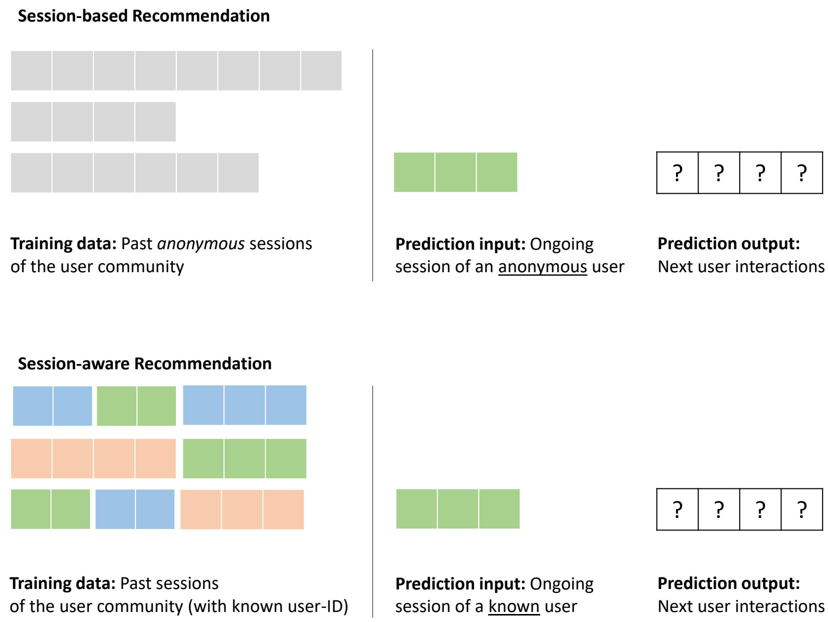

Figure 1: Comparison of session-based and session-aware recommendation problems. Different colors indicate different users.

given only the interactions of the current session. Today, session-based recommendation is a highly active research area due to its practical relevance.

- *Session-aware Recommender Systems:* This category is also referred to as *personalized session-based recommender systems*. It is similar to session-based recommendation in that the user actions are grouped into sessions. Also the prediction goal is identical. However, in this problem setting, users are not anonymous, i.e., it is possible to leverage information about past user sessions when predicting the next interaction for the current session.

Figure 1 illustrates the differences between session-*based* and session-

aware recommendation scenarios.[2] In both problem settings, the recommendation problem consists of predicting which action a user will do next in an ongoing session. In an e-commerce setting, for example, the problem is to predict which items are relevant for the user, given the last few observed interactions in an ongoing shopping session. The difference between the two scenarios however is that in one case we have long-term preference information about the user (session-aware recommendation), whereas in the other case such information is not available. Technically, this usually means that in session-aware scenarios we have user identifiers attached to the past usage sessions in the training data. In Figure 1, we indicate sessions by different users with different colors.

Unfortunately, the terminology in the literature is not entirely consistent. In this work, we will therefore use the categorization and terminology as described above to avoid confusion. Next, we review the main technical approaches in each category.

Sequential Recommendation Approaches. The first comparably simple approaches in this category were based on Markov models, e.g., [32]. Later on, more sophisticated approaches emerged which, for example, combined the advantages of matrix factorization techniques with sequence modeling approaches. Rendle et al. [38] proposed the Factorized Personalized Markov Chain (FPMC) approach as an early method for next-basket recommendation in e-commerce settings, where user interactions are represented as a

[2]Remember that sequential problems, which are not based on the concept of a session, are not in the scope of this work.

three dimensional tensor (user, current item, next-item). Kabbur et al. [22] later proposed FISM, a method based on an item-item factorization. FISM was then combined with factorized Markov chains to incorporate sequential information into the FOSSIL model [12].

In recent years, several sequential recommender systems based on neural networks were developed. Kang and McAuley [23], for example, proposed SASREC (self-attention based sequential model), a method that allows to capture long-term semantics like an RNN. However, through the use of an attention mechanism, it focuses only on a smaller set of interactions to make the item predictions. In the CASER method, Tang and Wang [45] embedded a sequence of recent items into latent spaces as an "image" in time, and proposed to learn sequential patterns as local features of the image with the help of convolutional filters. Most recently, Sun et al. [43] proposed BERT4REC, which employs a deep bidirectional self-attention mechanism to model user behavior sequences.

In this work, we do not consider this class of algorithms in our performance comparison because these methods, in their original designs, do not consider the concept of a session in the input data. While it is in principle possible to apply these methods in a particular way for session-based recommendation problems, a previous evaluation shows that sequential approaches are often not competitive with techniques that were specifically designed for the problem setting. Specifically, the evaluation presented in [27] included a number of sequential approaches, namely FPMC, MC, SMF, BPR-MF, FISM, and FOSSIL.[3] Their findings showed that *(i)* these approaches either are gen-

[3] FPMC: Factorized Personalized Markov Chains [38], MC: Markov Chains [32], SMF:

erally not competitive in this setting or only lead to competitive results in a few specific cases and *(ii)* that nearest neighbor recommenders outperform them in terms of prediction accuracy.

Session-based Recommendation Approaches. While there exist some earlier works on session-based recommendation, e.g., in the context of website navigation support and e-commerce [31, 42], research on this topic started to considerable grow only in the mid-2010s. These developments were particularly spurred by the release of datasets in the context of machine learning competitions, e.g., at ACM RecSys 2015. At around the same time, deep learning methods were increasingly applied for recommendation problems in general. The first deep learning approach to session-based recommendation was GRU4REC [15], which is based on Recurrent Neural Networks. Later on, various other types of neural architectures were explored, including attention mechanisms. convolutional neural networks, graph neural networks or hybrid architectures, see [47] for an overview.

Recent work however indicates that in many cases much simpler methods can achieve similar or even higher performance levels than today's deep learning models. Most recently, Ludewig et al. [29] benchmarked several of the mentioned neural methods against conceptually simpler session-based algorithms which, for example, rely on nearest-neighbor techniques. Quite interestingly, their analyses and similar previous works [8, 27] not only show

Session-based Matrix Factorization [27], BPR-MF: Bayesian Personalized Ranking [37], FISM: Factored Item Similarity Models [22], FOSSIL: FactOrized Sequential Prediction with Item SImilarity ModeLs [12].

the strong performance of conceptually simple techniques, but also revealed that two of the earlier neural methods, GRU4REC and NARM, often perform better than more recent complex techniques. In the performance comparison in this present work on session-aware recommendation, we include several techniques for session-based recommendation as baselines. This allows us to assess the added value of considering long-term preference information compared to a situation where such information is not available or not leveraged.

Session-aware Recommendation Approaches. The literature on session-aware recommendation is still quite sparse. An early approach is discussed in [20]. One main goal of their work was to understand the relative importance of short-term user intents when visiting an e-commerce site compared to the long-term preference model. Their analyses, which were based on a large but private e-commerce dataset, emphasized the importance of considering the most recent observed user behavior when recommending. Furthermore, it also turned out that *reminding* users of items that they have viewed before can be beneficial, both in terms of accuracy measures and business metrics.

While the work in [20] relied on deep learning for the final predictions in one of their models, the core of the proposed technical approach was based on feature engineering and the use of side information about the items to recommend. One of the earliest "pure" deep learning techniques for session-aware recommendation was proposed by [36]. Technically, the authors based their work on GRU4REC, and they used a second, parallel GRU layer to model information across sessions, resulting in a model called HGRU4REC. Their analyses showed that incorporating long-term preferences can be beneficial, i.e., HGRU4REC was outperforming an early version of GRU4REC in their experiments.

In the same year, Ruocco et al. [40] proposed the IIRNN model. Like HGRU4REC, this model uses an RNN architecture and extends a session-based technique to model inter-session and intra-session information. Like in the case of HGRU4REC, the authors investigate the value of considering long-term preference information by comparing their method to session-based techniques. RNNs were later on also used in the NSAR model [33] to encode session patterns in combination with user embeddings to represent long-term user preferences across sessions. In their experiments, the authors not only compare their model to session-based techniques, but also to HGRU4REC as a representative of a session-aware approach.

A number of neural architectures other than RNNs were proposed in recent years. Hu et al. [17], for example, combine the inter-session and intra-session context with a joint context encoder for item prediction in the NCSF approach. In the SHAN model [50], in contrast, the authors leverage a two-layer hierarchical attention network to model short-term and long-term user interests. In the SWIWO [16] approach, the authors were inspired by language modeling approaches like *word2vec*, where the underlying idea is that items can be seen as words, hence, predicting a relevant word based on context information is equivalent to recommending a relevant item in an ongoing session. Finally, Cai and Hu [4] proposed the SAMR method, which leverages a topic-based probabilistic model to define the users' listening behavior.

Unlike most previous works, which compare a newly-proposed session-aware model with previous session-based ones, our work compares session-aware methods against each other. Furthermore, we benchmark several of these recent session-aware methods with *(i)* existing session-based techniques and *(ii)* extended versions of them that also consider long-term preference information. We describe our research methodology next.

3. Research Methodology

In this section, we describe which algorithms we selected for inclusion in our comparison. Moreover, we provide details about the experimental configuration in terms of the evaluation protocol and the used datasets. As mentioned, all datasets and the code used in the experiments are shared online to ensure reproducibility.

3.1. Compared Algorithms

In our experiments, we compare neural session-aware algorithms with a number of baselines. Details about the algorithms are provided next.

3.1.1. Neural Session-Aware Algorithms

We identified five recent neural approaches, which we integrated into our evaluation framework using the source provided by the authors: HGRU4REC, IIRNN, SHAN, NCSF, and NSAR.

- HGRU4REC: This method [36] is based on the GRU4REC algorithm. To model the interactions of a user within a session, it utilizes RNNs based on a single GRU layer. By adding an extra GRU layer, it models information across user sessions. The user-level GRU initializes the hidden state of the session-level GRU and optionally propagates the user representation in the input to the session-level GRU to personalize its recommendations.

- IIRNN: This method [40] extends a session-based recommender built on RNN, called intra-session RNN, by using a second RNN that is called inter-session RNN. The intra-session RNN is the same as in the GRU4REC model. The inter-session RNN learns from the user's recent sessions and feeds the information to the intra-session RNN. At the beginning of every session, the final output of the inter-session RNN initializes the hidden state of the intra-session RNN.

- SHAN: This model [50] uses a two-layer hierarchical attention network to learn a hybrid representation for each user that combines the long-term and short-term preferences. It first embeds sparse user and item inputs into low-dimensional dense vectors. The long-term user representation is a weighted sum over the embeddings of items in the long-term item set. By learning the weights, the first attention layer learns user long-term preferences. The second attention layer returns the final user representation by combining the long-term user model and the embeddings of the items in the short-term item set.

- NCSF: This session-aware neural method [17] has three components: *(i)* the historical session encoder to represent the inter-session context, *(ii)* the current session encoder to represent the intra-session context, and *(iii)* the joint context encoder to integrate the information of the intra-session context and the inter-session context for item prediction.

- NSAR: This method [34] utilizes RNNs to encode session patterns (short-term user preferences) and user embeddings to represent long-term user preferences across sessions. It supports different ways of integrating user long-term preferences with the session patterns, where user embeddings can either be integrated with the input or the output of the session RNNs. Moreover, with the help of a gating mechanism, the contribution of each component can be fixed or adaptive.

To avoid a bias in the algorithm selection, we applied the following procedure to identify algorithms for inclusion in our experiments. An initial set of candidate works was retrieved through a search on Google Scholar using search terms like “session-aware recommendation” or “personalized session-based recommendation”. We inspected the returned results to see if the papers fulfilled our inclusion criteria. Besides being actually a work on session-aware recommendation according to the above definition, we required that the source code of the method was publicly available and could be integrated into our Python-based evaluation framework[4]. Moreover, we only considered papers that had undergone a peer review process, i.e., we did not include non-reviewed preprints. Finally, we included only works that did not consider side information about the items. As a result of this last constraint, we did not include works like [44] for the domain of news recommendation.

In Table 3.1.1, we show to which baselines the selected neural approaches were compared in their original publications. Note that in this table only the last two rows, HGRU4REC and SWIWO represent session-aware techniques. Our analysis furthermore shows that researchers use a variety of baselines in their experiments, which contributes to the difficulty of understanding what

[4] For example, the code used in [4] is not publicly available, and the method in [16] is written in MATLAB.

represents the state-of-the-art.

3.1.2. Neural and Non-Neural Session-based Baselines

We selected the baselines according to the results of [29]. Note that while GRU4REC and NARM are not the most recent neural methods, the analysis in [29] showed that they are highly competitive among the neural models for different datasets.

- GRU4REC: This neural model [15] employs RNNs based on Gated Recurrent Units (GRU) for the session-based recommendation task. It introduces several modifications, including a ranking loss function, to classic RNNs to adapt it for the recommendation task in the session-based setting. The authors later on improved the model with an alternative loss function and by applying further refinements, [14]. We include the latest version of GRU4REC in our experiments.

- NARM: This model [26] extends GRU4REC. It utilizes a hybrid encoder with an attention mechanism to model the sequential behavior of users and capture their main intent of the current session. A bilinear matching scheme is used to compute the recommendation scores of each candidate item based on a unified session representation.

- SR: The *Sequential Rules* method, proposed in [27], extends the simple *Association Rules* technique (also from [27]) and counts pairwise item co-occurrences in the training sessions. It considers the order of the items in a session as well as the distance between them when scoring the items.

Table 1: Overview of the baseline techniques that each neural session-aware approach was originally compared to. The methods are ordered chronologically by the date of publication. The marks (✗) indicate which baselines were used in the comparison.

		Method				
		HGRU4REC	IIRNN	SHAN	NCSF	NSAR
Baseline	MR		✗			
	POP		✗	✗	✗	
	PPOP	✗				
	IKNN	✗	✗		✗	
	BPR-MF		✗	✗		✗
	FPMC			✗	✗	
	FOSSIL			✗		
	GRU4REC	✗	✗		✗	✗
	GRU4REC2				✗	✗
	BPR-GRU4REC2					✗
	HRM			✗		
	CASER					✗
	HGRU4REC					✗
	SWIWO				✗	

MR: most recent interacted item; POP: most popular item in the dataset; PPOP: most popular item for the user; IKNN: Item-based kNN [15]; BPR-MF: Bayesian Personalized Ranking [37]; FPMC: Factorized Personalized Markov Chains [38]; FOSSIL: FactOrized Sequential Prediction with Item SImilarity ModeLs [12]; GRU4REC: [15]; GRU4REC2: the improved version of the GRU4REC model [14]; HRM: Hierarchical Representation Model [46]; CASER: Convolutional Sequence Embedding Recommendation Model [45]; BPR-GRU4REC2: a baseline proposed in [34] that merges the rankings returned by BPR-MF and GRU4REC2 following the proposed framework in [18] for session-aware setting; HGRU4REC: [36]; SWIWO: [16].

- VSKNN: This nearest-neighbor baseline for session-based recommendation was proposed in [27], and it is based on the SKNN method [19]. It first finds past sessions that contain the same items as the current session. The recommendation list is then generated using items that occur in the most similar sessions. This method considers the order of the items while computing both session similarities and item scores. Moreover, it applies the Inverse-Document-Frequency (IDF) weighting scheme to put more emphasis on less popular items.

- STAN: The *Sequence and Time Aware Neighborhood* method was proposed in [8]. It improves SKNN by considering three additional factors for making recommendations: *(i)* the recency of an item in the current session, *(ii)* the recency of a past session w.r.t. the current session, and *(iii)* the distance of a recommendable item w.r.t. a shared item in the neighboring sessions.

- VSTAN: This nearest-neighbor session-based recommendation algorithm combines all the extensions to SKNN from STAN and VSKNN in a single approach; proposed in [29].

3.1.3. Extensions of Session-based Baselines

We experimented with three simple ways of extending session-based algorithms in a way that they consider past preference information.

- EXTEND – *Extending the current session with recent interactions*: In case there is little information available about the ongoing session, e.g., when only a few first clicks are recorded, we extend the current session with interactions that we observed in the previous sessions of the user.

- BOOST – *Emphasizing previously seen items:* In some domains, repeated interactions with already seen or consumed items are not uncommon. We apply a simple "boosting" approach to slightly increase the scores computed by an underlying algorithm in case an item has appeared previously in the interaction history.

- REMIND – *Applying reminding techniques:* In [20], the authors proposed a number of "reminder" techniques to emphasize items the user has seen before. The general approach is to determine and score a set of candidate items that the current user has interacted with in the past. We considered different strategies that were inspired by [20] in our evaluations.

EXTEND. We implemented this strategy as follows. First, we choose a value for the "desired session length" d, which is a hyperparameter to be determined on the validation set. In case an ongoing session has fewer interactions than d, we extend the current session with previous interactions from the same user until the session length d is reached or no more previous interactions exist. The extension is done by simply prepending the elements of previous interactions to the current session in the order they appeared in the log.

BOOST. This simple approach can in principle be applied to any algorithm that returns scores. Methods like SR and VSKNN, for example, return scores based on item co-occurrences and the positions of the co-occurring items, as described in [27]. In our experiments, we used a hyperparameter b as a boost factor. Technically, we look up each item that is recommended by the underlying method and check if it occurred in the interaction history of the current user at least once. In case the item appeared previously in the history, we increase the original score by b %.

Remind. Different reminding strategies were proposed in [20]. In our experiments, we tested a number of alternative ways to select and score items to consider as reminders. In all of them, the reminder list is created by adding items of the user's last p sessions, which is a hyperparameter to be determined on the validation set. The assumption is that very old sessions at some stage might become irrelevant and should not be considered anymore. The following reminder strategies are the best performing ones according to our experiments.

Interaction Recency. In this strategy, we use a decay function to score reminder items, i.e., items in the interaction history of the user, based on the time of the user's last interaction with them:

$$\text{IRecScore}(i) = T_c/(T_c - T_i) \tag{1}$$

Here, T_c is the timestamp of the current session, and T_i is the timestamp of the last interaction of the user with the item i.

Session Similarity. In this strategy, the items of the last P sessions of the user are scored based on the similarity score(s) of the current session and the previous session(s) that they belong to. This strategy is based on a nearest-neighbor recommendation method to calculate the similarity scores between sessions:

$$\text{SSimScore}(i) = \sum_{p=1}^{P} Sim(S_c, S_p).1_p(i) \tag{2}$$

Here, S_c denotes the current session, and S_p denotes a session in the set of the last P past sessions of the user. *Sim* is the similarity function of the nearest-neighbor algorithm, and the indicator function $1_p(i)$ returns 1 if the session p contains item i and 0 otherwise.

In our present work, we used a *hybrid* approach to combine aspects of interaction recency (IRec), session similarity (SSim), and the item's relevance score (RelScore), as determined by a recommendation algorithm, in a weighted approach. The overall ranking score for an item i is therefore defined as

$$\text{OverallScore}(i) = W1 \times \text{RelScore}(i) + W2 \times \text{IRecScore}(i) + W3 \times \text{SSimScore}(i) \tag{3}$$

Here, W1, W2, and W3 are hyperparameters to tune, and the individual scores are normalized before they are combined. Note that for algorithms that are not based on nearest-neighbors, we do not compute a SSimScore value, and we thus set W3 to 0.

3.2. Datasets

We conducted our evaluations on four public datasets from three different domains: e-commerce, social networks and music.

- *RETAIL*: A dataset published by the e-commerce personalization company *Retail Rocket*. It covers user interactions with a real-world e-commerce website over 4.5 months.
- *XING*: A dataset published in the context of the ACM RecSys 2016 Challenge that contains interactions of job postings on a career-oriented

social networking site, *XING*, from about three months. It contains a fraction of *XING* users and job postings, and the data is enriched with artificial users for privacy reasons. The interactions include four types of actions: *click, bookmark, reply, and delete*. We filtered out all interactions of type *delete* from the dataset for our evaluation, as done in [36], because they are considered as negative interactions.

- *COSMETICS*: An e-commerce dataset containing the event history of a cosmetics shop for five months.[5] The interactions include four types of actions: *click, view, add-to-cart, remove-from-cart, purchase*. Since the methods examined in our work are not designed to consider multiple types of actions, we only used the interactions of one type (*item views*) for our evaluation, as was done also in various previous works [15, 26]. Furthermore, we randomly sampled 10% of the users of this large dataset because of scalability issues of some of the neural methods.

- *LASTFM*: A music dataset that contains the entire listening history of almost 1,000 users during five years. The dataset was retrieved from the online music service *Last.fm*[6].

For each dataset, we first partitioned the log into sessions by applying a commonly used 30-minute user inactivity threshold[7]. We kept multiple inter-

[5] `https://www.kaggle.com/mkechinov/ecommerce-events-history-in-cosmetics-shop`

[6] `https://www.last.fm/`

[7] The COSMETICS dataset already contains session IDs, but we noticed that there were large inactivity thresholds and some of the original sessions spanned several days.

actions with the same item in one session because repeated recommendations can help as reminders [20, 25]. Many previous works on session-aware recommendation use a single training-test split of the whole dataset or a sample of it for evaluation. Evaluating on only one split of data is risky because of possible random effects. We therefore split each dataset into five contiguous subsets by time and averaged the results across slices as done in [29]. To have about the same number of events for each slice, we skipped the first 500 days of the LASTFM dataset.

Following common practice in the field, we then further pre-processed each slice as follows. We first removed items with less than five interactions in the slice. Then, we removed sessions that contain only one event. For the LASTFM dataset, we also removed sessions with more than 20 events[8]. Finally, we filtered out users with less than three sessions. Table 2 shows the average characteristics of the slices for each dataset after the pre-preprocessing phase.

Table 2: Characteristics of the datasets. The values are averaged over all five slices.

	Events	Users	Sessions	Items	Sessions per User	Actions per Session
RETAIL	45,378	1,400	7,198	10,424	5.15	6.28
XING	333,625	13,533	59,318	61,006	4.38	5.62
COSMETICS	81,159	1,821	9,826	17,790	5.39	8.24
LASTFM	750,276	658	94,818	107,134	144.61	7.92

[8]The dataset contains a number of very long sessions with dozens of listening events, and the probability that users *actively* listened to tracks for many hours seems low. Therefore, we only considered the first 20 elements, which corresponds to a listening session about 1.5 hours for the case of pop music, see also [33, 40] for similar approaches.

3.3. Evaluation Protocol and Metrics

Creation of Training, Validation, and Test Splits. Since we are given user-IDs for the sessions, we are able to apply a user-wise data splitting approach. Specifically, like in [11, 33, 34, 36, 48], we use the last session of each user as test data. The second-to-last session is used as validation data to tune the parameters in our experiments, see also [33, 34, 48]. The remaining sessions are considered as training data. This splitting approach allows us to assess the performance both for users with shorter and longer interaction histories. We finally filter out the items from the validation and test sets of each of the five slices that did not appear in the training set of that slice.

Target Item Selection. We apply a procedure that is commonly used also in session-based recommendation, e.g., in [15] and many other works. Specifically, we iteratively reveal each item after the other in the test session and do the evaluation after each item[9].

We apply this approach as it reflects the most realistic user behavior in a session. Following previous works [27, 28, 29], we consider two evaluation scenarios. In one case, we only consider the immediate next item in the test session as a ground truth. In the other, more realistic case, all upcoming items in the test session are considered relevant.

---

[9] In the implementation of the NCSF method, we found that the authors use a *context window*, which includes items *before and after* a given target item t to make the prediction. In reality, however, we cannot know which items will appear after t. As we could not resolve this issue with the authors, we used an implementation that only uses items appearing *before* the target item as the context.

Accuracy Metrics. We use standard classification and ranking measures to evaluate the performance of the recommendation algorithms. We measure the performance in two different ways, as done in [29], according to the target item selection approach. First, when we consider only the immediate next item as the target item, the used metrics are the Hit Rate (HR) and Mean Reciprocal Rank (MRR). Second, when all items of the current session are assumed to be relevant to the user, we consider all the remaining items of an ongoing session as the target item. In this case, the used accuracy metrics are Precision, Recall, and Mean Average Precision (MAP).

Coverage and Popularity Bias Metrics. It is well known that factors other than accuracy can impact the effectiveness of a recommendation algorithm in practice [41]. In this work, we consider *coverage* and the tendency of an algorithm to focus on popular items (*popularity bias*) as relevant factors. *Coverage* tells us how many items actually ever appear in the top-n lists of users. This measure, which is also known as "aggregate diversity" [1], gives us some indication of how strongly personalized the recommendations of an algorithm are. A strong popularity bias, on the other hand, indicates that an algorithm is not focusing much on the long-tail of the item catalog, which, however, can be desirable in practice. We calculate the *popularity bias* as done in [27]. Specifically, we average the popularity scores of all recommended items. These popularity scores are computed by counting how often each item appears in the training set. To bound their values between 0 and 1, we apply min-max normalization.

Computational Complexity. Training deep learning models is often considered computationally demanding. Nearest-neighbor techniques, on the other hand, have no model-building phase, but the search for neighbors can, depending on the implementation, be computationally complex at run time. In our experiments, we therefore use the recency-based sampling approach proposed in [19] for all session-based nearest-neighbor approaches. To compare the computational complexity of the various methods, we measured the time that is needed by each algorithm, both for the training and the prediction phases. All these measurements were made on the same physical machine, which was exclusively used for the time measurements. The machine is equipped with a Nvidia Titan Xp GPU and an Intel i7-3820 CPU. The neural models used the GPU whereas the non-neural techniques only used the CPU.

Hyperparameter Optimization. To obtain reliable results, we systematically and automatically tuned the hyperparameters for each algorithm and dataset. Technically, we applied a random hyperparameter optimization procedure with 100 iterations to optimize MRR@20 as done, e.g., in [27][10]. For NARM, we however only ran 50 iterations as this method has a smaller set of hyperparameters. For SHAN, we only ran 9 hyperparameter configurations since they cover all possible value combinations according to the original paper[11]. For each dataset, we used the slice with the most number of events to tune hyperparameters.

[10] We also tried MAP@20 as the optimization target for some approaches, but this did not lead to a different ranking of the algorithms in terms of accuracy.

[11] We unsuccessfully contacted the authors regarding the hyperparameter spaces.

4. Results

Tables 3–6 show the results of our performance comparison of neural and non-neural methods, ordered by the values obtained for the MAP@20 metric. Here, we correspondingly report the values obtained by applying a cut-off threshold of 20. We performed additional experiments using alternative cut-off lengths (5 and 10). The rankings of the algorithms for those other cut-off values were generally in line with those observed at list length 20. Non-neural methods are highlighted with a light gray background in the tables. Neural session-*based* methods have a gray background. Session-*aware* techniques finally have a dark gray background.

Note that we do not report all possible combinations of the proposed extensions discussed in Section 3.1.3 for the sake of conciseness. We use the following naming scheme for the different algorithm variants.

- REMIND: We denote algorithm variants that were extended with the reminder technique with the postfix "_R". Note that it is not meaningful to incorporate the reminder extensions to session-*aware* methods, as these models should already be able to implicitly leverage the long-term preference information.

- EXTEND and BOOST: We report the effects of these extensions for the non-neural methods[12], and we denote the extended method by append-

[12]In principle, these mechanisms can also be applied to the neural session-based methods. Initial experiments with the EXTEND extension for GRU4REC and NARM however did not lead to performance gains. Note also that the adaptation of existing neural session-based methods is not the main focus of our work.

ing the postfix "_E" (extend) and "_B" (boost) to the algorithm name, e.g., SR_B, VSKNN_EB. These extensions can also be combined with the reminder technique, e.g., VSKNN_EBR. For the SR method, only boosting was applied because extending the session is not applicable for this algorithm, which by design only considers the last interaction in a session.

For the sake of clarity, for each model and dataset, we report *(i)* the combination of the extensions that led to the best performance according to MAP@20 and *(ii)* the results of the original models without extensions. In case the original model had better performance than the extended ones, we only report the results for the original model.

In our analysis, we focus on the performance comparison of non-neural methods and neural session-aware recommendation techniques. Therefore, in Table 3 to Table 6, the highest obtained values among algorithms of these two families are printed in bold. Moreover, we underline the highest value that is obtained by the other family of algorithms. Stars indicate significant differences ($p<0.05$) according to a Kruskal–Wallis test between all the models and a Wilcoxon signed-rank test between the best-performing techniques from either category (non-neural or neural session-aware recommendation methods).

4.1. Accuracy Results

We can summarize the accuracy results for the individual datasets as follows.

Table 3: Results of the performance comparison on the RETAIL dataset with the focus on the comparison of **simple (non-neural)** methods and **neural session-aware** ones. The best results for each metric are highlighted in bold font. The next best results for algorithms from the other category (either simple methods or session-aware ones) are underlined. Non-neural methods are highlighted in light gray, session-based ones in gray and neural session-aware ones in dark gray.

Metrics	MAP@20	P@20	R@20	HR@20	MRR@20	COV@20	POP@20
RETAIL							
STAN_ER	∗**0.0350**	**0.0759**	0.5214	0.7043	0.4411	0.8374	0.0576
VSTAN_EBR	∗**0.0350**	0.0758	∗**0.5215**	∗**0.7062**	∗**0.4432**	0.8525	0.0553
VSKNN_EBR	0.0343	0.0750	0.5097	0.6938	0.4155	∗**0.8632**	0.0534
NARM_R	0.0320	0.0691	0.4841	0.6419	0.3850	0.8008	0.0614
STAN	0.0311	0.0680	0.4837	0.6747	0.4411	0.7818	0.0582
VSTAN	0.0310	0.0675	0.4825	0.6745	0.4397	0.7947	0.0580
VSKNN	0.0303	0.0669	0.4646	0.6476	0.4164	0.8055	<u>0.0469</u>
SR_BR	0.0301	0.0662	0.4547	0.6018	0.3666	0.8063	0.0552
GRU4REC_R	0.0292	0.0621	0.4565	0.6417	0.4305	0.9086	0.0405
GRU4REC	0.0272	0.0576	0.4367	0.6172	0.4196	0.9059	0.0394
NARM	0.0252	0.0542	0.4086	0.5650	0.3566	0.7620	0.0622
IIRNN	<u>0.0239</u>	<u>0.0524</u>	<u>0.3775</u>	0.5108	0.3190	<u>0.7709</u>	0.0689
HGRU4REC	0.0226	0.0485	0.3681	<u>0.5165</u>	<u>0.3296</u>	0.7502	**0.0425**
NCSF	0.0217	0.0468	0.3625	0.5042	0.3120	0.6871	0.0967
SR	0.0213	0.0460	0.3477	0.4847	0.3265	0.7186	0.0528
SHAN	0.0205	0.0451	0.3448	0.4498	0.2673	0.3406	0.1276
NSAR	0.0169	0.0370	0.2830	0.3702	0.2160	0.5813	0.0671

Table 4: Results of the performance comparison on the XING dataset with the focus on the comparison of **simple (non-neural)** methods and **neural session-aware** ones. The best results for each metric are highlighted in bold font. The next best results for algorithms from the other category (either simple methods or session-aware ones) are underlined. Non-neural methods are highlighted in light gray, session-based ones in gray and neural session-aware ones in dark gray.

Metrics	MAP@20	P@20	R@20	HR@20	MRR@20	COV@20	POP@20
XING							
VSTAN_R	∗**0.0194**	∗**0.0497**	∗**0.3031**	∗**0.4445**	∗**0.2837**	0.9581	0.0373
STAN_R	∗**0.0194**	0.0495	0.3027	0.4436	0.2828	0.9563	0.0395
VSKNN_R	0.0182	0.0466	0.2880	0.4304	0.2691	∗**0.9660**	0.0344
NARM_R	0.0174	0.0448	0.2747	0.3961	0.2095	0.9400	0.0452
SR_BR	0.0158	0.0413	0.2452	0.3463	0.1852	0.9459	0.0371
GRU4REC_R	0.0151	0.0387	0.2512	0.3959	0.2721	0.9363	0.0311
VSKNN	0.0139	0.0357	0.2312	0.3783	0.2605	0.9193	<u>0.0305</u>
VSTAN	0.0138	0.0353	0.2368	0.3890	0.2747	0.8996	0.0353
STAN	0.0137	0.0352	0.2367	0.3887	0.2734	0.8950	0.0386
NARM	0.0117	0.0304	0.2031	0.3320	0.2035	0.8252	0.0480
GRU4REC	0.0113	0.0284	0.2007	0.3454	0.2653	0.9174	0.0270
NCSF	<u>0.0101</u>	<u>0.0262</u>	<u>0.1800</u>	<u>0.2982</u>	<u>0.1706</u>	0.7885	0.0683
SR	0.0092	0.0238	0.1567	0.2532	0.1633	0.8279	0.0321
NSAR	0.0086	0.0229	0.1449	0.2013	0.0968	0.8268	0.0361
HGRU4REC	0.0081	0.0203	0.1464	0.2524	0.1681	<u>0.8474</u>	**0.0296**
IIRNN	0.0072	0.0185	0.1274	0.2046	0.1254	0.8387	0.0484
SHAN	0.0051	0.0151	0.0932	0.1231	0.0503	0.2673	0.1329

Table 5: Results of the performance comparison on the COSMETICS dataset with the focus on the comparison of **simple (non-neural)** methods and **neural session-aware** ones. The best results for each metric are highlighted in bold font. The next best results for algorithms from the other category (either simple methods or session-aware ones) are underlined. Non-neural methods are highlighted in light gray, session-based ones in gray and neural session-aware ones in dark gray.

Metrics	MAP@20	P@20	R@20	HR@20	MRR@20	COV@20	POP@20
				COSMETICS			
STAN_EBR	∗**0.0212**	0.0741	∗**0.2819**	∗**0.4270**	0.1741	0.9585	0.0708
VSTAN_EBR	0.0210	∗**0.0751**	0.2766	0.3959	0.1682	0.9512	0.0606
VSKNN_EBR	0.0209	0.0749	0.2744	0.4128	0.1683	∗**0.9714**	0.0601
GRU4REC_R	0.0176	0.0617	0.2410	0.3808	0.1575	0.9114	0.0473
NARM_R	0.0175	0.0644	0.2337	0.3273	0.1223	0.9130	0.0674
VSKNN	0.0175	0.0628	0.2405	0.3870	0.1665	0.9626	<u>0.0562</u>
STAN	0.0173	0.0619	0.2494	0.4035	0.1760	0.9419	0.0655
VSTAN	0.0172	0.0615	0.2490	0.4041	∗**0.1765**	0.9449	0.0648
SR_BR	0.0170	0.0623	0.2286	0.3386	0.1354	0.9528	0.0613
GRU4REC	0.0143	0.0504	0.2083	0.3383	0.1417	0.8811	0.0449
NCSF	<u>0.0133</u>	<u>0.0489</u>	<u>0.1903</u>	<u>0.2969</u>	0.1099	0.6973	0.1043
NARM	0.0129	0.0473	0.1891	0.2970	0.1175	0.8566	0.0694
HGRU4REC	0.0123	0.0442	0.1797	<u>0.2969</u>	<u>0.1198</u>	<u>0.9332</u>	**0.0468**
IIRNN	0.0123	0.0458	0.1761	0.2825	0.1119	0.8909	0.0778
SR	0.0111	0.0411	0.1654	0.2755	0.1161	0.9179	0.0574
NSAR	0.0081	0.0324	0.1217	0.1845	0.0617	0.7748	0.0680
SHAN	0.0054	0.0226	0.0898	0.1225	0.0434	0.2166	0.1985

Table 6: Results of the performance comparison on the LASTFM dataset with the focus on the comparison of **simple (non-neural)** methods and **neural session-aware** ones. The best results for each metric are highlighted in bold font. The next best results for algorithms from the other category (either simple methods or session-aware ones) are underlined. Non-neural methods are highlighted in light gray, session-based ones in gray and neural session-aware ones in dark gray.

Metrics	MAP@20	P@20	R@20	HR@20	MRR@20	COV@20	POP@20
LASTFM							
VSKNN_EB	∗**0.0493**	∗**0.1003**	∗**0.4248**	∗**0.5383**	0.1771	0.5053	0.0499
VSKNN	0.0474	0.0964	0.4104	0.5227	0.1725	0.5091	<u>0.0479</u>
VSTAN_EB	0.0449	0.0943	0.4088	0.5291	0.1938	0.5080	0.0528
STAN_EBR	0.0442	0.0924	0.3972	0.5159	0.1902	0.4965	0.0564
STAN	0.0400	0.0856	0.3760	0.5012	0.1886	0.5067	0.0557
VSTAN	0.0394	0.0852	0.3791	0.5100	0.1935	<u>0.5115</u>	0.0555
SR_B	0.0359	0.0778	0.3408	0.4622	<u>0.3487</u>	0.5010	0.0549
SR	0.0348	0.0765	0.3383	0.4615	0.3350	0.4982	0.0544
IIRNN	<u>0.0311</u>	<u>0.0724</u>	<u>0.3270</u>	<u>0.4729</u>	**0.3491**	0.4443	0.0732
GRU4REC	0.0307	0.0701	0.3175	0.4681	0.3342	0.5124	0.0468
NSAR	0.0280	0.0675	0.3044	0.4350	0.2906	0.4937	0.0483
NARM	0.0272	0.0658	0.3002	0.4510	0.3192	0.4748	0.0633
NCSF	0.0219	0.0556	0.2639	0.4393	0.2434	0.4875	0.0712
HGRU4REC	0.0208	0.0517	0.2464	0.4206	0.3167	**0.5647**	**0.0404**
SHAN	0.0072	0.0223	0.0920	0.1149	0.0345	0.0824	0.1640

RETAIL. Quite surprisingly, simple nearest-neighbor recommenders win on all the accuracy measures on this dataset, with STAN_ER and VSTAN_EBR being the best-performing methods. Moreover, the heuristic extensions of the session-based algorithms further improve their accuracy performance in all but one cases (MRR for VSKNN_EBR).

Probably even more surprising is that we find the session-*aware* methods at the very end of the performance ranking. This means that they are actually outperformed by methods which do not consider any long-term preference information at all. Specifically, all neural and non-neural session-based methods, except the basic SR method, outperform all session-*aware* methods for all accuracy metrics.

The best results in the class of neural session-*aware* recommendation algorithms are achieved by IIRNN and HGRU4REC. These two methods are, however, the earliest proposed session-aware recommendation methods in this comparison. Differently from the original paper, HGRU4REC is not able to outperform the last version of GRU4REC on this dataset.[13]

XING. Similar patterns are also found for the XING dataset, where *(i)* the neighborhood-based methods led to the best results, *(ii)* the extensions resulted in performance improvements in all cases, *(iii)* and session-*aware* techniques were outperformed by all other methods, except the SR method. Among the session-*aware* recommendation methods, this time NCSF achieves the best accuracy results for all metrics, even though it was not originally

[13]Using alternative loss functions for HGRU4REC might help improving its performance. Such modifications of the original algorithms are however not in the scope of our work.

evaluated on this dataset.

Note that the reminders worked generally very well on this dataset. Reminders for example help to improve the performance of the neural session-based techniques (GRU4REC, NARM) to an extent that they sometimes outperform the basic nearest-neighbor techniques. However, when the extensions are also considered for the neighborhood-based techniques, their accuracy results are again much higher than those of the neural techniques.

COSMETICS. We observe similar results also on the COSMETICS dataset. Neighborhood-based methods lead to the best results for all accuracy metrics, and the extensions resulted in performance improvements in almost all cases. Moreover, all methods except SR and NARM outperform the session-*aware* techniques. Looking at the session-*aware* methods, we notice that NCSF has the best performance in all the accuracy metrics, except for the MRR, and HGRU4REC achieves the best results for the Hit Rate (along with NCSF) and the MRR.

LASTFM. The picture for this dataset is slightly different from the others in certain respects. First, while nearest-neighbor approaches again lead to the best results for Precision, Recall, MAP, and HR, these methods are outperformed by all neural session-based and session-aware methods except SHAN on the MRR. However, none of them, except IIRNN, outperforms the simple SR method and its extended version. The best results are obtained by IIRNN, which is however one of the earliest session-aware methods. Note, however, that the results by IIRNN are only minimally better than SR_B. Nonetheless, further investigations are needed to understand the reasons why some methods perform very well on the MRR on this particular dataset.

A second difference to the other datasets is that while applying the simple extensions EXTEND and BOOST again proves to be beneficial, the reminder extension does not improve the performance of the original methods in some cases for this dataset.[14] Therefore, we only report the results of the original neural session-based methods (GRU4REC and NARM).

Summary and Additional Observations. Table 7 summarizes the findings presented in Table 3–6 by reporting the best performing approaches on the individual datasets. In Table 7, we can see that VSKNN, STAN and VSTAN with their extensions largely dominate the field across the datasets and measures, and we recommend that these methods are considered as additional baselines in future performance comparisons. Only in one case a method from another category (i.e., IIRNN, an early proposed neural session-aware method) appeared among the best performing approaches.

Looking at the best models among the neural approaches, including both session-based and session-aware, in Table 8, we notice that NARM_R and GRU4REC_R outperform the session-aware models on the RETAIL, XING, and COSMETICS datasets. On the LASTFM dataset, where the reminder extension did not lead to any performance improvements for the neural session-based models, IIRNN is the best one. As a side observation, we see that the ranking of the neural algorithms is often not correlated with the publication year of the methods, i.e., newer methods are not consistently better than older ones.

[14]In some cases, the optimal weights for the reminders were 0 after tuning the hyperparameters.

Table 7: Best performing approaches for each dataset and each accuracy metric. Algorithms that were significantly better than the best performing algorithms from the other category (either **non-neural** or **neural session-aware**) are marked with *.

Datasets	RETAIL	XING	COSMETICS	LASTFM
MAP@20	*STAN_ER/VSTAN_EBR	*VSTAN_R/STAN_R	*STAN_EBR	*VSKNN_EB
P@20	STAN_ER	*VSTAN_R	*VSTAN_EBR	*VSKNN_EB
R@20	*VSTAN_EBR	*VSTAN_R	*STAN_EBR	*VSKNN_EB
HR@20	*VSTAN_EBR	*VSTAN_R	*STAN_EBR	*VSKNN_EB
MRR@20	*VSTAN_EBR	*VSTAN_R	*VSTAN	IIRNN

Table 8: Best performing approaches for each dataset and each accuracy metric for neural methods (session-based and session-aware).

Datasets	RETAIL	XING	COSMETICS	LASTFM
MAP@20	NARM_R	NARM_R	GRU4REC_R	IIRNN
P@20	NARM_R	NARM_R	NARM_R	IIRNN
R@20	NARM_R	NARM_R	GRU4REC_R	IIRNN
HR@20	NARM_R	NARM_R	GRU4REC_R	IIRNN
MRR@20	GRU4REC_R	GRU4REC_R	GRU4REC_R	IIRNN

4.2. Coverage and Popularity

Tables 3–6 also report the values of *coverage* and *popularity bias* of the algorithms. We can make the following observations.

Coverage. SHAN consistently has the lowest *coverage* value across all datasets. In other words, it has the highest tendency to recommend the same set of items to different users. This sets the algorithm apart from all other techniques. The coverage values of most other techniques are often not too far apart, and the difference between nearest-neighbor algorithms and neural algorithms is often small. However, all nearest-neighbors methods (both original and extended versions) outperform all session-*aware* models for all datasets except for the LASTFM dataset, where HGRU4REC achieves the best result. No other consistent pattern can be found here. What can be observed is that the achieved level of coverage and the ranking of the algorithms in this respect seems to depend on the datasets.

Popularity Bias. GRU4REC and HGRU4REC consistently have the lowest tendency to recommend popular items. In contrast, SHAN exhibits the highest *popularity bias* with a considerable difference. Moreover, IIRNN and NCSF achieve higher values than all other methods across all datasets. The neighborhood-based approaches are often in the middle. They are therefore not generally focusing more on popular items than neural approaches. Finally, we observe that using the proposed session-aware extensions in most cases leads to a higher *popularity bias*.

4.3. Scalability

For the sake of brevity, we only report the results for two datasets here and provide the results for the other datasets in the appendix. We made all the measurements on the validation slice (i.e., the largest slice in terms of the number of events) for each dataset. Table 9 and Table 10 show the results for the LASTFM and XING, respectively. We selected these two datasets for the discussion because they are larger than the other two datasets and differ in some key characteristics. The LASTFM dataset for example contains the largest number of events, but only a smaller number of users. The XING dataset, on the other hand, contains events from a larger number of users. The algorithms in these tables are grouped by the type of approach (non-

Table 9: Running times on the LASTFM dataset.

	Training Time (s)	Prediction Time (ms)
SR	4.05	25.88
VSKNN	1.23	169.29
STAN	1.02	251.66
VSTAN	1.17	405.32
SR_B	12.58	37.12
VSKNN_EB	2.08	188.65
STAN_EBR	2.81	466.23
VSTAN_EB	1.97	627.66
GRU4REC	101.20	30.18
NARM	3726.80	8.92
HGRU4REC	218.71	27.18
IIRNN	7702.04	245.06
NCSF	532.62	30.75
NSAR	1621.05	15.79
SHAN	35298.70	30.98

neural, session-based neural, session-aware neural). We report running times of the session-based algorithms (neural and non-neural) both for the best-performing extension and the corresponding method.

The overall results are similar across all datasets. In terms of the *training* times, the non-neural models are consistently the fastest because "training" for such models mainly involves the initialization of data structures (nearest-neighbor techniques) or the collection of simple count statistics (SR). Interestingly, strong differences can be observed between the neural session-*based* methods GRU4REC and NARM, with GRU4REC being orders of magnitude faster than NARM in the training phase. The performance of the session-

Table 10: Running times on the XING dataset.

	Training Time (s)	Prediction Time (ms)
SR	2.66	6.69
VSKNN	0.58	10.88
STAN	0.54	9.37
VSTAN	0.58	9.85
SR_BR	7.23	22.76
VSKNN_R	1.09	25.45
STAN_R	1.00	44.26
VSTAN_R	1.97	44.83
GRU4REC	53.89	10.54
NARM	1769.09	10.12
GRU4REC_R	47.64	23.26
NARM_R	1762.86	20.37
HGRU4REC	60.54	10.27
IIRNN	7229.03	250.39
NCSF	192.25	35.60
NSAR	12562.06	34.81
SHAN	13393.82	35.50

aware methods, finally, exhibit a large spread. HGRU4REC, which is based on GRU4REC, is the fastest method here. However, some session-aware methods can take very long[15], in particular the SHAN method.

Looking at the time that is needed to generate one recommendation list (*prediction time*, in milliseconds), we observe that most neural methods are among the fastest techniques on the LASTFM dataset, with prediction times being in the range of a few dozen milliseconds. However, the running time for

[15]Note that for the IIRNN method on the XING dataset we had to set max_epoch=20 (instead of 100) because of its very high computational complexity on this dataset.

the neural IIRNN method, which is the best performing neural method on this dataset[16], is much higher than for the other neural methods. For the XING dataset, it even is the slowest among all compared methods. The running times for the simple SR method are in the range of the neural methods. For the nearest-neighbor techniques, the prediction times depend on the dataset characteristics. For the XING dataset, for example, the prediction times are in the range of the neural methods. On the other hand, for the LASTFM dataset, where, on average, there is a large number of sessions in the history of the users, the time needed for creating one recommendation list can go up to a few hundred milliseconds.

Generally, the use of the proposed extensions (Boost, Extend, Remind) leads to increased computation times for training and predicting. Note, however, that the more efficient basic versions of the nearest-neighbor techniques already outperform all neural *session-aware* methods in all cases, except one.

Overall, as mentioned above, we observe quite a spread regarding the running times among the neural methods. A detailed theoretical analysis of the computational complexity of the underlying network architectures and individual architecture elements is however beyond the scope of our present work, which focuses on the empirical evaluation of various algorithms in terms of their prediction accuracy.

[16]Remember, however, that all nearest-neighbors methods—both the basic and extended versions—outperformed the IIRNN method on all accuracy metrics, except MRR.

5. Implications, Limitations, and Future Work

5.1. Implications and Guidelines

Different factors may have contributed to the surprising observations made in this paper. First, we can assume that session-*based* algorithms, which are often used as baselines to benchmark session-*aware* ones, have improved over time. This might for example be the case for the HGRU4REC method, which most probably used an earlier version of GRU4REC both as a session-based baseline and as a building block for the newly proposed method.

Another potential reason may however also lie in methodological issues and problematic research practices that were observed previously not only for more traditional top-n recommendation tasks, but also for other areas of applied machine learning such as information retrieval or time-series forecasting [3, 7, 30, 49]. According to several of these works, one main problem seems to be that researchers often invest substantial efforts to refine and fine-tune their newly proposed methods, but do not invest similar efforts to optimize the baselines to which they compare their new methods. This phenomenon can be seen as a form of a *confirmation bias*, where researchers mainly seek for evidence supporting their theories and claims, but do not appropriately consider other evidences or indications. In that context, also reproducibility issues can play a role; see also the discussion of reproducibility problems in AI and empirical computer science in general [9, 6]. While researchers in recent years more frequently publicly share the code of their newly proposed models, they only very rarely share the code of the baselines or the code that was used to fine-tune all algorithms in the experimental evaluations.

Another related problem can also lie in the *choice* of the baselines. In particular in recent years we observe that researchers only consider very recent baselines in their evaluations or limit themselves to neural methods [7]. This leads to the effect that long-known or more simple methods are overlooked and not considered as competitive baselines. This, in turn, can lead to a *cascade* effect, where only the most recent models are considered to represent the state-of-the-art, even though they are probably not better than what existed earlier.

A number of measures can be taken to avoid that we only observe illusions of progress in this area in the future. In terms of *reproducibility*, Gundersen et al. [10] recently proposed a number of guidelines for reproducible research in AI in general, which in principle also apply for research in recommender systems research. Transferred to our specific problem setting, it is important that scholars make it as easy as possible for others to replicate their research. This in particular includes the publication of a number of artifacts such as *(i)* the code of the proposed model *and* the baselines, *(ii)* the used datasets, before and after any preprocessing, and the code used for pre-processing, *(iii)* the code for running the experiments, including the code for tuning the hyper-paremeters, *(iv)* documentation and appropriate installation instructions.

Furthermore, in order to ensure progress, Ferrari Dacrema et al. [7] suggest a number of guidelines for top-n recommendation settings that are also relevant for session-aware recommendation problems. Specifically, researchers are encouraged to *(i)* include algorithms from different *families* in the experimental evaluations, i.e., not only consider neural techniques, *(ii)* optimize all baselines models in a systematic and automated way, and to *(iii)* carefully select and document the hyperparameter ranges and the optimization strategy.

5.2. Research Limitations and Future Work

In our work so far, we performed experiments for a variety of domains—including e-commerce, music, or job recommendation (social media)—using publicly available and commonly used datasets. Nonetheless, further experiments with additional datasets are needed and are part of our future work. Such experiments will help us ensure that our findings generalize to other domains and that they are not tied to specific dataset characteristics. Our experiments so far however indicate that the ranking of the *family* of algorithms is quite consistent across experiments, with today's neural approaches to session-aware recommendation not being among the top-performing techniques, and non-neural techniques working well across datasets.

Nonetheless, an interesting area for future work lies in an improved understanding in which ways dataset characteristics impact the performance and ranking of different algorithms, as was done for more traditional recommendation scenarios in [2]. Moreover, it would be interesting to analyze how sensitive the different algorithms are with respect to slight changes, including both the characteristics of the input data and hyperparameters. In practical environments, in which datasets continuously evolve due to new items and users, algorithms that are more robust with respect to these aspects, might be preferable. Finally, regarding running times—in particular for datasets where there is a larger set of past sessions per user—further performance enhancements of the nearest-neighbor methods might be achieved through additional engineering efforts.

Regarding the set of session-aware algorithms that were benchmarked in our work, we expect that a constant stream of new proposals will be published in the future. As such, our experimental evaluation can only represent a snapshot of the state-of-the-art in the area. Our current snapshot is furthermore limited to works where the source code was publicly available, as was done in [7]. An additional constraint that we applied when selecting baselines was that the code had to be written in Python. In our literature search, we only identified two related works that were not written in Python, [4] and [16]. The method proposed in [4] was written in Java, but no source code was provided. The work presented in [16] was written in MATLAB but is only marginally relevant for our work, as it *(i)* focuses on diversity aspects and *(ii)* provides a performance comparison only with session-based or sequential approaches, whereas our work focuses on session-aware techniques.

Finally, we observed that the size of the datasets used in the experiments, both in the original papers and in our work, are small compared to the amounts of data that are available in real-world deployments. The current limitations of the investigated deep learning models might therefore be a result of these dataset limitations, see also [21]. With more data and in particular with data spanning longer periods of time, neural techniques might be favorable and better suited to combine long-term preference signals with short-term user intents than today's methods.

6. Summary

Our in-depth empirical investigation of five recent neural approaches to session-aware recommendation has revealed that these methods, contrary to the claims in the respective papers, are not effective at leveraging long-term preference information for improved recommendations. According to our experiments, these methods are almost consistently outperformed by methods that only consider the very last interactions of a user. Furthermore, our analyses showed that non-neural methods based on nearest neighbors can lead to better performance results than ones based on deep learning, as was also previously observed for session-based recommendation in [29].

We see the reasons for these unexpected phenomena in methodological issues that are not limited to session-aware recommendation scenarios, in particular in the choice of the baselines in experimental evaluations or the lack of proper tuning of the baselines. On a more positive note, our findings suggest that there are many opportunities for the development of better neural and non-neural methods for session-aware recommendation problems. We in particular believe that it is promising to look at repeating patterns, seasonal effects, or trends in the data. Moreover, the incorporation of side information (e.g., category information about items) as well as contextual information should help to further improve the prediction performance of new algorithms.

Acknowledgements

We are grateful to Zhongli Filippo Hu for his contributions to the integration of the algorithms. We thank Malte Ludewig for his help and support during this research.

References

[1] Gediminas Adomavicius and YoungOk Kwon. Improving aggregate recommendation diversity using ranking-based techniques. *IEEE Transactions on Knowledge and Data Engineering*, 24(5):896–911, 2011.

[2] Gediminas Adomavicius and Jingjing Zhang. Impact of data characteristics on recommender systems performance. *ACM Transactions on Management Information Systems*, 3(1), 2012.

[3] Timothy G. Armstrong, Alistair Moffat, William Webber, and Justin 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.

[4] Zhao-quan Cai and Hui Hu. Session-aware music recommendation via a generative model approach. *Soft Computing*, 22(3):1023–1031, 2018.

[5] Pedro G. Campos, Fernando Díez, and Iván 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, 2014.

[6] Andy Cockburn, Pierre Dragicevic, Lonni Besançon, and Carl Gutwin. Threats of a replication crisis in empirical computer science. *Communications of the ACM*, 63(8):70–79, 2020.

[7] Maurizio Ferrari Dacrema, Simone Boglio, Paolo Cremonesi, and Dietmar Jannach. A troubling analysis of reproducibility and progress

in recommender systems research. *ACM Transactions on Information Systems*, 39, 2020.

[8] Diksha Garg, Priyanka Gupta, Pankaj Malhotra, Lovekesh Vig, and Gautam 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.

[9] Odd Erik Gundersen. The reproducibility crisis is real. *AI Magazine*, 41(3):103–106, Sep. 2020.

[10] Odd Erik Gundersen, Yolanda Gil, and David W. Aha. On Reproducible AI: Towards Reproducible Research, Open Science, and Digital Scholarship in AI Publications. *AI Magazine*, 39(3):56–68, Sep. 2018.

[11] Yupu Guo, Duolong Zhang, Yanxiang Ling, and Honghui Chen. A joint neural network for session-aware recommendation. *IEEE Access*, 8:74205–74215, 2020.

[12] Ruining He and Julian McAuley. Fusing similarity models with markov chains for sparse sequential recommendation. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 191–200, 2016.

[13] Ruining He and Julian J. McAuley. Fusing similarity models with markov chains for sparse sequential recommendation. In *IEEE 16th International Conference on Data Mining, ICDM '16*, pages 191–200, 2016.

[14] Balázs Hidasi and Alexandros 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.

[15] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. In *Proceedings International Conference on Learning Representations*, ICLR '16, 2016.

[16] Liang Hu, Longbing Cao, Shoujin Wang, Guandong Xu, Jian Cao, and Zhiping Gu. Diversifying personalized recommendation with user-session context. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, page 1858–1864, 2017.

[17] Liang Hu, Qingkui Chen, Haiyan Zhao, Songlei Jian, Longbing Cao, and Jian Cao. Neural cross-session filtering: Next-item prediction under intra- and inter-session context. *IEEE Intelligent Systems*, 33(6):57–67, Nov 2018.

[18] Dietmar Jannach, Lukas Lerche, and Michael Jugovac. Adaptation and evaluation of recommendations for short-term shopping goals. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, pages 211–218, 2015.

[19] Dietmar Jannach and Malte Ludewig. When recurrent neural networks meet the neighborhood for session-based recommendation. In *In Pro-*

ceedings of the Eleventh ACM Conference on Recommender Systems, RecSys '17, pages 306–310, 2017.

[20] Dietmar Jannach, Malte Ludewig, and Lukas 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.

[21] Dietmar Jannach, Gabriel De Souza P. Moreira, and Even Oldridge. Why are deep learning models not consistently winning recommender systems competitions yet? In *ACM RecSys Challenge Workshop*, Online, 2020.

[22] Santosh Kabbur, Xia Ning, and George Karypis. FISM: Factored item similarity models for top-n recommender systems. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '13*, pages 659–667, 2013.

[23] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 197–206, Nov 2018.

[24] Yehuda Koren. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, page 447–456, 2009.

[25] Lukas Lerche, Dietmar Jannach, and Malte Ludewig. On the value of reminders within e-commerce recommendations. In *Proceedings of*

the 2016 Conference on User Modeling Adaptation and Personalization, UMAP '16, page 27–35, 2016.

[26] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. Neural attentive session-based recommendation. In *In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 1419–1428, 2017.

[27] Malte Ludewig and Dietmar Jannach. Evaluation of session-based recommendation algorithms. *User-Modeling and User-Adapted Interaction*, 28(4–5):331–390, 2018.

[28] Malte Ludewig, Noemi Mauro, Sara Latifi, and Dietmar 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, page 462–466, 2019.

[29] Malte Ludewig, Noemi Mauro, Sara Latifi, and Dietmar Jannach. Empirical analysis of session-based recommendation algorithms. *User Modeling and User-Adapted Interaction*, forthcoming, 2020.

[30] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. Statistical and machine learning forecasting methods: Concerns and ways forward. *PloS one*, 13(3), 2018.

[31] Bamshad Mobasher, Honghua Dai, Tao Luo, and Miki 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.

[32] James Ritchie Norris. *Markov Chains.* Cambridge University Press, Cambridge, 1997.

[33] Tu Minh Phuong, Tran Cong Thanh, and Ngo Xuan Bach. Combining user-based and session-based recommendations with recurrent neural networks. In *In International Conference on Neural Information Processing*, ICONIP 2018, pages 487–498, 2018.

[34] Tu Minh Phuong, Tran Cong Thanh, and Ngo Xuan Bach. Neural session-aware recommendation. *IEEE Access*, 7:86884–86896, 2019.

[35] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. Sequence-aware recommender systems. *ACM Computing Surveys*, 51:1–36, 2018.

[36] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, page 130–137, 2017.

[37] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence, UAI '09*, pages 452–461, 2009.

[38] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the Web Conference, WWW '10*, pages 811–820, 2010.

[39] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John 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.

[40] Massimiliano Ruocco, Ole Steinar Lillestøl Skrede, and Helge Langseth. Inter-session modeling for session-based recommendation. In *Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems*, DLRS 2017, page 24–31, 2017.

[41] Guy Shani and Asela Gunawardana. *Evaluating Recommendation Systems*, pages 257–297. Springer US, Boston, MA, 2011.

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

[43] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 1441–1450, 2019.

[44] Panagiotis Symeonidis, Lidija Kirjackaja, and Markus Zanker. Session-aware news recommendations using random walks on time-evolving heterogeneous information networks. *User Modeling and User-Adapted Interaction*, 30:727–755, 03 2020.

[45] Jiaxi Tang and Ke 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.

[46] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, and Xueqi Cheng. Learning hierarchical representation model for nextbasket recommendation. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, page 403–412, 2015.

[47] Shoujin Wang, Longbing Cao, Yan Wang, Quan Z. Sheng, Mehmet Orgun, and Defu Lian. A survey on session-based recommender systems, 2020.

[48] Teng Xiao, Shangsong Liang, and Zaiqiao Meng. Hierarchical neural variational model for personalized sequential recommendation. In *In The World Wide Web Conference*, WWW '19, pages 3377–3383, 2019.

[49] Wei Yang, Kuang Lu, Peilin Yang, and Jimmy 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.

[50] Haochao Ying, Fuzhen Zhuang, Fuzheng Zhang, Yanchi Liu, Guandong Xu, Xing Xie, Hui Xiong, and Jian Wu. Sequential recommender system based on hierarchical attention network. In *Proceedings of the 27th In-*

ternational Joint Conference on Artificial Intelligence, IJCAI'18, page 3926–3932. AAAI Press, 2018.

Appendix A. Hyperparameter ranges and optimal values

Table A.11: Hyperparameter space for simple methods.

Algorithm/Extension	Hyperparameter	Type	Range	Steps
SR	steps	Integer	2 - 15	14
		Integer	20, 25, 30	
	weighting	Categorical	linear, div, quadratic, log	
VSKNN	k	Integer	50, 100, 500, 1000, 1500	
	sample_size	Integer	500, 1000, 2500, 5000, 10000	
	weighting	Categorical	same, div, linear, quadratic, log	
	weighting_score	Categorical	same, div, linear, quadratic, log	
	idf_weighting	Boolean/Integer	False, 1, 2, 5, 10	
STAN	k	Integer	100, 200, 500, 1000, 1500, 2000	
	sample_size	Integer	1000, 2500, 5000, 10000	
	lambda_spw	Real	0.00001, 0.4525, 0.905, 1.81, 3.62, 7.24	
	lambda_snh	Real	2.5, 5, 10, 20, 40, 80, 100	
	lambda_inh	Real	0.00001, 0.4525, 0.905, 1.81, 3.62, 7.24	
VSTAN	k	Integer	100, 200, 500, 1000, 1500, 2000	
	sample_size	Integer	1000, 2500, 5000, 10000	
	similarity	Categorical	cosine, vec	
	lambda_spw	Real	0.00001, 0.4525, 0.905, 1.81, 3.62, 7.24	
	lambda_snh	Real	2.5, 5, 10, 20, 40, 80, 100	
	lambda_inh	Real	0.00001, 0.4525, 0.905, 1.81, 3.62, 7.24	
	lambda_ipw	Real	0.00001, 0.4525, 0.905, 1.81, 3.62, 7.24	
	lambda_idf	Boolean/Integer	False, 1, 2, 5, 10	
BOOST (For SR, VSKNN, STAN and VSTAN) Tune together with other hyperparameters of the algorithm	boost_own_sessions	Real	0.1-3.9	20
EXTEND (For VSKNN, STAN and VSTAN) Tune together with other hyperparameters of the algorithm	extend_session_length	Integer	1 - 25	25
REMIND (For VSKNN, STAN and VSTAN) Tune with the optimal set of hyperparameters of the algorithm	reminders	Boolean	True	
	remind_strategy	Categorical	hybrid	
	remind_sessions_num	Integer	1 - 10	10
	weight_Rel	Integer	1 - 10	10
	weight_IRec	Integer	0 - 9	10
	weight_SSim	Integer	0 - 9	10
REMIND (For SR) Tune with the optimal set of hyperparameters of the algorithm	reminders	Boolean	True	
	remind_strategy	Categorical	hybrid	
	remind_sessions_num	Integer	1 - 10	10
	weight_Rel	Integer	1 - 10	10
	weight_IRec	Integer	0 - 9	10

For STAN and VSTAN, we used the same hyperparameters space for lambda_spw, lambda_inh, and lambda_ipw on all datasets. However, ranges for these hyperparameters can be set for each dataset separately based on its mean average length of the sessions. Ranges can be set as {0.785, 1.57, 3.14, 6.28, 12.56} for the RETAIL dataset, {0.7025, 1.405, 2.81, 5.62, 11.24} for the XING dataset, {1.03, 2.06, 4.12, 8.24, 16.48} for the COSMETICS dataset, and {0.99, 1.98, 3.96, 7.92, 15.84} for the LASTFM dataset. This will lead to slightly different results.

Table A.12: Hyperparameter search space for neural methods.

Algorithm	Fixed Hyperparameter Values	Hyperparameter	Type	Range	Steps
GRU4REC	layer_size = 100	learning_rate	Real	0.01 - 0.1	10
			Real	0.1 - 0.5	5
		momentum	Real	0 - 0.9	10
		loss	Categorical	bpr-max, top1-max	
		final_act	Categorical	elu-0.5, linear	
		dropout_p_hidden	Real	0 - 0.9	10
		constrained_embedding	Boolean	True, False	
GRU4REC_R	The optimal set of hyperparameters of GRU4REC	remind_sessions_num	Integer	1 - 10	10
	reminders = True	weight_Rel	Integer	1 - 10	10
	remind_strategy = hybrid	weight_IRec	Integer	0 - 9	10
NARM	epochs = 20	factors	Integer	50, 100	
	layer_size = 100	learning_rate	Real	0.01-0.001	10
			Real	0.001-0.0001	10
NARM_R	The optimal set of hyperparameters of NARM	remind_sessions_num	Integer	1 - 10	10
	reminders = True	weight_Rel	Integer	1 - 10	10
	remind_strategy = hybrid	weight_IRec	Integer	0 - 9	10
HGRU4REC	session_layers = 100	user_propagation_mode	Categorical	init, all	
	user_layers = 100	learning_rate	Real	0.01-0.1	10
	loss = top1		Real	0.1-0.5	5
	(as in the original paper)	momentum	Real	0 - 0.9	10
		final_activation	Categorical	linear, relu, tanh	
		dropout_p_hidden_usr	Real	0 - 0.9	10
		dropout_p_hidden_ses	Real	0 - 0.9	10
		dropout_p_init	Real	0 - 0.9	10
		batch_size	Integer	50, 100	
IIRNN	max_epoch = 100	dropout_pkeep	Real	0.1 - 1	10
	for RETAIL, LASTFM and COSMETICS	learning_rate	Real	0.01 - 0.001	10
	max_epoch = 20 for XING		Real	0.001 - 0.0001	10
	because of the computational complexity	embedding_size	Integer	50, 100	
		max_session_representations	Integer	1, 5, 10, 15, 20	
		use_last_hidden_state	Boolean	True, False	
SHAN	global_dimension = 100	lambda_uv	Real	0.01, 0.001, 0.0001	
	epochs = 100	lambda_a	Integer	1, 10, 50	
	(as in the original paper)				
NCSF		max_nb_his_sess	Integer	0, 1, 2, 5, 10	
		att_alpha	Real	0.01, 0.1, 1, 10	
		window_sz	Integer	1 - 10	10
NSAR	epochs = 20	learning_rate	Real	0.001 - 0.01	10
	keep_pr = 0.25		Real	0.01 - 0.05	5
	(as in the original paper)	hidden_units	Integer	50, 100	
	batch_size = 64				
	for RETAIL, LASTFM and COSMETICS				
	(as in the original paper)				
	batch_size = 32 for XING				
	because of memory limitations				

Table A.13: Optimal hyperparameters for simple methods.

Algorithm	Hyperparameter	RETAIL	XING	COSMETICS	LASTFM
SR	steps	15	25	15	8
	weighting	quadratic	quadratic	div	quadratic
SR with extensions	**best extension**	SR_BR	SR_BR	SR_BR	SR_B
	steps	12	30	15	20
	weighting	quadratic	quadratic	div	quadratic
	boost_own_sessions	3.1	1.9	3.7	3.1
	reminders	True	True	True	-
	remind_strategy	hybrid	hybrid	hybrid	-
	remind_sessions_num	2	6	9	-
	weight_base	5	8	8	-
	weight_IRec	3	4	3	-
VSKNN	k	50	100	100	50
	sample_size	500	500	10000	500
	weighting	log	log	quadratic	quadratic
	weighting_score	linear	quadratic	div	quadratic
	idf_weighting	10	10	10	5
VSKNN with extensions	**best extension**	VSKNN_EBR	VSKNN_R	VSKNN_EBR	VSKNN_EB
	k	1500	100	1500	50
	sample_size	1000	500	10000	500
	weighting	log	log	quadratic	quadratic
	weighting_score	linear	quadratic	div	quadratic
	idf_weighting	1	10	10	1
	extend_session_length	8	-	2	3
	boost_own_sessions	0.1	-	0.9	2.5
	reminders	True	True	True	-
	remind_strategy	hybrid	hybrid	hybrid	-
	remind_sessions_num	4	8	10	-
	weight_base	8	2	9	-
	weight_IRec	1	1	2	-
	weight_SSim	1	0	3	-
STAN	k	1500	100	500	100
	sample_size	2500	10000	2500	10000
	lambda_spw	0.905	0.4525	0.905	0.00001
	lambda_snh	100	80	40	80
	lambda_inh	0.4525	0.4525	0.4525	3.62
STAN with extensions	**best extension**	STAN_ER	STAN_R	STAN_EBR	STAN_EBR
	k	200	100	1500	100
	sample_size	1000	10000	5000	2500
	lambda_spw	0.905	0.4525	0.905	0.00001
	lambda_snh	100	80	100	100
	lambda_inh	0.905	0.4525	7.24	7.24
	extend_session_length	2	-	2	17
	boost_own_sessions	-	-	1.9	2.7
	reminders	True	True	True	True
	remind_strategy	hybrid	hybrid	hybrid	hybrid
	remind_sessions_num	9	3	4	3
	weight_base	10	10	10	5
	weight_IRec	3	2	1	0
	weight_SSim	2	1	1	6
VSTAN	k	200	1500	500	1000
	sample_size	5000	10000	1000	5000
	similarity	vec	cosine	cosine	cosine
	lambda_spw	1.81	3.62	3.62	1.81
	lambda_snh	40	20	80	100
	lambda_inh	0.905	0.4525	0.4525	1.81
	lambda_ipw	0.905	0.4525	0.905	0.0001
	lambda_idf	False	10	False	False
VSTAN with extensions	**best extension**	VSTAN_EBR	VSTAN_R	VSTAN_EBR	VSTAN_EB
	k	2000	1500	500	1000
	sample_size	10000	10000	1000	10000
	similarity	cosine	cosine	cosine	cosine
	lambda_spw	0.905	3.62	0.905	0.4525
	lambda_snh	80	20	80	100
	lambda_inh	1.81	0.4525	0.4525	3.62
	lambda_ipw	3.62	0.4525	3.62	0.4525
	lambda_idf	5	10	1	5
	extend_session_length	5	-	1	7
	boost_own_sessions	0.1	-	3.1	3.7
	reminders	True	True	True	-
	remind_strategy	hybrid	hybrid	hybrid	-
	remind_sessions_num	2	3	5	-
	weight_base	6	9	7	-
	weight_IRec	2	1	1	-
	weight_SSim	0	5	0	-

Table A.14: Optimal hyperparameters for neural methods.

Algorithm	Hyperparameter	RETAIL	XING	COSMETICS	LASTFM
GRU4REC	learning_rate	0.08	0.05	0.03	0.04
	momentum	0.1	0.6	0.3	0.1
	loss	top1-max	top1-max	bpr-max	bpr-max
	final_act	linear	elu-0.5	linear	linear
	dropout_p_hidden	0.7	0.8	0.7	0
	constrained_embedding	True	True	True	False
GRU4REC_R	learning_rate	0.08	0.05	0.03	0.04
	momentum	0.1	0.6	0.3	0.1
	loss	top1-max	top1-max	bpr-max	bpr-max
	final_act	linear	elu-0.5	linear	linear
	dropout_p_hidden	0.7	0.8	0.7	0
	constrained_embedding	True	True	True	False
	reminders	True	True	True	True
	remind_strategy	hybrid	hybrid	hybrid	hybrid
	remind_sessions_num	3	3	2	4
	weight_base	9	9	7	4
	weight_IRec	2	4	3	0
NARM	factors	50	100	100	100
	learning_rate	0.01	0.007	0.007	0.007
NARM_R	factors	50	100	100	100
	learning_rate	0.01	0.007	0.007	0.007
	reminders	True	True	True	True
	remind_strategy	hybrid	hybrid	hybrid	hybrid
	remind_sessions_num	4	7	3	6
	weight_base	10	7	7	3
	weight_IRec	7	3	6	0
HGRU4REC	user_propagation_mode	all	all	init	all
	learning_rate	0.06	0.08	0.04	0.09
	momentum	0.3	0.6	0.5	0.5
	final_act	linear	tanh	linear	linear
	dropout_p_hidden_usr	0.4	0.8	0.5	0.7
	dropout_p_hidden_ses	0.3	0	0.3	0.1
	dropout_p_init	0.4	0.6	0.1	0.3
	batch_size	50	100	50	50
IIRNN	dropout_pkeep	0.4	0.6	0.5	0.6
	learning_rate	0.002	0.002	0.001	0.001
	embedding_size	100	100	100	100
	max_session_representations	15	1	1	20
	use_last_hidden_state	False	False	True	True
SHAN	lambda_uv	0.01	0.01	0.01	0.01
	lambda_a	1	1	1	10
NCSF	max_nb_his_sess	5	0	0	0
	att_alpha	10	10	1	1
	window_sz	2	2	3	1
NSAR	learning_rate	0.01	0.004	0.007	0.003
	hidden_units	100	100	100	100

Table A.15: Running times on the RETAIL dataset.

	Training Time (s)	Prediction Time (ms)
SR	0.42	3.24
VSKNN	0.09	4.42
STAN	0.08	4.29
VSTAN	0.09	4.55
SR_BR	0.78	12.99
VSKNN_EBR	0.24	20.85
STAN_ER	0.19	24.42
VSTAN_EBR	0.24	14.99
GRU4REC	12.95	4.00
NARM	102.73	5.50
GRU4REC_R	10.46	13.61
NARM_R	93.90	15.07
HGRU4REC	23.92	4.07
IIRNN	1307.41	87.34
NCSF	82.54	25.81
NSAR	475.46	21.51
SHAN	1661.60	12.32

Table A.16: Running times on the COSMETICS dataset.

	Training Time (s)	Prediction Time (ms)
SR	0.62	2.64
VSKNN	0.14	4.45
STAN	0.13	6.06
VSTAN	0.14	6.81
SR_BR	1.55	8.14
VSKNN_EBR	0.33	14.92
STAN_EBR	0.32	18.37
VSTAN_EBR	0.33	12.42
GRU4REC	13.56	3.86
NARM	159.79	5.62
GRU4REC_R	11.74	9.32
NARM_R	135.13	21.91
HGRU4REC	36.74	3.85
IIRNN	1034.02	59.08
NCSF	40.59	10.66
NSAR	387.43	11.26
SHAN	2344.50	11.58