entropy

MDPI

Article

Comprehensive Empirical Evaluation of Deep Learning Approaches for Session-Based Recommendation in E-Commerce

Mohamed Maher [1], Perseverance Munga Ngoy [1], Aleksandrs Rebriks [1], Cagri Ozcinar [1], Josue Cuevas [2], Rajasekhar Sanagavarapu [2] and Gholamreza Anbarjafari [1,3,4,*]

[1] iCV Lab, Institute of Technology, University of Tartu, 51009 Tartu, Estonia
[2] Machine Learning Group, Big Data Department, Rakuten Inc., Tokyo 158-0094, Japan
[3] PwC Advisory, 00180 Helsinki, Finland
[4] Institute of Higher Education, Yildiz Technical University, Yildiz, Beşiktaş District, Istanbul 34349, Turkey
* Correspondence: shb@ut.ee; Tel.: +372-737-4855

Abstract: Boosting the sales of e-commerce services is guaranteed once users find more items matching their interests in a short amount of time. Consequently, recommendation systems have become a crucial part of any successful e-commerce service. Although various recommendation techniques could be used in e-commerce, a considerable amount of attention has been drawn to session-based recommendation systems in recent years. This growing interest is due to security concerns over collecting personalized user behavior data, especially due to recent general data protection regulations. In this work, we present a comprehensive evaluation of the state-of-the-art deep learning approaches used in the session-based recommendation. In session-based recommendation, a recommendation system counts on the sequence of events made by a user within the same session to predict and endorse other items that are more likely to correlate with their preferences. Our extensive experiments investigate baseline techniques (e.g., nearest neighbors and pattern mining algorithms) and deep learning approaches (e.g., recurrent neural networks, graph neural networks, and attention-based networks). Our evaluations show that advanced neural-based models and session-based nearest neighbor algorithms outperform the baseline techniques in most scenarios. However, we found that these models suffer more in the case of long sessions when there exists drift in user interests, and when there are not enough data to correctly model different items during training. Our study suggests that using the hybrid models of different approaches combined with baseline algorithms could lead to substantial results in session-based recommendations based on dataset characteristics. We also discuss the drawbacks of current session-based recommendation algorithms and further open research directions in this field.

Keywords: session-based recommendation; information systems; deep learning; evaluation; E-commerce

check for updates

Citation: Maher, M.; Ngoy, P.M.; Rebriks, A.; Ozcinar, C.; Cuevas, J.; Sanagavarapu, R.; Anbarjafari, G. Comprehensive Empirical Evaluation of Deep Learning Approaches for Session-Based Recommendation in E-Commerce. *Entropy* **2022**, *24*, 1575. https://doi.org/10.3390/e24111575

Academic Editor: Sotiris Kotsiantis

Received: 29 August 2022
Accepted: 26 October 2022
Published: 31 October 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Most e-commerce services use recommendation systems to help their customers find their items of interest based on their navigation behavior through these services. Recommendation systems are considered a category of information-filtering systems that aim to predict user preferences based on their behavior. They have become a crucial part of any successful business that helps satisfy user needs and boost the business sales volume [1]. Recommendation systems have been used in a wide range of domains including images [2], music [3], videos [4], and even news [5] recommendations.

Various types of recommendation systems have been proposed in the literature, categorized as time-aware and session-based recommendation systems. The former can adapt to the temporal dynamics and user preferences drift over time [6,7] and recommendation systems based on social information datasets [8,9]. The latter relies on the user navigation

behavior and sequence of actions and mouse clicks on different items solely to recommend the items that match the user's interests.

In recent years, more attention has been paid to session-based recommendation due to the security policies concerning the collection of personalized user behavior data. In particular, a greater research focus has been directed towards anonymous session-based recommendation systems. The main reason for this research focus is to comply with the recent (*GDPR*) rules that make the collection of personalized data about users more challenging to protect the user's privacy [10]. Furthermore, it is not easy to collect enough long-term user profile data to reliably recommend the following items. Figure 1 shows an example of a session-based recommendation where the user has a stream of click events on multiple items. The recommendation system tries to predict the next items to be viewed by the same user only based on the information made available during the same session.

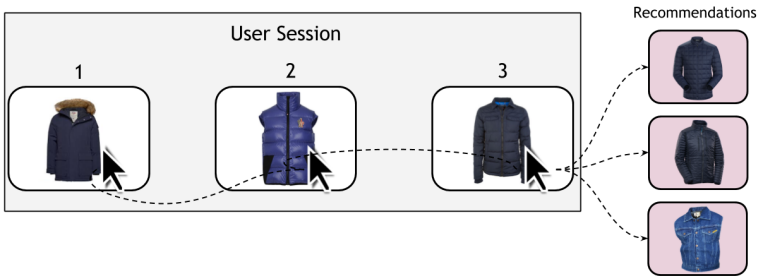

Figure 1. An example of session-based recommendation. Three different items are clicked by the user. The recommendation system predicts other candidate items that can be subsequently viewed by the same user.

Deep neural networks are a subset of machine learning technologies that have attracted significant attention in the past decade. Such techniques have achieved outstanding performance in a wide range of domains, including natural language processing, medical diagnosis, speech recognition, and computer vision. In practice, the main advantage of deep learning over traditional machine learning techniques is their automatic feature extraction ability. This allows learning complex functions to be mapped from the input space to the output space without human intervention [11]. Recently, different approaches have been proposed to use deep neural networks in recommendation systems [12,13]. In particular, different deep learning models were used for modeling the sequence of the user navigation behavior in online services to be used in the next item recommendation [14–17]. These works showed a competitive performance compared to traditional approaches such as sequential pattern mining techniques, nearest neighbor algorithms, and traditional Markov models [18].

Few studies have been conducted to evaluate session-based algorithms. Jannach et al. compared the heuristics-based nearest neighbor baseline algorithm with a basic recurrent neural network (RNN) [19]. The results of this study showed that deep learning methods fall behind basic algorithms such as neighborhood methods. However, during the last couple of years, many advancements have been proposed using deep-learning in session-based recommendation, leading to the rise of several neural-based architectures. Ludewig et al. [18], for instance, conducted a study to compare many baseline algorithms in session-based recommendation using four datasets in the e-commerce field and four others in music and playlists recommendation. Although this study included only a single neural-based model and lacked the evaluations of different deep-learning approaches in the literature, it was recently extended to include state-of-the-art deep learning models [20]. However, the empirical evaluation conducted in [18,20] by training models on full datasets makes it difficult to understand the drawbacks of each model and why exactly a specific model outperforms others in a particular dataset.

In this paper, we extend previous studies [18,20,21] that compared the overall performance of 11 simple algorithms and 6 deep-learning models on four e-commerce datasets. In this work, we focus on studying the effect of varying the characteristics of a dataset on the performance of each model. In particular, our main contributions are as follows:

- We carry out an extensive evaluation and benchmarking of the state-of-the-art neural-based approaches in session-based recommendation, including recurrent neural networks, convolutional neural networks, and attention-based networks along with a group of the most popular baseline techniques in the recommendation field such as nearest neighbors, frequent pattern mining, and matrix factorization.
- We evaluate the performance of the different models based on various characteristics of training and test dataset splits obtained from four different e-commerce benchmark datasets, namely *RECSYS* (http://2015.recsyschallenge.com/, accessed on 1 August 2022), *CIKMCUP* (https://cikm2016.cs.iupui.edu/cikm-cup/, accessed on 1 August 2022), *RETAIL ROCKET* (https://www.kaggle.com/retailrocket/ecommerce-dataset, accessed on 1 August 2022), and *TMALL* (https://www.tmall.com/, accessed on 1 August 2022)
- Our experiments elaborate on the evaluation process based on various dataset characteristics. Hence, we divide the datasets according to the values of various characteristics such as session length, item frequency, and data sizes. These experiments revealed some insights that could help understand when some models are poorly performing and open new research horizons of improvements that are needed for each model, which were difficult to deduce from the previous studies.
- An interpretable decision tree model is used to accurately recommend the best-performing model according to the dataset characteristics.
- Current drawbacks of session-based recommendation systems are discussed with proposed solutions to overcome these issues, which could yield better results in some domains.

We divide our benchmarking study into separate sets of experiments each aiming to answer a different research question. First, we evaluated the performance of different models against different session lengths and frequencies of items. Second, we investigated the effect of the recency of the collected data on the models' performance, which could help avoid data leakage problems and deceiving accuracy during training. Third, the effect of the training data size is evaluated for different models. Finally, we present a comparison of different approaches in terms of time and memory resource consumption during both training and inference. The aim of this study was mainly to determine and understand the main characteristics of the datasets that profoundly affect different models' performances by carrying out a micro-analysis evaluation for the session-based algorithms on real-world e-commerce datasets. This study could help improve the selection of the recommendation algorithm according to the target dataset and highlight the weaknesses of different models for further improvement.

The paper is organized as follows. In Section 2, a short survey of session-based recommendation systems is discussed. Section 3 presents a detailed description of different algorithms and models evaluated in our experiments. Section 4 describes the experiment setup and the research questions to be answered, and Section 5 shows the results and discussion of the evaluation experiments. Finally, in Section 6, the main insights of our study are summarized in addition to the thoughts of future research directions.

2. Review of Deep Learning Approaches in Session-Based Recommendation

The session-based recommendation is a particular type of sequence-aware recommendation that is a general class of recommendation systems. The decisions made by these systems are mainly based on the short-term user intention defined by a session. This session is represented by a set of user–item interaction pairs in a short period of time. Furthermore, various types of attributes can characterize these interactions, such as user attributes (e.g., gender and age), item attributes (e.g., color and size), and action types (e.g., add-to-cart

and add-to-wish-list). The input of these recommendation systems is a chronologically ordered set of user–item actions and the output is a score-list of the ranking of items based on the likelihood that user preferences match these items [22]. Even though e-commerce is the most critical application for the session-based recommendation, there are many other applications such as recommendations for music playlists, films, and online course [23].

Early research works tackled session-based recommendation problems with the nearest neighbors and frequent pattern mining techniques [24]. However, these works are instance-based algorithms that take a significant amount of time to make predictions. Therefore, they are not suitable for real-time use cases, such as e-commerce. Later on, other research works proposed using more advanced techniques, such as Markov chain models in sequence modeling [25,26]. The problem of the state-space explosion in Markov models was treated using the attributes of some items to limit the space of the next items to be recommended [27]. Additionally, classical matrix factorization techniques were combined with Markov chains in different variations and applied in a wide range of domains as in [28,29].

In recent years, different deep learning approaches have been adopted in session-based recommendation. The main advantage of deep learning approaches is their ability to automatically extract features. This advantage allows learning complex functions to be mapped from the input space to the output space without human intervention [11]. For example, a neural-based model named GRU4Rec used gated recurrent units (GRUs) in RNNs to predict the next item to be clicked by the user [14]. The model was trained by minimizing the loss functions that include pairwise losses comparing the target item score with the maximal score among negative samples. The likelihood of these samples is taken into account in proportion to the target item's maximal score. The used losses showed excellent performance by correctly ranking the predicted items and overcoming the vanishing gradient problem in RNNs [30].

The GRU4Rec architecture was further extended by using a modified version of the original negative sampling approach, where the likelihood score of the next recommended item is calculated for a subset of items as it would be impractical to do it for the whole list of items [30]. The new sampling method uses additional negative samples shared by all the session sequences within the same mini-batch. Additionally, it updates a small percentage of the network weights for each mini-batch to make the training process faster. These samples were chosen based on the items' popularity, which gives more chances to include most of the high scoring negative examples. This approach leads to excellent improvement in the performance of the model. Furthermore, the same architecture was adopted to support multiple item features instead of unique identifiers only in a parallel training scheme. It was evaluated against item K-nearest neighbors showing a good improvement [31].

Furthermore, Quadrana et al. proposed a method for adapting RNN in personalized session-based recommendation with cross-session information transfer among user sessions using a hierarchical RNN model such that the output hidden state from the network for a particular session is passed as input to a higher level RNN for the next session of the same user [32]. A hybrid architecture of two RNNs was proposed for a personalized session-based recommendation that mainly aims to target the session cold-start problem by learning from the user's recent personal sessions [33]. Convolutional neural networks (CNNs) have also been used in session-based recommendation. In particular, Tuan et al. used a 3D-CNN with character-level encoding to combine session clicks with the textual descriptions of the items to generate recommendations [34]. Similarly, a generative CNN was proposed by embedding the clicked items into a two-dimensional matrix and treated as input images to the CNN [35]. Graph neural networks were recently used to capture complex transitions among items after modeling the sequence of events of a session as graph-structured data without adequate user behavior in a session [8]. Wang et al. proposed a novel framework using two parallel memory encoders to make use of the information of

collaborative neighborhood sessions in addition to the current session information followed by a selective fusion of both encoders' output [36].

After discovering the attention concept in neural networks which leads to a great improvement in terms of neural machine translation tasks [37,38], attention networks were widely adopted in the session-based recommendation [16,39,40]. For instance, a hybrid encoder with attention is used to model user sequential behavior [39], which outperforms long-term memory models such as GRU4Rec [39]. Furthermore, a short term attention priority model was introduced such that attention weights are computed from the total session context and enhanced by the current user's interest represented by the last clicked item [16]. Additionally, Sun et al. adopted the current state-of-the-art BERT transformer network [41], widely used in the natural language processing domain, in personalized session-based recommendation [42]. Most neural-based solutions, in the session-based recommendation, generate a static representation for users' long-term interests. Such representation might be an issue as its importance in predicting the next recommended item is dynamic and also related to short-term preferences. Hence, a co-attention network was proposed to recognize the dynamic interaction between the user's long and short-term interests to generate a co-dependent representation of the users' interests [40]. However, the usage of the transformer networks in generalized session-based recommendations with the incorporation of item features are still open research areas. Table 1 summarizes current state-of-the-art neural network architectures for personalized/non-personalized session-based recommendation.

Table 1. Summary of current state-of-the-art neural session-based recommendation architectures.

Model Name	Authors Date	Personalized Recommendation	Main Layers of Architecture	Attention/ Memory	Supports Item Features	Framework	Open Source
Item2Vec [43]	Barkan et al., 2016	X	Feed-forward network	X	X	Genism https://github.com/Bekyilma/Recommendation-based-on-sequence- accessed on 1 August 2022	✓
GRU4Rec [14]	Hidasi et al., 2015	X	Recurrent layers network	X	X	Theano https://github.com/hidasib/GRU4Rec accessed on 1 August 2022	✓
P-GRU4Rec [31]	Hidasi et al., 2016	X	Recurrent layers network	X	✓	X	X
Conv3D4Rec [34]	Tuan et al., 2017	X	3D convolutional networks	X	✓	X	X
NARM [39]	Li et al., 2017	X	Recurrent layers network	✓	X	Theano https://github.com/lijingsdu/sessionRec_NARM accessed on 1 August 2022	✓
IIRNN [33]	Ruocco et al., 2017	✓	Recurrent layers network	X	X	Tensorflow https://github.com/olesls/master_thesis accessed on 1 August 2022	✓
HGRU4Rec [32]	Quadrana et al., 2017	✓	Recurrent layers network	X	✓	Theano https://github.com/mquad/hgru4rec accessed on 1 August 2022	✓
GRU4Rec+ [30]	Hidasi et al., 2018	X	Recurrent layers network	X	X	Theano https://github.com/hidasib/GRU4Rec accessed on 1 August 2022	✓
STAMP [16]	Liu et al., 2018	X	Feed-forward network	✓	X	Tensorflow https://github.com/uestcnlp/STAMP accessed on 1 August 2022	✓
SASRec [15]	Kang et al., 2018	✓	Feed-forward network	✓	X	Tensorflow https://github.com/kang205/SASRec accessed on 1 August 2022	✓

Table 1. *Cont.*

Model Name	Authors Date	Personalized Recommendation	Main Layers of Architecture	Attention/ Memory	Supports Item Features	Framework	Open Source
CASER [44]	Tang et al., 2018	✓	Convolutional layers network	X	X	Pytorch https://github.com/graytowne/caser_pytorchaccessed on 1 August 2022	✓
NextItNet [35]	Yuan et al., 2019	X	Dilated convolutional layers network	X	X	Tensorflow (https://github.com/fajieyuan/nextitnet) accessed on 1 August 2022	✓
SRGNN [17]	Wu et al., 2019	X	Graph neural network	✓	X	Tensorflow/Pytorch (https://github.com/CRIPAC-DIG/SR-GNN) accessed on 1 August 2022	✓
CSRM [36]	Wang et al., 2019	X	Recurrent layers network	✓	X	Tensorflow (https://github.com/wmeirui/CSRM_SIGIR2019) accessed on 1 August 2022	✓
BERT4Rec [42]	Sun et al., 2019	✓	Transformer-based network	✓	X	Tensorflow (https://github.com/FeiSun/BERT4Rec) accessed on 1 August 2022	✓
DCN-SR [40]	Chen et al., 2019	X	Recurrent layers network	✓	X	X	X

3. Detailed Evaluated Approaches

In this section, all the algorithms covered in our evaluation study, ash shown in Figure 2, are explained in detail, and for the sake of simplicity, the following notation in Table 2 is used throughout.

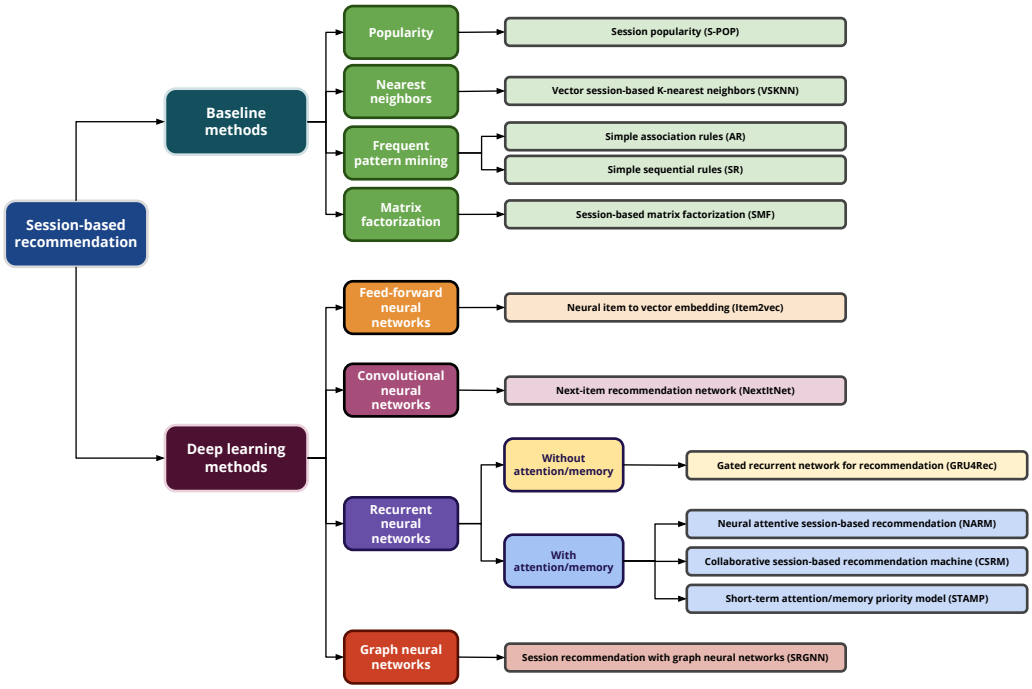

Figure 2. List of evaluated baseline and deep learning session-based recommendation methods.

Table 2. Notation of different variables used in explaining the evaluated methods.

Symbol	Description
$\mathcal{I}$	Set of available items
N	Total number of available items = $\lvert\mathcal{I}\rvert$
I_n	Item with index $n \in \mathcal{I}$ where $n \in N$
x_{i_t}	The ith click event in a session starting at time t
L_t	The length of session starting at time t
S_t	A session started at time t representing the sequence of clicked items $\{x_{1_t}, x_{2_t}, \ldots, x_{L_t}\}$
$1_{EQ}(x, y)$	$= 1$ if $x = y$, and 0 otherwise
$\mathcal{S}_D, \mathcal{S}_{TR}, \mathcal{S}_{TE}$	Set of general dataset D, training and testing sets sessions, respectively
$dis(j, k)$	Distance between items at indices j and k in the session click stream
$sim(S_i, S_j)$	Similarity distance between sessions S_i and S_j
$1_{IN}(x, \mathbf{Y})$	$= 1$ if x is one of elements in vector Y, and 0 otherwise
$r_{IN}(x, \mathbf{Y})$	$=$ rank of x if it is one of the elements in vector Y, and 0 otherwise
$W_t(S_t)$	A weighting function for items clicked in session S_t
E_{S_t}, E_i	Embedding vector for session S_t, or item I_i, respectively
$_K(\hat{\mathbf{Y}})$	Index of items with top K predicted scores in vector $\mathbf{Y}$

3.1. Baseline Approaches

We selected a set of five baseline algorithms to be included in this study based on the previous study in [18]. In particular, our selection was based on two different criteria. First, we selected at least one method from each family of algorithms, which showed excellent performance at different session-based recommendation tasks. Second, we chose the method with the best overall performance compared to other methods within the same family. Therefore, the selected algorithms are as follows: session-based popular products (S-POP) as a simple heuristic algorithm [45]; and simple association rules (AR) and simple sequential rules (SR) as representatives of frequent pattern mining algorithms [46]. Vector session-based K-nearest neighbors (VSKNN) [18] and session-based matrix factorization (SMF) [18] were selected from the nearest neighbors and factorization-based methods, respectively.

3.1.1. Session-Based Popular Products

S-POP is one of the most widely used baseline recommendation algorithms [45,47]. These algorithms make a recommendation based on the most frequent item viewed by the user in the current session. In short, if a user clicked on an item I_n multiple times during the same session, this reflects a clear sign of the user's interest in that item. Hence, recommending the same item to the user again is a reasonable decision. In some cases, the S-POP recommendation process is limited to the top popular K items while ignoring the rest of the items. This constraint ensures that the recommended items belong to the most popular ones among all users.

The score of a specific item I_n in a session S_t is computed as follows:

$$Score(I_n, S_t) = \sum_{i=1}^{L_t} 1_{EQ}(x_{i_t}, I_n). \quad (1)$$

3.1.2. Simplified Association Rules

Association rules (ARs) are a frequently used pattern mining approach that can capture the size for the frequency of patterns of events, N, and recommend the most frequent ones [46]. In the case of session-based recommendation, Ludewig et al. [18] used a simplified version of association rules of size $N = 2$ to have reasonable computational complexity. In their work, the occurrence of any two subsequent items (I_i, I_j) at the same session S was stored. During prediction, the last item viewed by the user, x_{L_t}, was used to find all the

candidate similar items by choosing the most frequent item pairs, (x_{L_t}, I_n) where $n \in N$. Therefore, an arbitrary item I_n was recommended if it has a score among the top predicted ones. This score is computed as follows:

$$Score(I_n, S_t) = \sum_{S_i \in \mathcal{S}_{TR}} \sum_{j=1}^{|L_i|} \sum_{k=1}^{|L_i|} 1_{EQ}(x_{L_t}, x_{j_i}).1_{EQ}(I_n, x_{k_i}). \quad (2)$$

3.1.3. Simplified Sequential Rules

Sequential rules are also a frequent pattern mining approach. Here, the order of the session events is taken into account in contrast with AR which depends on the support of the items only. A simplified form of sequential rules (SR) is used such that a rule is created between two items (I_i, I_j) when they appear in sequential events [21]. Each rule in SR is assigned a weight that is a function of the linear distance between the items (I_i, I_j) as in Equation (3). The rules between proximate events are assigned larger weights than rules between distant events. The scores of different items to be recommended can be evaluated using the following:

$$Score(I_n, S_t) = \sum_{S_i \in \mathcal{S}_{TR}} \sum_{j=2}^{|L_i|} \sum_{k=1}^{x-1} 1_{EQ}(x_{L_t}, x_{k_i}).1_{EQ}(I_n, x_{j_i}).dis(j,k), \quad (3)$$

where $dis(j,k) = (1 - 0.1(j-k))$ if $j - k < 10$ otherwise $dis(j,k) = 0$

3.1.4. Vector Multiplication Session-Based K-Nearest Neighbors

Nearest neighbor algorithms show excellent performance in session-based recommendation [21]. However, they have many different variant schemes which can be applied according to the domain type-like item-based nearest neighbors [48] which depends on predicting similar items to the last one viewed by the user. On the other hand, session-based nearest neighbors consider the viewed items in the whole session and try to find neighboring sessions with similar items to be used in predicting the next recommended items [24]. Ludewig et al. [18] evaluated the multiple variants of nearest neighbor algorithms. In their work, it has been shown that vector multiplication session-based K-nearest neighbors (VSKNN) outperformed pattern mining and matrix factorization methods in most evaluated datasets. Additionally, it has a competitive performance rivaling RNNs and even outperforms them in multiple datasets. VSKNN is considered to be one of the session-based nearest neighbors algorithms, where recent items clicked by the user take larger weights than older items. As such, more emphasis is given for the recent events made by the user. The score of an item I_n to be recommended for the next item is computed as

$$Score(I_n, S_t) = \sum_{S_i \in \mathcal{S}_{TR}} [sim(S_t, S_i).W_t(S_t)]\, 1_{IN}(I_n, S_i), \quad (4)$$

where the similarity distance, $sim(S_t, S_i)$, can be set to the cosine distance, and $W_t(S_t)$ is a weighting function of the items according to their positions in the session S_t. This weighting function usually gives higher weights to the recently clicked items [18].

3.1.5. Session-Based Matrix Factorization

SMF is a matrix factorization-based approach designed for the task of session-based recommendation [18]. This approach was inspired by the factorized personalized Markov chains [29,49] for sequential recommendation tasks. In SMF, classical matrix factorization and factorized Markov chains are combined with a hybrid approach. In particular, the latent user vector was replaced by an embedding vector that represents the current session. During the prediction process, the score of a candidate item is computed as the weighted sum of the whole session preferences and the sequential dynamics representing the transition

probability from the last clicked item by the user to the candidate item to be recommended by the model. We used the model implementation by Ludewig et al. [18]. The SMF showed a better performance than other factorization-based methods over multiple datasets.

3.2. Deep Learning Approaches

Many deep learning architectures were proposed in the literature for session-based recommendation. These architectures vary in the types of their layers. For instance, Hidasi et al. [14] presented the first study using RNNs with GRUs in session-based recommendation. Tuan et al. [34] and Yuan et al. [35] used convolutional networks in modeling the session context. Li et al. and Liu et al. [16,39] proposed different attention mechanisms to enhance the performance of RNNs. Recently, Wu et al. [17] exploited graph neural networks in session-based recommendation.

In our study, we limited the selection to only include the current state-of-the-art and well-cited architectures proposed in the range of the last four years, published in top tier venues, and that were implemented as open source. Additionally, we refined our list to select the models that can be used in making generalized (non-personal) predictions without the need to collect a personal user profile to easily comply with the GDPR requirements (Section 1). The final list of the chosen architectures includes the neural item embedding algorithm (Item2Vec) proposed by Barkan et al. [43], extended version of GRUs neural networks (GRU4Rec+) by Hidasi et al. [50], neural attentive network (NARM) by Li et al. [39], graph neural network proposed by Wu et al. [17], short-term attention priority network (STAMP) by Liu et al. [16], as well as convolutional generative network for session-based recommendation (NextItNet) [35], and collaborative neural network with parallel memory modules (CSRM) proposed by Wang et al. [36].

3.2.1. Neural Item to Vector Embedding

Barkan et al. introduced a conversion for the items into embedding vectors in a latent space based on the session context of clicked items. This idea is an adaptation of the *Word2Vec* algorithm that converts the words into a vector space in an efficient way that enhances neural machine translation task performance by having two close vectors for similar words used in the same context [51]. Similarly, Item2Vec uses the skip-gram with negative sampling neural word embedding to determine vector representations for different items that infer the relationship between an item and its surrounding items in a session. During the prediction phase, candidate items obtain scores according to the similarity distance between their embedding vectors and the average of the embedding vectors of the session items [43].

3.2.2. Gated Recurrent Neural Networks for Session-Based Recommendation

One of the first successful approaches for using RNNs in the recommendation domain is the GRU4Rec network [14]. An RNN with GRUs was used for the session-based recommendation. A novel training mechanism called session-parallel mini-batches is used in GRU4Rec, as shown in Figure 3. Each position in a mini-batch belongs to a particular session in the training data. The network finds a hidden state for each position in the batch separately, but this hidden state is kept and used in the next iteration at the positions when the same session continues with the next batch. However, it is erased at the positions of new sessions coming up with the start of the next batch. The network is always updated with the session beginning and used to predict the subsequent events.

The GRU4Rec architecture is composed of an embedding layer followed by multiple optional GRU layers, a feed-forward network, and a softmax layer for output score predictions for candidate items. The session items are one-hot-encoded in a vector representing all items' space to be fed into the network as input. On the other hand, a similar output vector is obtained from the softmax layer to represent the predicted ranking of items. Additionally, the authors designed two new loss functions, namely Bayesian personalized ranking (BPR) loss and regularized approximation of the relative rank of the relevant item (TOP1) loss.

BPR uses a pairwise ranking loss function by averaging the target item's score with several negative ones sampled in the loss value. TOP1 is the regularized approximation of the relative rank of the relevant item loss. Later, Hidasi et al. [30] extended their work by modifying the two-loss functions previously introduced by solving the issues of a vanishing gradient faced by TOP1 and BPR when the negative samples have a very low predicted likelihood that approaches zero. The newly proposed losses merge between the knowledge from the deep learning and the literature of learning to rank. The evaluation of the new extended version shows clear superiority over the older version of the network. Thus, we included the extended version of the GRU4Rec network, denoted by GRU4Rec+, in our evaluation study.

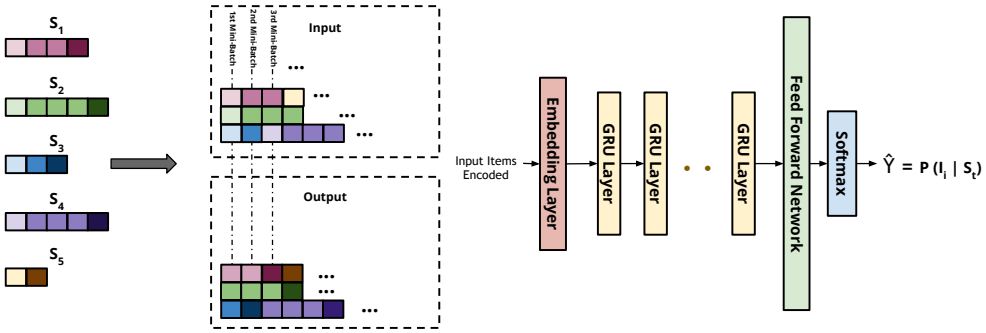

Figure 3. GRU4Rec model architecture [14].

3.2.3. Neural Attentive Session-Based Recommendation

NARM is one of the session-based recommendation systems based on sequence modeling using an attention mechanism [39]. The main advantage of this model is introducing a solution to long-term memory models such as GRU4Rec (Section 3.2.2). The model is characterized by hybrid encoders with an attention network to model the user sequential behavior and capture the main purpose of the session combined as a unified session representation.

NARM architecture has two types of encoders:

1. GRU network represents the global encoder, which takes the entire previous user interactions during the session as input and produces the user's sequential behavior as output.
2. Local encoder that is a GRU network similar to the global encoder. However, its role is to involve an item-level attention mechanism to allow the decoder to dynamically select a linear combination of different items from the input sequence, and focus more on important items that can capture the user's main purpose within a particular session.

Finally, both encoders' outputs are concatenated with each other to form an extended representation of the session. They are fed again into a bi-linear decoder along with item embedding vectors to compute the similarity score between the current session representation and candidate items to be used in ranking the items to be predicted next.

3.2.4. Short-Term Attention/Memory Priority Model

STAMP is one of the approaches that replaces complex recurrent computations in RNNs with self-attention layers [16]. The model presents a novel attention mechanism in which the attention scores are computed from the user's current session context and enhanced by the sessions' history. Thus, the model can capture the user interest drifts, especially during long sessions and outperform other approaches like GRU4Rec [14] that uses long term memory but still not efficient in capturing user drifts.

Figure 4 shows the model architecture where the input is two embedding vectors (E_L, E_{S_t}). The former denotes the embedding of the last item x_L clicked by the user in

the current session, which represents the short term memory of the user's interest. The latter represents their overall interest through the full session clicked items. The E_{S_t} vector is computed by averaging the items embedding vectors throughout the whole session memory $(x_1, x_2, \ldots, x_L)$. An attention layer is used to produce a real-valued vector E_a, where this layer is responsible for computing the attention weights corresponding to each item in the current session. As such, we avoid treating each item in the session as equally important and paying more attention to only related items, which improves the capturing of the drifts in the user interest. Both E_a and E_L flow into two multi-layer perceptron networks that are identical in shape but have separate independent parameters for feature abstraction. Finally, a trilinear composition function, followed by a softmax function, is used for the likelihood calculation of the available items to be clicked next by the user and to be used in the recommendation process.

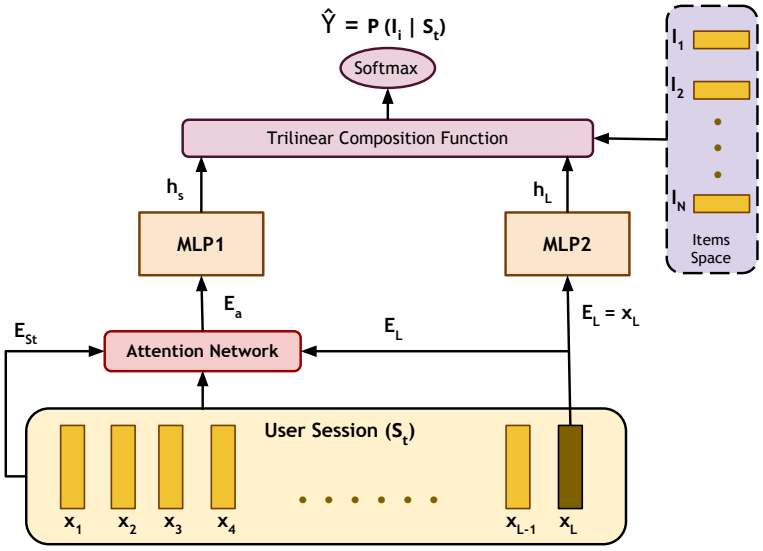

Figure 4. STAMP model architecture [16].

3.2.5. Simple Generative Convolutional Network

NextItNet was proposed to use convolutional neural networks in the session-based recommendation. The session made by a user is converted into a two-dimensional latent matrix and fed into convolutional neural network-like images [35].

NextItNet is considered as an extension of the recent convolutional sequence embedding recommendation model (Caser) by Tang et al. [44]. However, NextItNet addresses the two main limitations of applying CNNs in sequence modeling in Caser, which are obvious in long sessions. First, the items sequences in a session can have a variable length, which means that a large number of different size images are needed to represent a session. Consequently, fixed-size convolutional filters may fail in dealing with such cases. However, large filters with a filter width similar to the image width of an item inside the session sequence, and followed by max-pooling layers, are used to ensure that the produced feature maps have the same length. Second, these small filters are not able to find well-representing embedding vectors for the session items. In NextItNet, a huge number of inefficient convolutional filters are replaced with a series of one-dimensional dilated convolution layers. The dilated layers are responsible for increasing the receptive field and dealing with different session lengths instead of the standard 2D convolution layers. Thus, the max-pooling layers are omitted as they cannot distinguish the important features in the map if they occur once or multiple times while ignoring the position of these features. Additionally, NextItNet effectively makes use of the residual blocks in the recommendation systems, which can ease the optimization for much deeper networks than the shallow convolutional network in Caser that cannot model complex relations between items in a user session.

3.2.6. Session-Based Recommendation with Graph Neural Networks

SRGNN was introduced recently by Wu et al. [17]. The session sequences are modeled as a graph-structured data and the graph neural network (GNN) task is to capture the complex transitions among items. This architecture was proposed to solve mainly two problems with other approaches. First, most other models cannot estimate the user interest without adequate interactions in a session. Secondly, most of the models focus on single-way transitions between items and neglect transitions among the context instead.

Each session is modeled as a separate sub-graph. In this sub-graph, a node represents an item, and an edge represents a user interaction with that item. Session S_t in Figure 5 is shown as an example of session sub-graph. Each edge is assigned a normalized weight calculated by the division of the edge occurrence by the out-degree of that edge's starting node. Then, using an attention network, each session sub-graph is proceeded one by one through a gated GNN to produce an embedding vector for each node. The role of SRGNN is to capture the complex transitions in the session context and generate accurate corresponding item embedding vectors. This method can be adapted if the nodes of the items have multiple features such as price, color, size, and brand by concatenating them with the node embedding vector. Furthermore, the session embedding vector adds information about the session's local embedding vector defined by the last clicked item vector, which is E_7 in Figure 5, and the global embedding vector E_{S_t} defined by the aggregation of all the previous items vectors. This hybrid embedding approach performs a linear transformation over the concatenation of both the local and global embedding vectors, followed by a softmax layer to predict the next item probabilities.

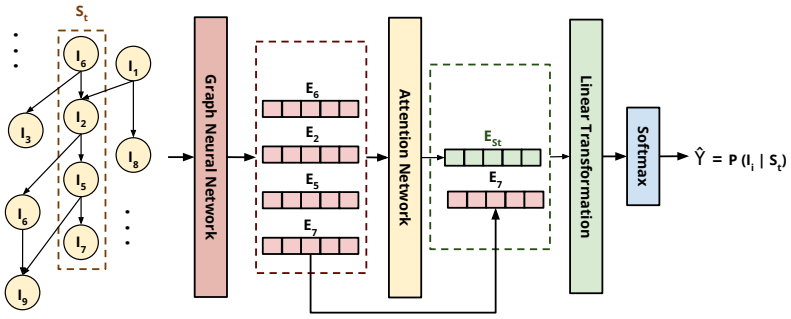

Figure 5. SRGNN model architecture [8].

3.2.7. Collaborative Session-Based Recommendation Machine

A hybrid framework applying collaborative neighborhood information to session-based recommendation was proposed by Wang et al. [36] who hypothesized that neighborhood sessions to the current session can contain useful information in improving the recommendation system predictions—even those made by different users.

The architecture implementation, shown in Figure 6, includes two main encoders. First, the inner memory encoder models the user behavior during their current session using an RNN with an attention mechanism fed with the hidden state of the network from the previous layer h_{t-1}, and current session S_t items. This encoder outputs two concatenated vector embeddings C^{Inner} of the current session behavior representing the whole session items, and the key items clicked during the session. Second, the outer memory encoder looks for the neighborhood sessions that contain patterns similar to the current session out of a subset of the recently stored sessions $(S_{t-1}, S_{t-2}, \ldots)$, which are used to enhance the recommendation process. The final output from the outer memory encoder C^{Outer} represents the influence of other sessions' representations in the neighborhood memory network M in the current session. The final current session representation C_t is formed by a selective fusion between both encoders' output. Finally, the output scores for all items are predicted using a bi-linear decoding scheme between the embedding of item I_i, and the

final representation vector of the current session C_t, followed by a softmax layer. The other two main advantages in CSRM are:

1. Storing recent sessions and looking for neighborhoods within these sessions can be beneficial, especially in e-commerce where temporal drifts in user interests occur frequently.
2. Ease of including different item features in the item embedding vector, which can enhance the recommendations' accuracy.

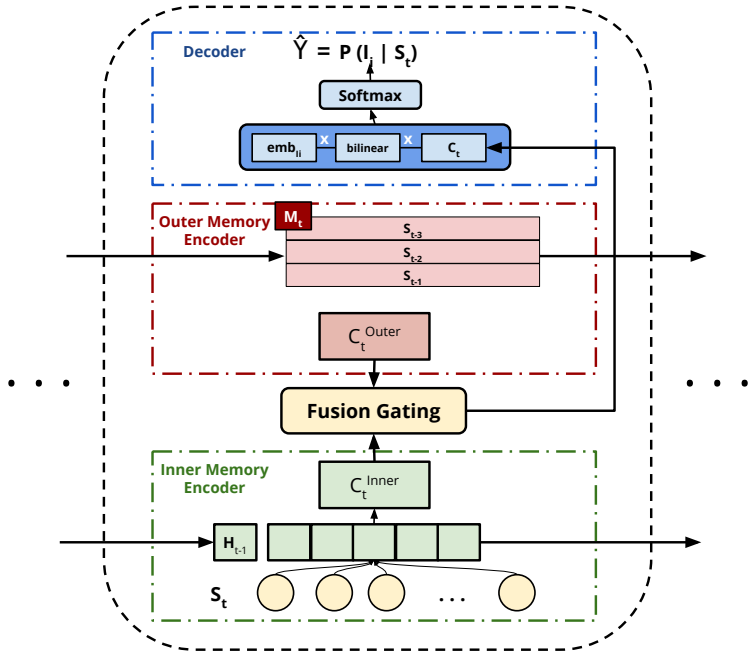

Figure 6. One unit of the CSRM architecture [36].

4. Methodology

4.1. Datasets

All experiments were based on benchmark datasets in the e-commerce domain:

4.1.1. YOOCHOOSE

The first dataset was collected by YOOCHOOSE (https://www.yoochoose.com/ accessed on 1 August 2022) incorporation and published in RecSys Challenge 2015 (http://2015.recsyschallenge.com/ accessed on 1 August 2022). The dataset contains a collection of sessions from a retailer, where each session includes the click events that the user performed in the session. The data were collected during ≈ 6 months in 2014, reflecting the clicks and purchases performed by the users of an online retailer in Europe. The main characteristics that distinguish this dataset from others are having the largest number of clicks and the smallest number of items, which leads to a high presence of most of the items in the dataset. Following the previous literature, we used the last day sessions as a testing set and the rest sessions as a training set. This dataset is referred to as RECSYS [14,50].

4.1.2. Diginetica

Diginetica dataset was used in CIKM Cup 2016 (https://cikm2016.cs.iupui.edu/cikm-cup/ accessed on 1 August 2022) for the personalized e-commerce search challenge. The dataset was provided by DIGINETICA (http://diginetica.com/ accessed on 1 August 2022) corporation containing anonymized search and browsing logs, product data, and anonymized transactions collected for five months from e-commerce websites. We used the transaction data only in our experiments. Similar to the RECSYS dataset, we used the

last-day sessions as a testing set and the remaining sessions as the training set. We use the name CIKMCUP to refer to this dataset in the rest of this paper.

4.1.3. TMall

TMall is a large dataset that consists of interaction logs from the e-commerce TMall website (https://www.tmall.com/ accessed on 1 August 2022). The dataset was collected in six months, including the user–item views logs; however, the time recorded for each event was at the granularity of days. Thus, we used transactions made by the same user in one day as one session, which leads to much longer sessions than the other datasets. Due to the constraints in the computational resources, we used only the dataset in the range from the beginning of September to the end of October (two months) as the training set, and the subsequent day (first of November) as the testing set. We refer to this dataset as TMALL.

4.1.4. Retail Rocket

Finally, the retail-rocket dataset was collected and published by retail-rocket e-commerce personalization company (https://retailrocket.net/ accessed on 1 August 2022) aiming to motivate studies in the field of recommendation systems. The dataset includes user behavioral data from a real-world e-commerce website throughout $\approx$ 4.5 months such as views, add-to-carts, and transactions in addition to items identifiers and their properties in a hashed format. Only the views and add-to-cart events were considered in our experiments, while transaction events were discarded. This dataset and the CIKMCUP dataset were characterized by the small number of clicks compared to the number of existing unique items. Additionally, they also have fewer sessions than both the TMALL and RECSYS datasets. We used the last two days as the testing set and the rest of the sessions as the training set. We refer to this dataset as ROCKET.

During the preprocessing of all datasets, we filtered out sessions of length one as they do not include enough items for evaluation. Additionally, we filtered the clicked items in the test sets, which do not exist in the corresponding training sets in all the experiments. Multiple consecutive clicks on the same item in one session are replaced by a single click on that item. This step was performed as it does not make sense to recommend the same item currently viewed by the user, and it is always preferable to recommend new related items. For example, a session of a click sequence of (1, 1, 1, 2, 2, 3, 4, 4, 1) is preprocessed to (1, 2, 3, 4, 1). Ignoring this step, as in previous studies [18,20], falls in favor of baseline methods such as nearest neighbors and frequent pattern mining over neural-based methods. We kept all the items in the training set and did not remove low-frequency items. During the evaluation, we computed the accuracy of recommendations on all the possible splits starting from the first click of every single session. For instance, in a session represented by the vector (1, 2, 3, 4), we evaluated the recommendations on the session of a single click on (1) with target item 2, the (1, 2) session with target item 3, and the (1, 2, 3) session with target item 4. Finally, the average performance measurements are reported. The statistics of the datasets after the preprocessing are summarized in Table 3

Table 3. Final statistics of the datasets used in the evaluation experiments.

	RECSYS	CIKMCUP	TMALL	ROCKET
Number of items	37.48 K	122.53 K	618.77 K	134.71 K
Number of sessions	7.98 M	310.33 K	1.58 M	367.59 K
Number of clicks	27.68 M	1.16M	10.83 M	1.06 M
Timespan in days	175	152	62	138
Average item frequency	738.5	9.5	17.5	7.9
Average session length	3.47	3.75	6.86	2.88

4.2. Experiments Description

Our study included eight different sets of experiments repeated for each model on all the evaluated datasets. The aim of these experiments was to answer the following research questions (RQs):

- RQ1: Different training session lengths:
  We aimed to evaluate which models can learn from short sessions in length during the training process, and which ones can make better use of lengthy sessions to accurately specify the user's interest. To answer this RQ, we divided each training dataset into three different splits according to their length. We only kept sessions of length <5 in the first split, ≥ 5 and <10 in the second split, and ≥ 10 for the third one. We chose these thresholds as it sounds challenging to determine a correct session context with <5 clicked items. Sessions of length >5 and <10 have an adequate number of items to determine the user's preferences. Sessions with >10 items also have more than enough items to model the user's preferences, but it is also more likely to have a drift in the session context that may or may not be captured by the model. This selection was also made following Liu et al. [16], who chose a threshold of 5 to distinguish between short and long sessions as all datasets have an average session length that is close to 5, as shown in Table 3. All the models were trained on each split of the training sessions. However, the evaluation was performed on the testing splits extracted from the original datasets without further pruning.
- RQ2: Different testing session lengths:
  In this experiment, we measured the models' performances with different session lengths during inference. The main target of this experiment was to observe the model performance during the start of the session and after having an adequate number of interactions from the user. This experiment could help determine which models cannot perform well at long sessions, which usually include user drifts in preferences. On the contrary to the previous RQ, we fixed the same training dataset and divided the test sets to three different splits of sessions of maximum length of 5, 10, >10, respectively. All models are trained on the same training set and evaluated on each test split.
- RQ3: Prediction of items with different popularity in the training set:
  In this set of experiments, we investigated how the models' performance changes concerning the items' frequency in the training set. Answering this RQ can help determine which models can learn well from less frequent items in the training set and accurately predict them during evaluation. Additionally, this experiment can show how models are biased towards predicting the more popular items. In this experiment, we divided the test sets of each dataset to only keep items whose frequencies do not exceed a specific threshold in the training set. The frequency thresholds used for different splits were (50, 100, 200, 300, >300) for the RECSYS and TMALL datasets, and (10, 30, 60, 100, >100) for the CIKMCUP and ROCKET datasets. This categorization was chosen based on the distribution of the items' frequency in the training sets such that each category of a range of frequencies has an adequate number of items (>1000 items) covering the whole range of frequencies, as shown in Figure 7. All models were trained on the same training set, and the evaluation metrics were computed on each set of items in the testing set, satisfying the above frequency threshold conditions.

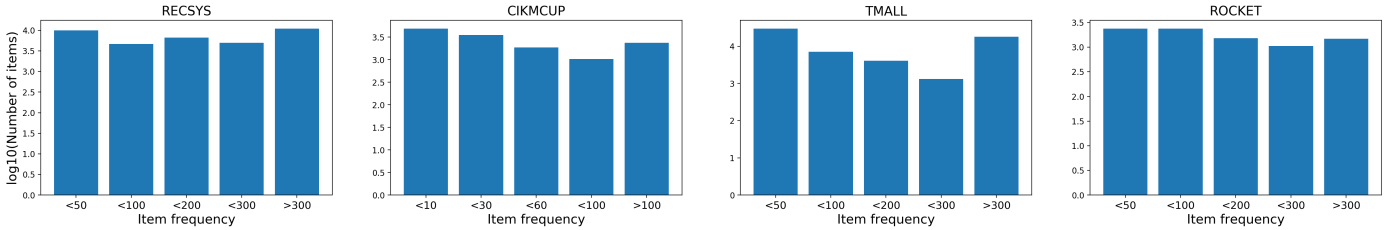

Figure 7. Frequency distribution of the items in the training sets.

- RQ4: Effect of data recency:
  In this experiment, we divided our training sets into three data portions. Each portion was collected during a period that is equal in length to the others but different in terms of the creation date (recency). The first portion represents the most recent collected data sessions. The second one represents the eldest sessions, and the last one is a mixture between the more recent half of the first portion and the older half of the second portion. All models were evaluated on the same test set. We aimed in this experiment to show whether it is crucial to have a dynamic time series modeling to be taken into account while fitting the different models or seasonal changes cannot make a significant drift in the user preferences in the e-commerce domain. For example, in fashion e-commerce, users tend to look for light clothes during summer, which makes it unlikely that they will learn from data collected during winter, when users usually have more preference for heavy clothes. Additionally, it is quite essential to determine how data recency could lead to data leakage problems that could result in a deceptive model accuracy during training.
- RQ5: Effect of training data size:
  We aimed by this experiment to observe the models' performance on different dataset sizes. Answering this RQ can help understand the suitable dataset size corresponding to the number of available items in a dataset such that the model performance is not profoundly affected. Consequently, we save more computational resources without impairment in the models' performance. We randomly selected different splits of the original training datasets such that the sizes of these splits are equal to $\frac{1}{P}$ of the original training set size where $P \in 2, 8, 16, 64, 256$.
- RQ6: Effect of Training Data Time Span:
  Additionally, to check whether the results obtained from RQ4 and RQ5 can be generalized, we ran a similar experiment to RQ5; however, instead of selecting random portions of training sets, we divided them according to the time taken during the data collection. For example, we only used the most recent sessions collected during the last m days before the period used as a testing set to train the model. In this experiment, we used $m \in 2, 7, 14, 30$, and we aimed to know the time span required to train different models and achieve the best performance according to the different data set properties such as the number of items, and average session length.
- RQ7: Items Popularity and Coverage:
  What are the coverage and popularity of the items of each model on the fixed dataset splits? Given the models' predictions, we computed the coverage and popularity of these predictions out of the total number of unique items in the dataset. These measurements can provide a good indication of the models' tendency to predict the most frequent items only, or they can cover the space of items to a large extent. The coverage of the model predictions is a measure of what is called aggregate diversity and how the model is adapted to different sessions' contexts [52]. A small coverage value shows that the model is always recommending a small set of items for all users, as the most popular or frequent items in the training set. A high coverage shows that it recommends a wide range of items with different sessions context [20]. The coverage can also be shown and confirmed by the popularity metric that computes the average frequency of occurrence of the predicted items from the training set normalized by

the frequency of the most popular item. We used the full original training and testing sets of CIKMCUP and ROCKET datasets. Additionally, we used a random split of $\frac{1}{16}$ of the RECSYS and TMALL datasets for training with the last 2-day sessions for testing. The coverage and popularity of the top five predicted items per session are reported in each of these experiments. Models with high accuracy in terms of HR and MRR and the high coverage of items are usually preferred over accurate models with lower coverage. This reflects how the model provides different predictions that are adaptable to the context of the sessions.

- RQ8: Computational Resources:
  What are the required time and memory resources for each model during both training and inference phases? In this experiment, we reported the different computational resources required by each model to observe the trade-off between the model performance and its complexity and if it is worth having more computationally expensive models than simpler ones. Additionally, we aimed to find the suitability of different models to be practically used in making real-time predictions.

The properties of all training and testing splits used in each experiment are summarized in Table A1. We used an early stopping approach during training with a validation split of 10% of the training split for the deep learning models. Additionally, as hyper-parameter optimization is an essential part of determining the performance of models, we ran a random search of 20 iterations for all models on each dataset to tune the most effective hyper-parameters suggested to be tuned by their authors or based on our own experiments. However, we kept the rest of the networks' hyper-parameters as their default values, as mentioned in their corresponding papers. We selected the hyper-parameter settings achieving the highest *HR*@20 for each dataset. The list of tuned hyper-parameters for each model, along with their ranges, can be found in Table A9.

Our work was carried out in part in the high-performance computing center of the University of Tartu (https://hpc.ut.ee/ accessed on 1 August 2022) In case that graphical processing unit (GPU) is used for neural network models, we used NVIDIA Tesla P100 GPU. The memory size was limited to 20GB RAM, Intel(R) Xeon(R) CPU E5-2660 v2 @ 2.20GHz processors with up to 30 cores were allocated from the computing center to run the models that do not support GPU. During reporting training and testing time and memory consumption in our results, we did not use any GPUs to make the comparison fair among all models. However, all neural-based models support the use of GPUs, which is a significant advantage over other algorithms. The source codes used in this study, and the logs of the results, were made publicly available (https://github.com/mmaher22/iCV-SBR accessed on 1 August 2022).

4.3. Evaluation Metrics of Models Performance

We measured the performance of all models in our experiments using several evaluation metrics:

1. Hit Rate (HR@K) is the rate of matching the correct item clicked by the user with any of the list of predictions made by the model. The metric value is set to 1 if the target item is present among top K predictions and 0 otherwise. The formula of *HR@K* for a dataset *D* is described as follows:

$$HR@K = \frac{1}{|\mathcal{S}_D|} \sum_{i=1}^{|\mathcal{S}_D|} 1_{IN}\left(I_{target,K}\left(\hat{\mathbf{Y}}_\mathbf{i}\right)\right). \tag{5}$$

2. Mean reciprocal rank (MRR@K) is the average of reciprocal ranks of the target item if the score of this item is among the top K predictions made by the model. Otherwise, the reciprocal rank is set to zero [14]. The computation of *MRR@K* is given by

$$MRR@K = \frac{1}{|\mathcal{S}_D|} \sum_{i=1}^{|\mathcal{S}_D|} \frac{1}{r_{IN}\left(I_{target,K}\left(\hat{\mathbf{Y}}_\mathbf{i}\right)\right)}. \tag{6}$$

3. Item coverage (COV@K) is the measurement of how the model predicts a variety of items and not only biased to a small subset of frequent items. The item coverage is the ratio of the number of unique items predicted by the model to the total number of items in the training set. Given that K is the number of top predictions to be considered from the model for each session, item coverage is described as

$$\text{COV@K} = \frac{\left|\mathcal{U}\left(_K\left(\hat{\mathbf{Y}}_{i=1,2,\ldots,|\mathcal{S}_{TE}|}\right)\right)\right|}{|\mathcal{U}(\mathcal{S}_{TR})|}. \tag{7}$$

4. Item Popularity (POP@K) is a representation of how the model tends to predict popular items. This metric can reveal models that achieve good performance based on the popularity of certain items in the training set instead of recommending items that match the session context and the user's preferences. The item popularity is the ratio between the average of the predicted items' frequencies to the frequency of the most popular item in the training set. Given that K is the number of top predictions to be considered from the model for each session, item popularity is given by

$$POP@K = \frac{\sum_{i=1}^{|S_{TE}|}\left(\sum_{I\in_K(\hat{\mathbf{Y}}_i)}\mathcal{F}(I)\right)}{|S_{TE}|\cdot max(\mathcal{F}(S_{TR}))}. \tag{8}$$

5. Results

In this section, we report and discuss the results obtained from our extensive evaluation for the different models trying to answer the different research questions proposed in Section 4

5.1. RQ1: Different Training Session Lengths

In the presented diagrams, we report the results for the HR and MRR of each model on all the evaluated datasets. On the contrary to some previous work such as that by Ludewig et al., who used a prediction cut-off threshold of 20 recommendations [18], we chose to set the number of predictions cut-off to five as it is more reasonable to recommend around five items to the user in a real-use case. Furthermore, 20 recommendation is a large number to be used in real-life e-commerce scenarios. However, full results for different predictions cut-off thresholds (1, 3, 5, 10, 20), in this experiment and all the following ones, can be found in our online repository (https://github.com/mmaher22/iCV-SBR/tree/master/Results accessed on 1 August 2022).

As shown in Figure 8, most neural models outperform the non-neural baseline models, except for S-POP, which is the best model in the TMALL dataset due to the nature of the dataset where the clicked items in one session are more frequently repeated within the same session than in other datasets. TMALL has an item frequency per session of 1.204 on average compared with 1.097, 1.115, 1.103 for RECSYS, CIKMCUP, and ROCKET, respectively. This difference means that it is more likely that the same item appears multiple times in a single session in the TMALL dataset. Furthermore, VSKNN has a relatively good performance in the ROCKET dataset. However, the top three models in terms of either HR or MRR in RECSYS and CIKMCUP datasets are always neural-based. Additionally, two neural models are always among the top three performing models in the TMALL and ROCKET datasets. NextItNet has the highest performance in the RECSYS dataset characterized by the highest average item frequency among the used datasets. This property is more suitable for convolutional networks that require a large number of sessions covering all items to model them correctly. However, there is a small decrease in the performance of CSRM, SRGNN, and STAMP when training using a short session length, which is apparent in RECSYS and TMALL that have mostly intermediate to long sessions in the corresponding testing sets. On the other hand, GRU4Rec+ has the highest performance when training using an intermediate and short session length. In contrast, this performance degrades on long sessions since a drift in the user preferences is more likely to occur. Although

NARM also uses a similar network to GRU4Rec+, it has a better performance thanks to the attention layers in its architecture. This improvement in performance was significant compared with a baseline of the GRU4Rec network, which suffers from the vanishing gradient problem, especially in long sessions of 11–17 clicks [39].

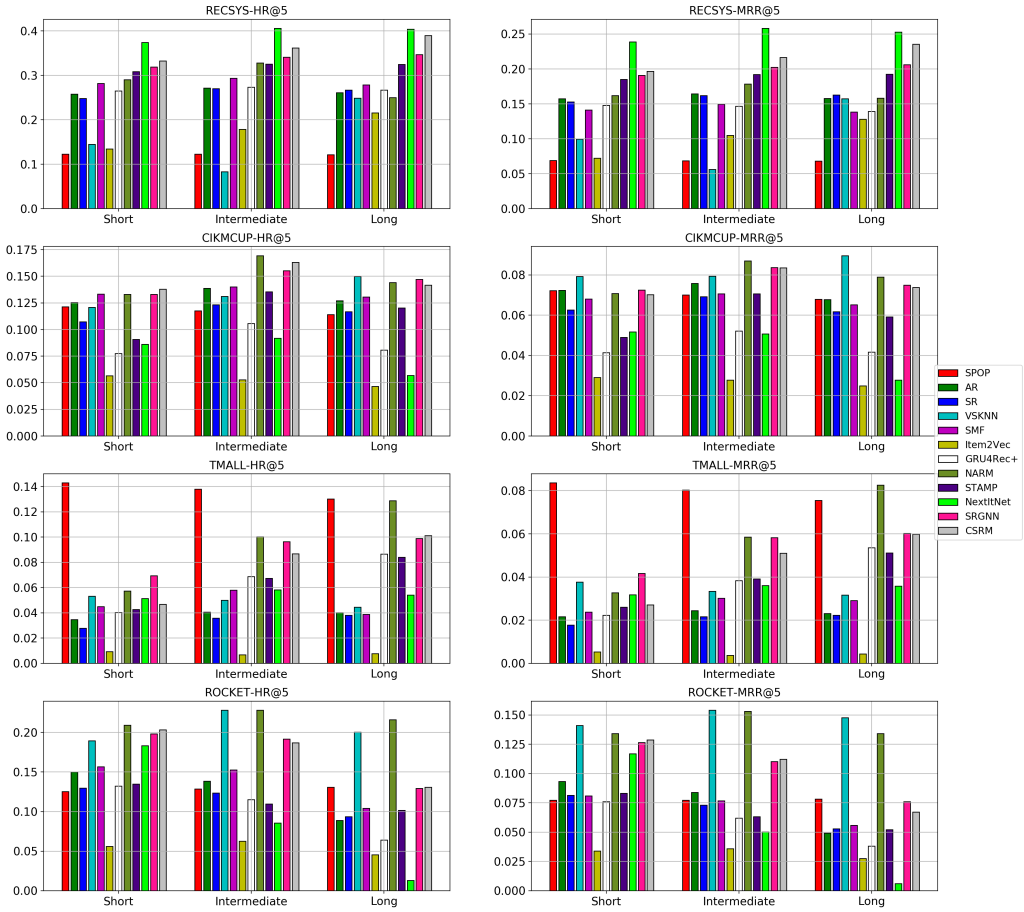

Figure 8. RQ1: Effect of using different training session length on algorithms performance.

Regarding baseline models such as S-POP, AR, and SR, there is a big and consistent change in their performance while changing the training sessions' length, especially in the RECSYS dataset. However, it is clear that these models have comparable performances to neural-based models in datasets where the average item frequency is small and insufficient for learning good item representations such as in CIKMCUP, ROCKET, and TMALL datasets. Overall, baseline methods including S-POP, VSKNN, and SMF are more suitable for short sessions when there are not enough session events to represent the user's preferences. On the other hand, as the session includes a larger number of events, more complex models become better than baselines. This improvement is only apparent for intermediate sessions compared with short sessions. However, long sessions have similar or slightly worse performance since the user preferences are more likely to change in the long sessions. The number of these sessions is not sufficiently large to train the neural models well, especially in the CIKMCUP and ROCKET datasets. However, the total number of events in all training splits of these datasets is close since long sessions have an higher average session length than short ones, as shown in Table A1. In contrast, the TMALL dataset has many more events in the long sessions than intermediate and short ones. This difference in split size could contribute to the noticeable improvement in the performance of the neural models trained with long sessions such as NARM, as also discussed in Section 5.5.

5.2. RQ2: Different Testing Session Lengths

While training using different session lengths gives useful insights about the performance of models, this performance is still highly correlated with the length of the testing sessions. For example, if a model is trained with short sessions while most of the testing set sessions are long, it will not make accurate predictions.

Figure 9 shows the performance of different models using the same training set for all of them while choosing a subset of sessions according to their length as a testing set. S-POP has an increasing performance while the session goes longer since longer sessions have a higher probability of the user clicking on the item again that they liked previously in the same session. However, in lengthy sessions, personalization still pays off since the session has adequate information to precisely model the user preferences [22].

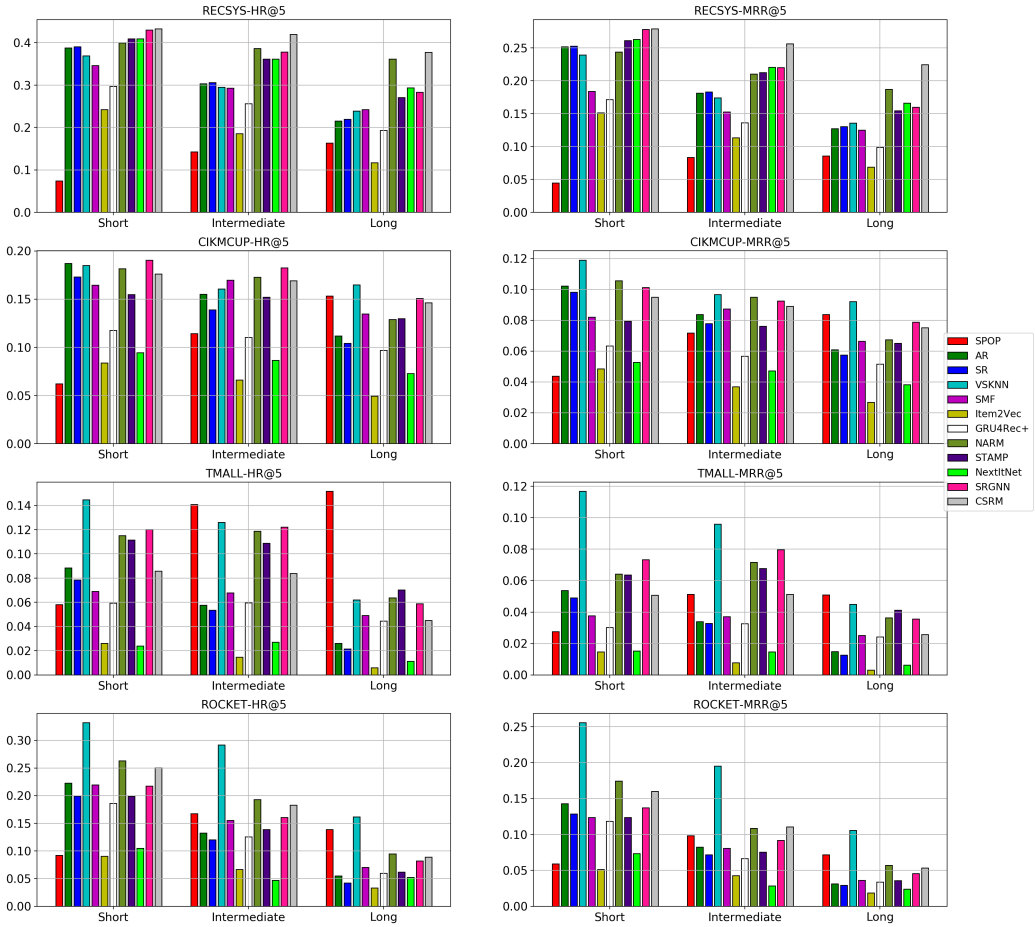

Figure 9. RQ2: Performance of the models on different testing session lengths.

The performance of SR and AR degrade consistently by a large degree, for all the datasets as the session length increases. This impairment is due to the small window of interest that both these algorithms look at while computing the frequent patterns of items, which means that they will not make use of longer sessions. Although the window size for the computation of frequent patterns could be increased, the computational complexity grows exponentially, making it infeasible to use them with long windows sizes. The same effect holds for VSKNN when the selected weighting function for the clicked items in the session gives much higher weights to the recent items such as the quadratic, multiplicative inverse, and log weighting functions. Thus, the weights given to the remaining items outside a specific window size are almost neglected. Additionally, almost all the neural models' accuracy decreases at long sessions, which shows that it is still one of the challenges to model the user drift of interests during the same session. This decrease in performance

is very clear in NextItNet, Item2Vec, and GRU4Rec+ among all datasets, while it is slightly less observable for CSRM, STAMP, and NARM, which could be due to the memory and attention mechanisms applied in these models. NextItNet never outperformed in the RECSYS dataset compared with the results obtained in RQ1. This decrease in performance is due to the smaller training split size used in the experiments of RQ2 compared with those used in RQ1. This claim is also confirmed in the experiments of RQ5 and RQ6, where the deep architecture of convolutional layers in NextItNet requires many instances per item to model it adequately compared with attention networks.

Overall, it seems that achieving excellent performance in long sessions is still a challenging problem for most of the models. On the one hand, pattern mining models do not pay attention to long sessions. They only make use of a small window frame around the item of interest while ignoring the information made earlier by the user in the same session. On the other hand, neural models cannot still accurately detect user-drifts and might suffer from vanishing gradient problems in RNNs, especially for very long sessions [30,39]. Interestingly, much research has focused on a solution to the cold-start problem when users do not make enough clicks to capture their preferences. However, it seems that more attention also needs to be paid to improve the recommendation performance for the long sessions.

5.3. RQ3: Prediction of Items with Different Abundance in the Training Set

Figure 10 shows the performances of different models when trying to predict an item below a specific threshold of occurrences in the training set. This experiment shows how model performance is affected by the number of items' occurrences during fitting. In this experiment, we used frequency thresholds of (<50, <100, <200, and <300) for RECSYS and TMALL datasets which have higher average item frequency, and (<10, <30, <60, <100) for CIKMCUP and ROCKET datasets.

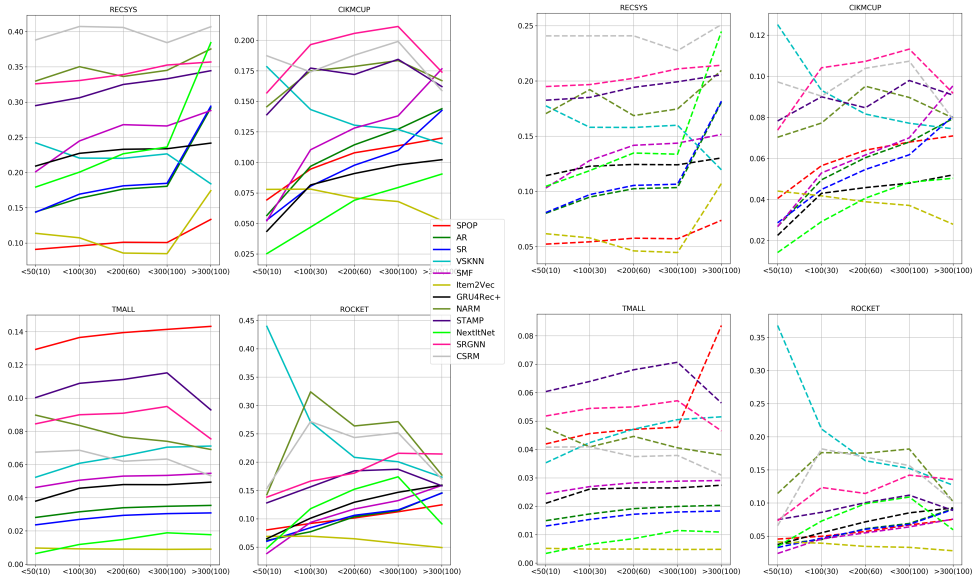

Figure 10. RQ3: Performance of different models using different items' frequency values in training sets (<50, <100, <200, <300, >300) for RECSYS and TMALL and (<10, <30, <60, <100, >100) for CIKMCUP and ROCKET.

The performances of AR, SR, SMF, and GRU4Rec+ are always improved by increasing the frequency threshold among all datasets. To a lesser extent, NextItNet and SRGNN have a slight gain in performance by increasing the frequency threshold where this gain is stopped at very high frequencies as in TMALL and ROCKET datasets. On the other hand, NARM, STAMP, CSRM, Item2Vec, and VSKNN performance measurements do not have a consistent trend while increasing the frequency threshold. In general, some models can have a better performance by increasing the items' occurrences in the training set to be able

to accurately model these items like AR and SR. On the contrary, some other models do not need this high frequency of occurrences, and it is enough to be represented only a few times in the training set as NARM, STAMP, and CSRM—which all have various attention mechanisms.

5.4. RQ4: Effect of Data Recency

Using session-based recommendation models in e-commerce always requires being up-to-date with sufficiently recent data to model the current users' trends. In this experiment, we tested our hypothesis by training the models using the sessions collected from the most recent five days, the five earliest days, and a mix of half of the recent and half of the old splits (in ROCKET), in which we used ten days instead of five as the dataset is smaller than the rest. The test set was fixed for all these different training splits. In Figure 11, it is shown that it is always preferable to train the models using the most recent sessions. It is consistent among all datasets that old sessions have an observable lower performance along almost all the models than recent and mixed splits. Although, there is no large difference between the models' performance on the recent and mixed splits, especially in RECSYS and ROCKET datasets, there is still a small difference in favor of recent splits for CSRM, SRGNN, and NARM in CIKMCUP and TMALL datasets. Surprisingly, VSKNN is the only model with higher performance on old splits in two out of the four datasets and comparable performance in the other two datasets. This behavior could be interpreted as the algorithm only caring about the neighboring sessions of exactly similar items as those clicked by users, which indeed results in better recommendations if matching sessions were found. Additionally, VSKNN has a better overall performance than other models in the ROCKET and CIKMCUP datasets as both of them are characterized by a lower average item frequency than the RECSYS and TMALL datasets, as discussed in RQ3. It is worth mentioning that although a similar trend was observed for the models over each dataset, the differences in each dataset are not equal because some datasets span different time periods. For example, GRU4Rec+ has a higher HR@5 performance in the recent split than the old split by $\sim$ 7% in the RECSYS dataset that spans six months. However, this difference is just $\sim$ 0.5% in the TMALL dataset that spans two months. Hence, the time difference between the old and recent splits in RECSYS is much bigger than that of the TMALL dataset, which could explain the differences in performance among all datasets.

These results confirm that we should account for time-series dynamic modeling in the session-based recommendation to model the trends in users' preferences. Additionally, in case the collected data are much older, there is a high chance that the nearest neighbor algorithms outperform other models.

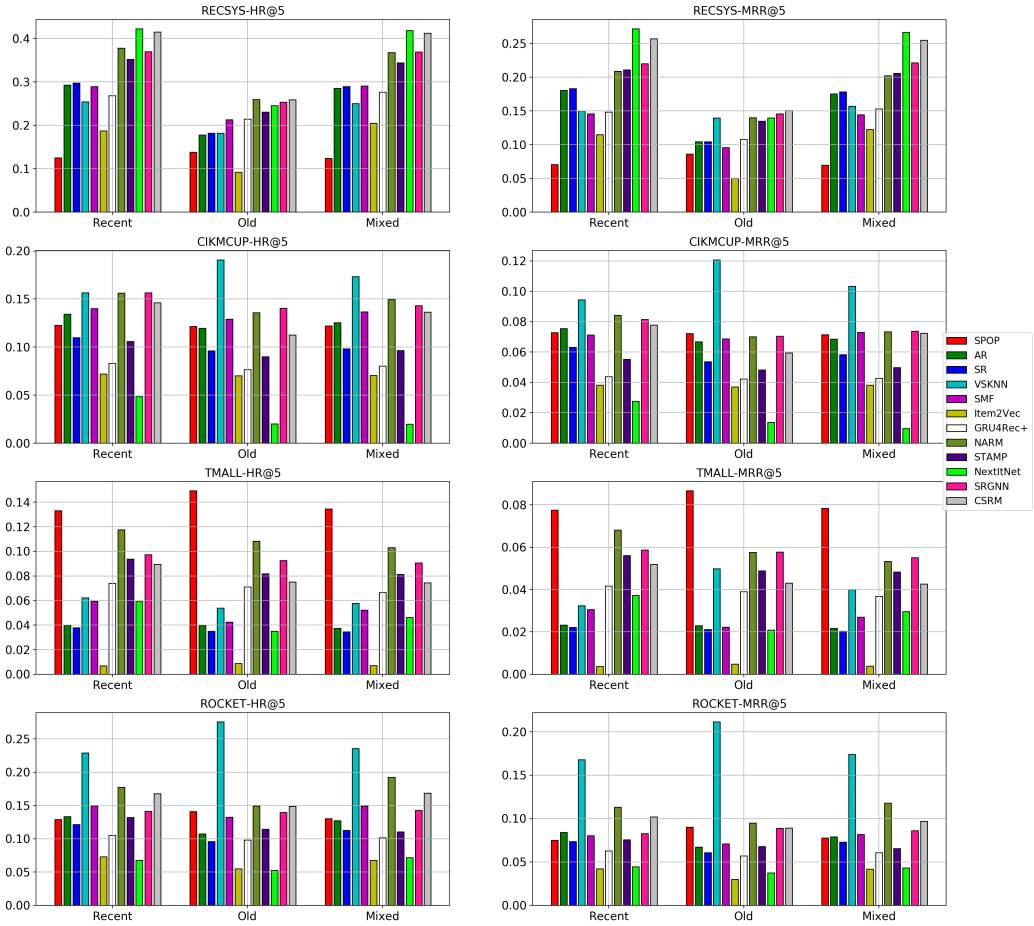

Figure 11. RQ4: Effect of data recency on models performance for different datasets.

5.5. RQ5: Effect of Training Data Size

In this experiment, we aimed to determine the suitable training data sizes corresponding to the different datasets with various characteristics. Additionally, we investigated how the evaluated models performed while using these different training sets sizes. We divided ROCKET and CIKMCUP into splits of $(\frac{1}{2}, \frac{1}{8}, \frac{1}{16}, \frac{1}{64})$ of the original training set size. TMALL and RECSYS were divided into $(\frac{1}{8}, \frac{1}{16}, \frac{1}{64}, \frac{1}{256})$ splits as they are bigger. We refer to these splits as (large, medium, small, very small), respectively.

Figure 12 shows a heat map for the *HR*@5 and *MRR*@5 of different models while increasing the training set size. S-POP and VSKNN are the only algorithms that do not benefit from larger data sizes. It can be easily observed that VSKNN achieved its highest performance along with the four datasets when using the very-small training data portions while S-POP has the same behavior except for the RECSYS dataset. All the neural models' performance is increased when using more training data sessions, which agrees with the nature of deep learning models that are data-hungry. However, SRGNN, NextItNet, and GRU4Rec+ are consistently getting better when increasing training data sizes over all the datasets, while NARM, STAMP, and CSRM are less improved than the former models. Although there is a small improvement in the performance of AR, SR, and SMF in the RECSYS dataset, this improvement is not clear enough in other datasets to generalize the same observation.

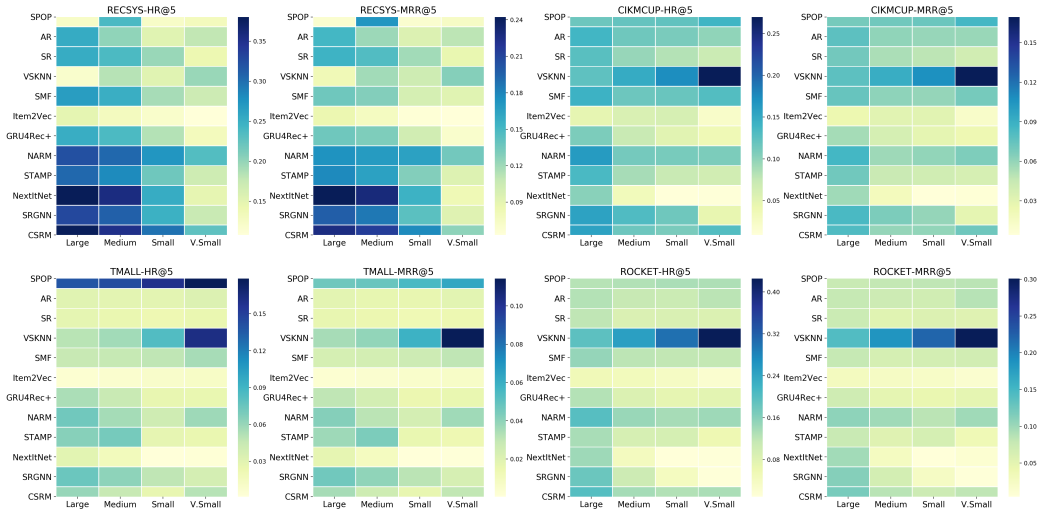

Figure 12. RQ5: Heat map of the models' performance on different training set sizes (darker means better performance). The value of each color code can be mapped to the corresponding numerical value from the vertical bar beside each subplot.

5.6. RQ6: Effect of Training Data Time-Span

Getting some insights out of RQ4 and RQ5 about the importance of training data recency and sizes should reveal sufficient information about the length of the time span required to collect training data sessions. Similarly to RQ5, Figure 13 illustrates a heat-map of the $HR@5$ and $MRR@5$ metrics when training using splits of the most recent x days from the full training set for each dataset, where $x = 2, 7, 14, 30$. Similar to what was previously shown in RQ5, in Figure 12, VSKNN and S-POP still have the best performance when training using a time-span of just two days. Additionally, the performance of neural models becomes better with an increasing dataset time-span. However, in RECSYS dataset, the model improvement is almost ceased as it has a small number of items, and the number of sessions in a 2-day-time-span is quite sufficient to be used to accurately model the context of the sessions.

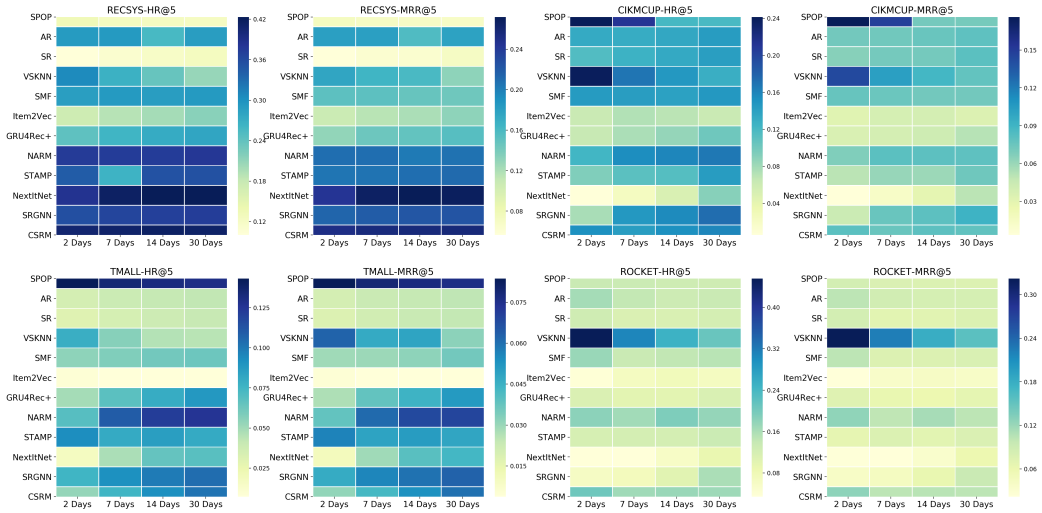

Figure 13. RQ6: Heat map of the models' performance on different training set time spans (darker means better performance).

5.7. RQ7: Items Coverage and Popularity

Item coverage and popularity are good indications of how models tend to cover the space of items in the training set in making recommendations. A model with small coverage and high popularity means that it tends to predict the same items for all users, regardless

of the session's context. Figure 14 shows the natural logarithm of items' coverage (*COV*@5) and popularity (*POP*@5) using the same training and testing splits for each of the datasets. In general, a similar trend is observed among all the datasets comparing the baselines and neural models. For instance, S-POP has the lowest coverage and highest popularity since it only predicts the most frequent items. On the other hand, Item2Vec has the highest coverage and lowest popularity. However, this is not the case in most real-life scenarios. There are usually some popular items that the users usually click on, such as the items with high discounts. As such, Item2Vec still has the lowest performance in terms of HR and MRR since its output vectors are usually dispersed in the vector space, and simple distance measurements are sufficient to capture the similarity among the session context and vectors of the items [53]. Regarding baselines, AR, SR, and VSKNN have quite similar coverage and popularity except for CIKMCUP where VSKNN has higher item coverage. SMF has smaller coverage and popularity.

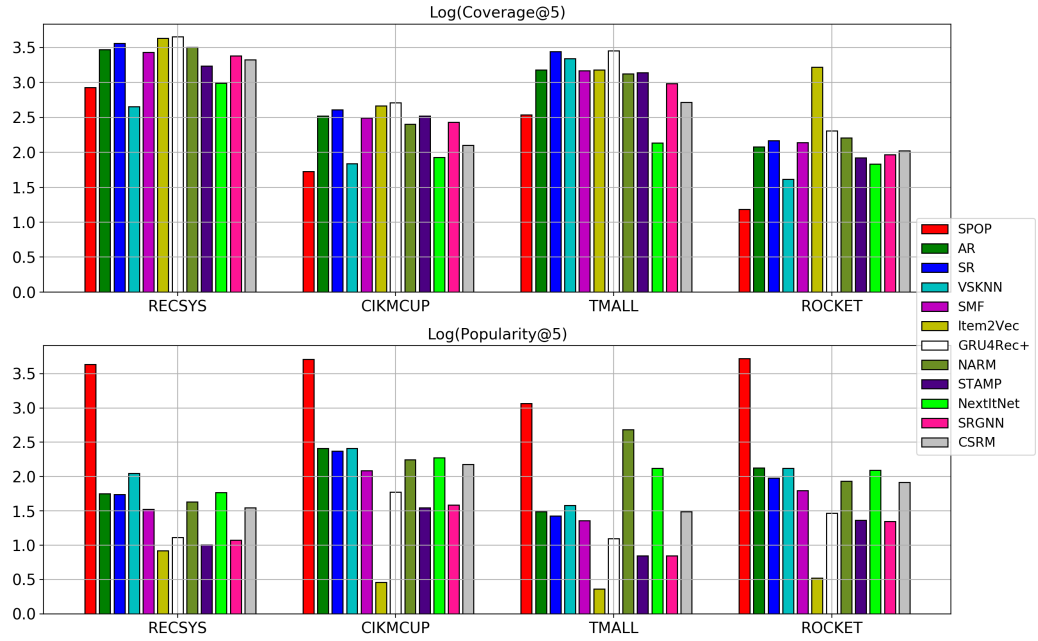

Figure 14. RQ7: Item Coverage and Popularity of Model Predictions.

Regarding the neural models, GRU4Rec+ has slightly higher coverage for its predictions, followed by NARM, STAMP, SRGNN, and CSRM, however, these differences are too small and can barely be observed. CSRM has a memory for storing the most recent sessions, and it predicts items based on the neighborhoods within these sessions. As such, it is always biased towards a subset of recently clicked and popular items than other models. On the other hand, NextItNet has the lowest item coverage with comparable popularity to NARM and CSRM, which suggests that this model is more likely to be over-fitted to a small subset of items, and needs better regularization approaches to be applied to the model. NextItNet is characterized by the presence of the convolutional filters in its architecture, which require much more occurrences per item to generalize well compared with the attention-based networks [54]. SRGNN and STAMP have smaller average popularity across their predictions than other neural models over all datasets. Additionally, NARM has a high item coverage and high popularity with a relatively high accuracy performance according to the HR and MRR metrics. This performance suggests that NARM has the advantage of recommending a wide range of items according to the different sessions' context. However, using SRGNN and STAMP could still be preferable if they have similar accuracy performance to other models as they cover more unpopular items in their recommendations. Detailed results for other predictions' thresholds in this experiment can be found in Table A8 in the Appendix B.

5.8. RQ8: Computational Resources

It is quite important to have a short testing time to make predictions quickly as it is required to provide the user with recommendations in real time after making a specific action. Simultaneously, training computational complexity is important in terms of the scalability of the model and the ability to train it easily every short period of time. Figure 15 summarizes the computational complexity of different models during both training and testing phases in the RECSYS and CIKMCUP datasets. S-POP, AR, SR, and VSKNN are instance-based algorithms where the learning process occurs during inference by iterating over the training set for each test instance. Consequently, the computational resources for training these models are almost neglected. On the other hand, they take a very long during inference, which means that they are not suitable to be employed in making real-time predictions. However, they do not consume much memory as only the dataset and very few parameters are required to be stored. Item2Vec and SMF have quite long training and testing times. Although Item2Vec is considered a neural model, during inference, the similarity distance is computed between the predicted session embedding vector and the items' vectors, which takes a long time. Additionally, SMF is a matrix factorization algorithm that performs heavy matrix multiplication operations during both training and inference. These operations are computationally expensive in terms of both time and memory consumption.

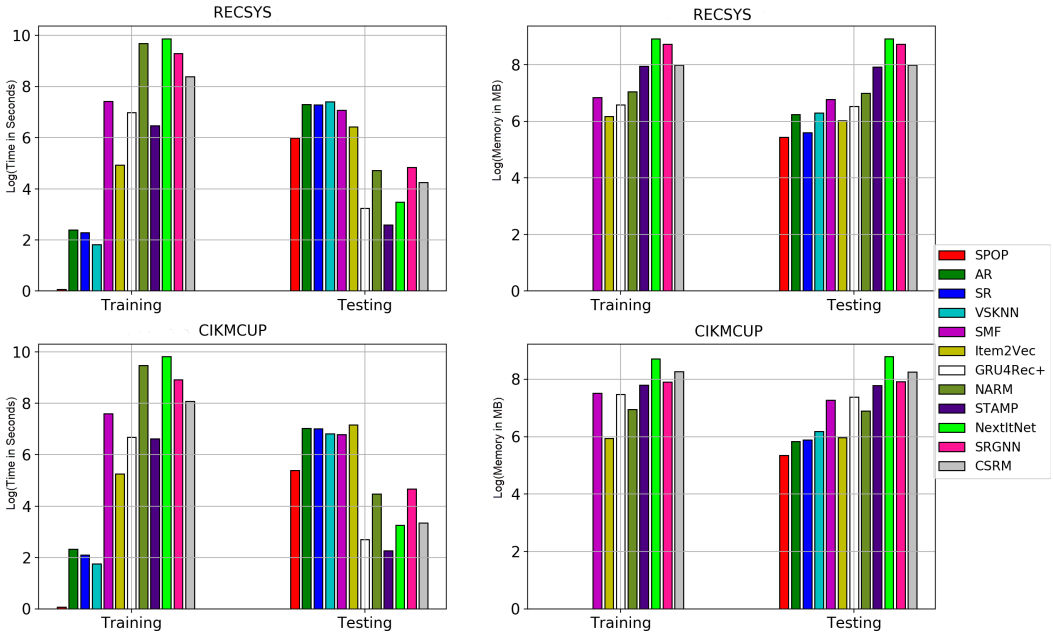

Figure 15. RQ8: Models time and memory consumption on RECSYS and CIKMCUP datasets during training and testing.

Regarding neural models, all of them have a relatively high training time and memory resources; however, they are still characterized by a short time during inference. they only need only a single forward pass to make predictions for one batch of instances. This performance suggests that neural-based models are suitable for real-time predictions. The differences in the training and inference time of the neural models are proportional to the size of each network, the number of layers, and the types of these layers. STAMP has the lowest training and testing time consumption as it is the smallest model in size, followed by GRU4Rec+, CSRM, and then NARM and SRGNN in ascending order. NextItNet is characterized by the presence of multiple convolutional layers and a relatively large model which has high memory consumption due to the mapping of the sessions into images. However, NextItNet has a smaller testing time than NARM, SRGNN, and CSRM due to the weight-sharing properties of the convolutional layers. Additionally, SRGNN, a graph

neural network, has a relatively large memory and time consumption due to the large size of the graph network created by mapping the items and sessions into the corresponding nodes and edges. Overall, all neural models are more compatible with the requirements of real-time predictions. However, they need ample computational resources during training using the back-propagation scheme compared with the simple baseline algorithms.

5.9. Interpretable Meta-Model for Best Model Predictions

Based on our empirical study, we trained a decision tree of a maximum depth of 6 levels and a minimum impurity split of 0.3 to keep it simple and interpretable. This tree model is used to predict the best outperforming model based on dataset characteristics. We used all the experiments that we carried out in our study to construct a new tabular dataset. The features listed in this dataset include the number of sessions, average session length, and average item frequency in both the training and testing sets. We set the target variable as the best performing model out of the whole list of the evaluated models according to the *MRR*@5 evaluation metric. These models are distributed as 14, 36, 7, 4, 10, 5, and 10 instances for the S-POP, VSKNN, NARM, STAMP, NextItNet, SRGNN, and CSRM, respectively. On the other hand, the remaining models did not outperform all the data splits used in our study. Our dataset was divided into ten cross-fold training and hold-out splits. The same decision tree was fitted to each training split to achieve an average accuracy of 87.17% and 87.5% on the training and hold-out splits, respectively. The visualization of one of these fitted trees and the class distribution in the dataset can be found in Figure A1 in the Appendix B.

The most important features used in determining the outperforming model turned out to be in the following order: the average item frequency in the training set, the average session length in the testing set, the number of sessions in the training set, and the number of items in the training set. This simple tree model supports our previous findings of how different dataset characteristics can affect the performance of different models, and choosing the best one. In practice, such interpretable models can help the user shorten the list of models that are more likely to perform well given the characteristics of the given dataset. Additionally, they can help assign weights for different models' predictions if an ensemble of multiple models is used for recommendation using the recommendations corresponding to each model. This experiment shows the potential of finding similar interpretable models that help in developing rules that guide the user to choose the suitable models for a specific dataset.

Our study suggests that using different models according to the different datasets' characteristics could lead to a better performance in the session-based recommendation task. Similar approaches to our decision-tree meta-model can predict which models will perform well with different dataset properties. This information can help combine the predictions out of multiple candidate models, which will consequently improve the final set of recommended items. Our dataset can also be easily extended with more e-commerce datasets that can increase the meta-model accuracy and reliability of predicting the best models.

5.10. Overall Performance

To judge the overall performance of the different models, we used a box-plot in Figure 16 to summarize the ranks of the examined models along with the evaluated datasets. Each chart represents a comparison among the ranks of the models in all the experiments related to one particular dataset. The model with the best performance (highest *HR*@5/*MRR*@5) takes a ranking of one, and the one with the worst performance takes a ranking of twelve. In general, NARM, SRGNN, and CSRM are the top three neural-based models in terms of both *HR*@5 and *MRR*@5 in all datasets. VSKNN has a good performance in the ROCKET dataset with the smallest average session length among all datasets. In contrast with previous studies [18,20], VSKNN has a worse performance than expected. When we investigated the reasons for this performance impairment, we

found out that the preprocessing steps carried out in our study by removing consecutive clicks on the same items, keeping the items of low frequency, and the different evaluation procedures are the main reasons for the differences from these studies. S-POP has the best performance in the TMALL dataset with the largest number of items and average session length. NextItNet has a good performance only in the RECSYS dataset with the smallest number of items and largest average item frequency in the training set, which means that most items are well-represented in the training set by many times. In Table A8, in the Appendix B, the different metrics are evaluated for $1, 5, 20$ predictions cut-off thresholds. Overall, the performance of neural-based models was greatly improved with the new different architectures that emerged in the session-based recommendation. This improvement can also be observed when comparing the performance of neural-based models in older studies [18,19] compared with more recent ones [20]. Hence, neural-based models have a comparable accuracy performance to the most developed nearest neighbor algorithms, and yet more research is needed to further extend these models.

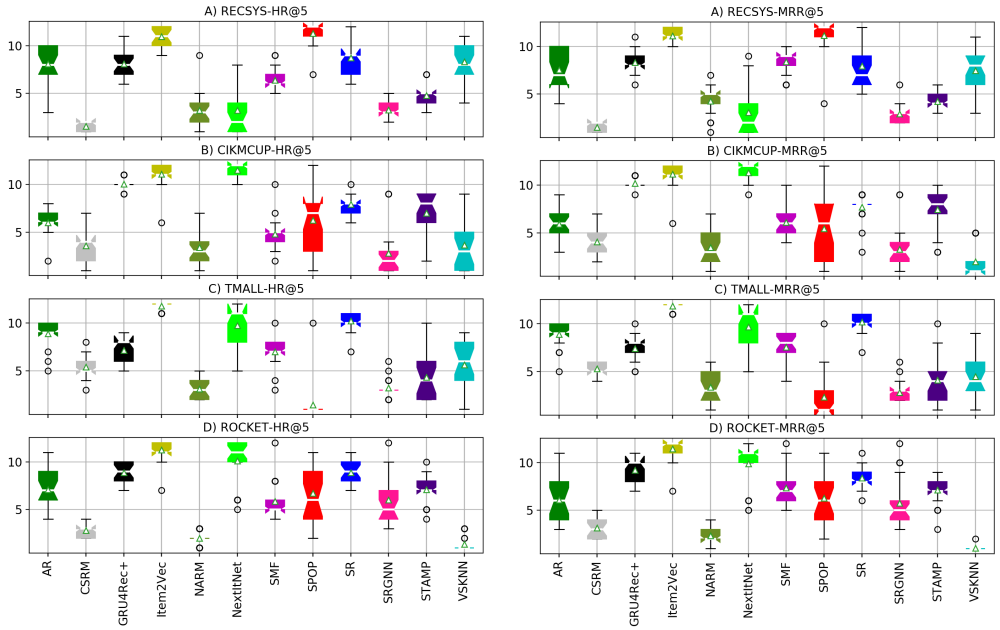

Figure 16. Ranking of different models on each dataset in all the experiments (the lowest rank is the best).

When comparing the results obtained from this study with previous benchmarking studies such as [18–20], it can be observed that the relative overall performance of the evaluated methods on the whole datasets is not always the same. Furthermore, the performance of the models among these studies changes by doing slightly different preprocessing steps. For example, the performance of the STAMP model drops considerably in [17] compared with the reported performance in [16] on the same evaluated datasets (RECSYS and CIKMCUP). For this reason, we find it difficult to draw general conclusions about the relative performance of the evaluated models. Additionally, there are very few publicly available real-world datasets in e-commerce. Hence, this suggests that understanding every single model's performance concerning the different dataset characteristics could provide the user with more insights to help them select the model that suits their dataset better in an interpretable way. Having a large number of real-world datasets covering the whole space of these characteristics sounds practically impossible. Thus, herein, we rely on creating artificially altered data splits out of the original datasets that could better understand the performance of the evaluated models.

6. Conclusions

6.1. Main Insights

In this study, we investigated the current state-of-the-art neural-based models in addition to other baseline algorithms for the session-based recommendation task. Different experiments were carried out trying to answer a set of research questions covering different characteristics of the evaluated datasets, in the e-commerce domain, during both training and testing phases. We used different evaluation metrics covering the accuracy of the models' recommendations, the coverage of predicted items, and their average popularity. Additionally, the consumption of computational resources during training and inference was discussed in terms of the suitability to real-life e-commerce portals.

In general, neural-based models with attention mechanisms such as NARM and CSRM in addition to recurrent models including GRU4Rec+ and the simple VSKNN algorithm are the top-performing models among the majority of datasets with different characteristics. Additionally, the neural-based models are characterized by having reasonable training time budgets and real-time processing during inference. Our results suggest that the training data recency and sizes have an observable effect on the prediction accuracy during inference. In e-commerce, it is clear that dynamic time modeling is a crucial part that needs to be further investigated and included in session-based algorithms to model general trends through different periods. Additionally, dataset characteristics such as average session length, average item frequency, and the total number of sessions do have an impact on the models' performance.

Baseline models including nearest neighbors still outperform all other models even when they have relatively small training sizes or short sessions. Additionally, most of the models' performance degrades slowly on very long sessions, which suggests the need to improve the models' performance in these cases and accurately detect drifts in the user's preferences while making efficient use of older events in the session. In some cases, baseline algorithms outperform neural models; however, due to the computational complexity of these algorithms, especially during the inference time, neural-based models are preferable in making real-time recommendations.

6.2. Challenges and Future Work

Despite the recent leap achieved in improving the performance of neural-based methods in the session-based recommendation, many challenges still need to be tackled with new solutions. As future work, we suggest the following research points that help the community better understand the current models, and tackle these challenges with new solutions:

- E-commerce domain is usually characterized by frequent changes in item properties. For example, the sale campaigns on some items can heavily affect the users' interest. In addition, temporal changes such as weather changes in different seasons, and trends in fashion items can also lead to significant drifts in user preferences. Thus, it is quite important to start looking for models that can deal with possibly different types of items attributes, which are either nominal, numerical, or categorical, to improve the prediction accuracy. Additionally, temporal changes in these attributes should be taken into account while predicting different items. The possible effects of these trends were previously analyzed using an e-commerce use case [55]. However, this has been barely explored in the literature due to the lack of publicly available datasets including sufficient relevant information. Thus, more effort should be made to collect and publish such types of datasets that can help the research community better analyze these trends with different domains and a wide range of scenarios.
- Most current solutions require unique item identifiers to be used during training and prediction phases. However, in many domains, having a fixed set of items is not a feasible solution. For example, new items can be added, and others can run out of stock in the e-commerce domain. Thus, training new models can be a tedious solution, especially for large datasets. Additionally, models can severely suffer from the cold-

start problem for those recently added items. Research work needs to investigate how to use the concept of the dynamic item embedding [56] to be utilized in both the training and inferences phases instead of using a fixed set of unique identifiers.

- The current session-based recommendation systems do not take into account the different user interactions made during the session. On the one hand, different events such as item view, add-to-cart, and add to wishlist show different levels of interest from the user towards the items. On the other hand, other interactions can account for drifts in the user preferences including remove-from-cart and remove-from-wishlist. We believe that modeling such kinds of different interactions in a general way can lead to an improvement in the session-based recommendation.
- Although tuning the models' hyper-parameters can be computationally expensive, it is quite important for further studies to perform extensive experiments on the most promising models to investigate the effect of changing different architecture hyper-parameters. Previous studies have shown that it is always the case that few hyper-parameters have a significant impact on the models' performance [57,58]. As a candidate solution to reduce the search space to be investigated, many studies have introduced solutions to automated machine learning, including the neural network architecture search and hyperparameter optimization, which help carry out these studies more fairly and efficiently [59].
- The extensive evaluation of deep learning approaches in session-based recommendation in domains other than e-commerce, such as music playlist recommendations, has not yet been investigated. As different domains' characteristics can affect the properties of collected data, and the performance of different models, it is quite important to answer similar research questions to those also investigated here other domains. Additionally, other evaluation metrics could also be computed to reveal new interesting information about the behavior of the models and the diversity of their predictions [52]. For example, the Gini index could be evaluated for the models' recommendations to understand whether the predictions are biased towards some items more than others.
- In session-based recommendation, it is typical to train a model using the sessions collected during a specific period and evaluating the model using the sessions collected in the subsequent days to that period. Although this approach was always followed in our study, it is also possible to experiment using different training–test splits in some experiments such as RQ4, RQ5, and RQ6, where a random set of sessions were chosen from the entire list of existing sessions. One of the limitations of this work is that we only used a single training–test split due to our experiments' computational complexity. Hence, as a future work, evaluating the results of some research questions using multiple training–test splits can be used to confirm and generalize our main conclusions.

Author Contributions: Conceptualization, M.M., P.M.N., C.O. and G.A.; metholodgy, M.M., P.M.N., C.O. and G.A.; software, M.M., P.M.N., A.R.; formal analysis, M.M., P.M.N., C.O. and G.A.; writing—original draft preparation, M.M. and P.M.N.; writing—review and editing, M.M., P.M.N., A.R., C.O., J.C., R.S. and G.A., funding acquisition, C.O. and G.A. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partially supported by the Estonian Centre of Excellence in IT (EXCITE) funded by the European Regional Development Fund and the Rakuten, Inc. (Grant VLTTI19503).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors also gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X Pascal GPU.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Properties of the Datasets in All the Experiments

Table A1. Final statistics of dataset splits used in all the different evaluation experiments.

Target RQ		RECSYS						CIKMCUP					
		Training Set			Test Set			Training Set			Test Set		
		No. of Sessions	Avg. Session Length	Avg. Item Freq.	No. of Sessions	Avg. Session Length	Avg. Item Freq.	No. of Sessions	Avg. Session Length	Avg. Item Freq.	No. of Sessions	Avg. Session Length	Avg. Item Freq.
RQ1	Long Sessions	447K	13.82	187.06	13K	4.58	1000	26K	13.14	4.33	2K	4.80	20.95
	Intermediate Sessions	1402K	5.59	229.87	13K	4.58	1336	66K	6.05	5.04	2K	4.76	28.28
	Short Sessions	4888K	2.53	336.73	13K	4.58	2184	116K	2.71	4.38	2K	4.66	33.07
RQ2	Short Sessions	145K	4.15	31.67	9K	2.57	1059	62K	5.04	4.67	1.2K	2.64	31.21
	Intermediate Sessions	145K	4.15	31.67	3K	5.77	889.1	62K	5.04	4.67	716	5.53	32.50
	Long Sessions	145K	4.15	31.67	1.4K	14.38	669.7	62K	5.04	4.67	299	11.25	30.00
RQ3	Very Low Freq.	145K	4.15	31.67	13K	4.05	19.89	62K	5.04	4.67	2K	3.19	4.09
	Low Freq.	145K	4.15	31.67	13K	3.80	36.96	62K	5.04	4.67	2K	2.10	9.54
	Intermediate Freq.	145K	4.15	31.67	13K	3.44	71.10	62K	5.04	4.67	2K	1.56	15.15
	High Freq.	145K	4.15	31.67	13K	3.17	104.86	62K	5.04	4.67	2K	1.27	20.31
RQ4	Mixed	743K	3.97	94.88	13K	4.58	1453	18K	4.97	2.49	1.9K	4.47	14.80
	Recent	470K	4.02	79.19	13K	4.58	1602	23K	5.03	2.99	2K	4.52	11.18
	Old	662K	3.91	100.17	4.5K	3.45	519.7	22K	5.00	3.01	1.4K	3.81	10.39
RQ5	Large Portion	842K	3.91	99.60	13K	4.57	564.3	104K	5.06	5.64	2K	4.85	39.23
	Medium Portion	421K	3.91	55.49	13K	4.57	282.6	26K	5.08	2.75	2K	4.47	11.89
	Small Portion	105K	3.93	18.81	13K	4.54	72.63	13K	5.08	2.06	1.8K	4.17	7.05
	Very Small Portion	26K	3.89	7.35	12K	4.16	21.78	3K	5.13	1.37	1.2K	3.53	3.12
RQ6	2 days	145K	4.15	31.67	13K	4.57	852.8	5K	5.01	1.74	1.6K	4.04	5.18
	7 days	354K	4.1	102.91	13K	4.58	1437	16K	5.01	2.59	2K	4.44	11.40
	14 days	635K	4.04	63.21	13K	4.58	1835	28K	4.98	3.25	2K	4.59	17.21
	30 days	1241K	4.05	174.07	13K	4.58	2472	62K	5.04	4.67	2K	4.77	31.27
RQ7		421K	3.91	55.49	13K	4.57	282.6	207K	5.07	8.57	2K	4.93	75.20
RQ8		421K	3.91	55.49	13K	4.57	282.6	207K	5.07	8.57	2K	4.93	75.20
		TMALL						ROCKET					
		Training Set			Test Set			Training Set			Test Set		
		No. of Sessions	Avg. Session Length	Avg. Item Freq.	No. of Sessions	Avg. Session Length	Avg. Item Freq.	No. of Sessions	Avg. Session Length	Avg. Item Freq.	No. of Sessions	Avg. Session Length	Avg. Item Freq.
RQ1	Long Sessions	488K	12.79	11.82	9K	9.72	143.9	13K	16.73	4.2	1.3K	3.93	17.49
	Intermediate Sessions	462K	5.35	7.26	9K	9.03	80.10	41K	5.53	4.32	1.4K	3.72	17.76
	Short Sessions	627K	2.6	6.05	9K	8.56	70.97	205K	2.43	4.33	1.8K	3.60	36.42
RQ2	Short Sessions	71K	7.86	3.81	3K	2.59	42.05	51K	3.47	3.17	1.2K	2.38	26.07
	Intermediate Sessions	71K	7.86	3.81	3K	5.21	36.92	51K	3.47	3.17	260	5.14	20.00
	Long Sessions	71K	7.86	3.81	3K	17.43	25.92	51K	3.47	3.17	96	16.5	18.72
RQ3	Very Low Freq.	71K	7.86	3.81	9K	2.12	11.06	259K	3.65	7.03	1.8K	3.03	4.78
	Low Freq.	71K	7.86	3.81	9K	1.52	15.99	259K	3.65	7.03	1.8K	2.23	12.12
	Intermediate Freq.	71K	7.86	3.81	9K	1.18	21.67	259K	3.65	7.03	1.8K	1.73	19.85
	High Freq.	71K	7.86	3.81	9K	1.08	24.76	259K	3.65	7.03	1.8K	1.44	27.36
RQ4	Mixed	265K	7.44	6.1	9K	9.39	63.09	37K	3.6	2.71	1.4K	3.66	14.31
	Recent	327K	7.25	7.62	9K	9.52	91.46	34K	3.47	2.72	1.4K	3.7	17.07
	Old	231K	7.14	6.52	7K	6.53	50.95	42K	3.72	3.18	1K	3.36	11.28
RQ5	Large Portion	186K	7.03	4.92	9K	8.76	40.27	130K	3.66	4.77	1.7K	3.68	33.80
	Medium Portion	91K	7.16	3.58	8.5K	8.06	22.93	32K	3.68	2.46	1.3K	3.59	10.33
	Small Portion	23K	7.26	2.17	7K	6.5	8.22	16K	3.67	1.89	1K	3.5	6.26
	Very Small Portion	5.5K	7.22	1.56	5K	4.82	3.54	4K	3.61	1.31	512	3.11	2.95
RQ6	2 days	71K	7.86	3.81	9K	8.61	28.94	3.8K	3.41	1.46	674	3.23	5.51
	7 days	228K	7.46	6.42	9K	9.37	69.88	13K	3.49	1.97	1.1K	3.61	9.70
	14 days	462K	6.95	8.98	9K	9.62	123.2	24K	3.47	2.41	1.3K	3.68	13.68
	30 days	883K	6.42	12.02	9K	9.76	184.4	51K	3.47	3.17	1.6K	3.72	21.92
RQ7		91K	7.16	3.58	8.5K	8.06	22.93	259K	3.65	7.03	1.8K	3.68	64.43
RQ8		91K	7.16	3.58	8.5K	8.06	22.93	259K	3.65	7.03	1.8K	3.68	64.43

Appendix B. Experiments Results

Table A2. RQ1: Effect of using different training session lengths on algorithms performance.

	RECSYS						CIKMCUP					
	HR@5			MRR@5			HR@5			MRR@5		
	Short	Intermediate	Long	Short	Intermediate	Long	Short	Intermediate	Long	Short	Intermediate	Long
SPOP	0.12266	0.12276	0.12179	0.06912	0.0686	0.068	0.12132	0.11757	0.11394	0.07221	0.07001	0.06793
AR	0.2579	0.27188	0.26112	0.15759	0.16427	0.15784	0.12547	0.1388	0.127	0.07238	0.07587	0.06782
SR	0.24833	0.27026	0.26726	0.15295	0.16194	0.16272	0.10735	0.12339	0.1169	0.06262	0.06919	0.06181
VSKNN	0.14516	0.0834	0.24868	0.09956	0.05627	0.15752	0.12081	0.13116	0.14978	0.07924	0.07936	0.08961
SMF	0.28226	0.29364	0.27855	0.14128	0.14973	0.13823	0.13326	0.14014	0.1307	0.06803	0.07067	0.06515
Item2Vec	0.13476	0.17862	0.21602	0.07227	0.10501	0.12798	0.05672	0.05291	0.04657	0.02915	0.02772	0.02499
GRU4Rec+	0.26486	0.27336	0.26708	0.14788	0.14683	0.13923	0.07762	0.1057	0.08062	0.04137	0.05222	0.0417
NARM	0.29068	0.32845	0.25034	0.16194	0.17854	0.15833	0.13304	0.16946	0.14416	0.07081	0.08692	0.07892
STAMP	0.30889	0.32552	0.32518	0.185	0.19192	0.19246	0.09075	0.13554	0.12024	0.04903	0.07064	0.05912
NextItNet	0.37462	0.40625	0.40444	0.23878	0.25829	0.25296	0.08629	0.09191	0.05682	0.05168	0.05077	0.02778
SRGNN	0.31917	0.34112	0.34709	0.19072	0.20263	0.20599	0.13318	0.15517	0.14709	0.07248	0.08364	0.07491
CSRM	0.33285	0.36188	0.38993	0.19668	0.21635	0.23569	0.13793	0.16326	0.14169	0.07025	0.08349	0.07384
	TMALL						**ROCKET**					
	HR@5			MRR@5			HR@5			MRR@5		
	Short	Intermediate	Long	Short	Intermediate	Long	Short	Intermediate	Long	Short	Intermediate	Long
SPOP	0.14287	0.13787	0.13027	0.08359	0.08036	0.07551	0.1253	0.12876	0.13085	0.07723	0.07728	0.07821
AR	0.03466	0.04054	0.04016	0.02166	0.02438	0.02309	0.15006	0.13845	0.08871	0.09321	0.08381	0.04915
SR	0.02776	0.03571	0.03799	0.01776	0.02157	0.02231	0.12972	0.12353	0.09365	0.08145	0.07307	0.05296
VSKNN	0.0531	0.04991	0.04448	0.03762	0.03345	0.03167	0.18953	0.2281	0.2005	0.14098	0.15403	0.1476
SMF	0.0449	0.05804	0.03869	0.02375	0.03019	0.02905	0.15691	0.15284	0.10432	0.08085	0.07681	0.05583
Item2Vec	0.00941	0.00677	0.00769	0.00525	0.00375	0.00431	0.05597	0.06276	0.04536	0.03381	0.03583	0.02738
GRU4Rec+	0.04041	0.0687	0.08644	0.0223	0.03839	0.05352	0.13234	0.11502	0.06406	0.07596	0.06185	0.03817
NARM	0.05725	0.10016	0.12878	0.03277	0.05849	0.08264	0.20912	0.22816	0.2162	0.13406	0.15309	0.13416
STAMP	0.04271	0.06745	0.08406	0.02609	0.03915	0.05112	0.13486	0.10982	0.10178	0.08309	0.06312	0.05202
NextItNet	0.05145	0.05813	0.0542	0.03182	0.03605	0.03585	0.18345	0.08576	0.01296	0.11701	0.05013	0.00614
SRGNN	0.06934	0.09626	0.09892	0.04171	0.05824	0.0602	0.19835	0.19193	0.12954	0.12643	0.11033	0.0759
CSRM	0.04689	0.08688	0.10122	0.02716	0.05099	0.05988	0.20338	0.18693	0.13085	0.12872	0.1123	0.06709

Table A3. RQ2: Performance of the models on different testing session lengths.

	RECSYS						CIKMCUP					
	HR@5			MRR@5			HR@5			MRR@5		
	Short	Intermediate	Long	Short	Intermediate	Long	Short	Intermediate	Long	Short	Intermediate	Long
SPOP	0.07384	0.14258	0.1636	0.04449	0.08335	0.08591	0.06207	0.11427	0.15337	0.04385	0.07187	0.08378
AR	0.38767	0.30338	0.21566	0.25205	0.18148	0.12749	0.18727	0.15503	0.11187	0.10216	0.08386	0.06085
SR	0.39096	0.30644	0.21935	0.25272	0.18307	0.13043	0.17307	0.13897	0.10409	0.09832	0.07778	0.05749
VSKNN	0.36927	0.29541	0.2387	0.23924	0.17427	0.13559	0.18488	0.1607	0.1648	0.11907	0.09674	0.09222
SMF	0.34652	0.29268	0.24241	0.18395	0.15282	0.12526	0.16446	0.16972	0.13466	0.08197	0.08744	0.06645
Item2Vec	0.24224	0.18583	0.11741	0.15142	0.1136	0.06868	0.08369	0.06619	0.04921	0.04861	0.03696	0.02681
GRU4Rec+	0.29743	0.25613	0.19353	0.17157	0.13615	0.09878	0.11752	0.11022	0.09677	0.06347	0.05678	0.05154
NARM	0.39978	0.38658	0.36131	0.24387	0.21054	0.18721	0.18182	0.1728	0.12893	0.10568	0.09507	0.06741
STAMP	0.40926	0.36166	0.27058	0.26132	0.21244	0.1547	0.15474	0.15195	0.1297	0.07936	0.07619	0.06506
NextItNet	0.40924	0.36179	0.2936	0.26304	0.22057	0.16627	0.09462	0.08665	0.07292	0.05279	0.04716	0.03819
SRGNN	0.4302	0.37851	0.28337	0.27815	0.22035	0.15973	0.19062	0.18269	0.15094	0.10134	0.09253	0.07886
CSRM	0.43302	0.41941	0.37752	0.27919	0.25634	0.22471	0.17629	0.16913	0.14617	0.09492	0.08908	0.07515
	TMALL						**ROCKET**					
	HR@5			MRR@5			HR@5			MRR@5		
	Short	Intermediate	Long	Short	Intermediate	Long	Short	Intermediate	Long	Short	Intermediate	Long
SPOP	0.05817	0.14093	0.15166	0.02761	0.05123	0.05096	0.09225	0.16759	0.13865	0.05905	0.09838	0.07192
AR	0.08851	0.05766	0.02613	0.05383	0.03403	0.01485	0.22263	0.13241	0.05492	0.14306	0.08228	0.03132
SR	0.07861	0.05352	0.02145	0.04914	0.03276	0.01263	0.19926	0.12037	0.0422	0.12845	0.07194	0.02945
VSKNN	0.14484	0.12595	0.06213	0.11678	0.09593	0.04496	0.33197	0.29157	0.16186	0.25566	0.1952	0.1059
SMF	0.06913	0.06787	0.04912	0.03776	0.03706	0.02514	0.21956	0.15556	0.07033	0.12352	0.08088	0.03639
Item2Vec	0.02605	0.01485	0.00604	0.01472	0.00778	0.00316	0.09039	0.0666	0.03342	0.05138	0.04295	0.01849
GRU4Rec+	0.05928	0.05955	0.04445	0.03018	0.03272	0.02423	0.18638	0.12534	0.05988	0.11851	0.06665	0.03383
NARM	0.11503	0.11868	0.06368	0.06421	0.07165	0.03628	0.26322	0.19321	0.09484	0.1743	0.10842	0.0572
STAMP	0.11151	0.1088	0.07018	0.06363	0.0677	0.04122	0.19888	0.13902	0.06162	0.12361	0.07547	0.03569
NextItNet	0.02386	0.02716	0.01139	0.01531	0.01463	0.00624	0.10473	0.04688	0.05208	0.07328	0.02845	0.02378
SRGNN	0.12027	0.1221	0.05887	0.07332	0.07965	0.03554	0.21755	0.16059	0.08203	0.13708	0.09193	0.0458
CSRM	0.0859	0.08377	0.04497	0.05066	0.05133	0.02565	0.25015	0.18316	0.08871	0.15998	0.11055	0.05343

Table A4. RQ3: Performance of different models using different items' frequency values in training set ($< 50, < 100, < 200, < 300$) for RECSYS and TMALL and ($< 10, < 30, < 60, < 100$) for CIKMCUP and ROCKET.

	RECSYS								CIKMCUP							
	HR@5				MRR@5				HR@5				MRR@5			
	$< 50(< 10)$	$< 100(< 30)$	$< 200(< 60)$	$< 300(< 100)$	< 50	< 100	< 200	< 300	< 50	< 100	< 200	< 300	< 50	< 100	< 200	< 300
SPOP	0.09101	0.09597	0.10111	0.10068	0.05233	0.0544	0.0577	0.05716	0.06932	0.09455	0.1079	0.1137	0.04055	0.0565	0.06404	0.06799
AR	0.1443	0.16365	0.17659	0.18055	0.08025	0.0949	0.10249	0.10354	0.05657	0.09687	0.11444	0.12731	0.02686	0.04963	0.0604	0.06803
SR	0.14384	0.16941	0.18121	0.18478	0.0809	0.09716	0.10548	0.10658	0.05284	0.08061	0.09772	0.10986	0.02869	0.04493	0.05465	0.06181
VSKNN	0.24221	0.22056	0.22025	0.22658	0.17753	0.1581	0.15794	0.15997	0.17844	0.14333	0.13052	0.12706	0.12518	0.09318	0.08164	0.07717
SMF	0.20131	0.24493	0.26802	0.26617	0.10311	0.12805	0.14179	0.14373	0.05227	0.11041	0.12816	0.13817	0.02685	0.05297	0.0617	0.07005
Item2Vec	0.1138	0.10743	0.08578	0.08493	0.06174	0.05795	0.04619	0.04471	0.07792	0.07816	0.07091	0.06801	0.04414	0.04182	0.03894	0.03716
GRU4Rec+	0.20933	0.22735	0.23308	0.23388	0.11431	0.12285	0.12446	0.12414	0.04358	0.08158	0.09095	0.09798	0.02258	0.04305	0.04584	0.048
NARM	0.32998	0.35038	0.33644	0.34514	0.17048	0.19217	0.16868	0.17472	0.14566	0.17503	0.17865	0.18348	0.07034	0.07726	0.09518	0.0897
STAMP	0.29511	0.30625	0.32509	0.33313	0.18261	0.18517	0.19428	0.19928	0.1391	0.1773	0.17204	0.18448	0.07826	0.09003	0.08469	0.09796
NextItNet	0.17952	0.20069	0.22674	0.23628	0.10464	0.11888	0.13481	0.13372	0.02529	0.04702	0.06904	0.07946	0.01424	0.02926	0.04069	0.0484
SRGNN	0.32603	0.33067	0.33917	0.35266	0.19498	0.19677	0.20246	0.21098	0.15698	0.19654	0.20573	0.21139	0.07377	0.10416	0.10727	0.11323
CSRM	0.38835	0.4074	0.40604	0.38443	0.24086	0.24088	0.24091	0.22746	0.18739	0.17416	0.18789	0.19913	0.09726	0.09033	0.10391	0.10734
	TMALL								ROCKET							
	HR@5				MRR@5				HR@5				MRR@5			
	< 50	< 100	< 200	< 300	< 50	< 100	< 200	< 300	< 50	< 100	< 200	< 300	< 50	< 100	< 200	< 300
SPOP	0.12941	0.13657	0.13952	0.14145	0.04196	0.04561	0.0471	0.04786	0.08056	0.09195	0.10168	0.11253	0.04565	0.04991	0.05716	0.06696
AR	0.0282	0.03158	0.03403	0.0349	0.01495	0.01731	0.01918	0.01996	0.06217	0.07735	0.10371	0.11454	0.03765	0.04492	0.06138	0.06922
SR	0.02373	0.027	0.02944	0.03048	0.01306	0.01536	0.01721	0.018	0.06042	0.08445	0.10632	0.1158	0.03292	0.04688	0.06048	0.06783
VSKNN	0.05241	0.0608	0.06517	0.0705	0.03533	0.04243	0.04716	0.05053	0.43962	0.27098	0.20856	0.20077	0.36826	0.21194	0.16397	0.15238
SMF	0.04627	0.0506	0.05308	0.0535	0.02445	0.02694	0.02828	0.02885	0.03886	0.09381	0.11738	0.13227	0.02394	0.04583	0.05507	0.06403
Item2Vec	0.00976	0.00925	0.0091	0.00896	0.00519	0.00495	0.00494	0.0048	0.06908	0.06943	0.06494	0.05686	0.04131	0.03946	0.03455	0.03315
GRU4Rec+	0.03803	0.04578	0.04798	0.04791	0.02108	0.02607	0.02645	0.02649	0.06506	0.10127	0.12935	0.14689	0.03739	0.05511	0.0717	0.08526
NARM	0.08979	0.0836	0.0766	0.07404	0.0476	0.04091	0.04463	0.0406	0.14272	0.32406	0.26391	0.27157	0.11455	0.17624	0.17538	0.18172
STAMP	0.10038	0.10896	0.11129	0.11527	0.06041	0.06397	0.06808	0.07073	0.12821	0.15655	0.18476	0.1875	0.07507	0.08605	0.10065	0.11198
NextItNet	0.00644	0.01192	0.0149	0.01892	0.0034	0.00663	0.00863	0.01148	0.04817	0.11797	0.1523	0.17422	0.03458	0.07285	0.09908	0.10913
SRGNN	0.08458	0.09004	0.09095	0.09503	0.05181	0.05447	0.05501	0.05718	0.1383	0.16667	0.18072	0.21574	0.0734	0.12361	0.11451	0.14222
CSRM	0.06751	0.06864	0.06201	0.06335	0.04083	0.04097	0.0375	0.03792	0.15385	0.2716	0.24348	0.25185	0.06581	0.18189	0.16942	0.15667

Table A5. RQ4: Effect of data recency on model performance for different datasets.

	RECSYS						CIKMCUP					
	HR@5			MRR@5			HR@5			MRR@5		
	Recent	**Old**	**Mixed**	**Recent**	**Old**	**Mixed**	**Recent**	**Old**	**Mixed**	**Recent**	**Old**	**Mixed**
SPOP	0.12481	0.13773	0.12352	0.07032	0.08607	0.06964	0.12274	0.12135	0.12213	0.07284	0.07215	0.07153
AR	0.29245	0.1776	0.28489	0.1806	0.10443	0.17524	0.13427	0.11955	0.12539	0.07544	0.06687	0.0685
SR	0.29712	0.18192	0.28927	0.18315	0.10445	0.1783	0.10969	0.09605	0.09827	0.06309	0.05371	0.05832
VSKNN	0.25384	0.18167	0.24997	0.15025	0.13974	0.15719	0.15665	0.19082	0.17319	0.09448	0.12078	0.1034
SMF	0.28946	0.21291	0.29022	0.14583	0.0957	0.14433	0.14018	0.12912	0.13678	0.0712	0.06884	0.07295
Item2Vec	0.18745	0.09194	0.2049	0.11464	0.05003	0.12256	0.07201	0.07004	0.07049	0.03823	0.03716	0.03827
GRU4Rec+	0.26846	0.21387	0.27605	0.14821	0.10796	0.15316	0.08287	0.07644	0.08018	0.04384	0.04222	0.0426
NARM	0.37792	0.25972	0.36731	0.20859	0.14024	0.20211	0.15625	0.13568	0.14965	0.08421	0.07001	0.07337
STAMP	0.35214	0.23059	0.3436	0.21096	0.135	0.20582	0.10582	0.09011	0.09635	0.05519	0.04826	0.04977
NextItNet	0.4224	0.24509	0.41825	0.27186	0.13956	0.26655	0.04861	0.02009	0.01979	0.02762	0.01363	0.00967
SRGNN	0.36947	0.25332	0.36916	0.22015	0.14564	0.22149	0.15666	0.14037	0.14298	0.08158	0.07055	0.07383
CSRM	0.41451	0.2589	0.41225	0.25714	0.15053	0.25472	0.14608	0.11261	0.13633	0.07785	0.05953	0.07246
	TMALL						ROCKET					
	HR@5			MRR@5			HR@5			MRR@5		
	Recent	**Old**	**Mixed**	**Recent**	**Old**	**Mixed**	**Recent**	**Old**	**Mixed**	**Recent**	**Old**	**Mixed**
SPOP	0.13305	0.14922	0.1345	0.07756	0.08661	0.07834	0.12907	0.14063	0.13015	0.07498	0.09026	0.07752
AR	0.03989	0.03953	0.03737	0.02317	0.0229	0.02173	0.13345	0.10742	0.12696	0.0839	0.06721	0.07907
SR	0.03784	0.03508	0.03458	0.02214	0.02112	0.02033	0.12158	0.09594	0.11262	0.07349	0.06082	0.07304
VSKNN	0.06215	0.05379	0.05775	0.03237	0.04978	0.04004	0.2291	0.27547	0.23572	0.16797	0.21138	0.17384
SMF	0.05925	0.04245	0.05225	0.03063	0.02233	0.027	0.14972	0.13243	0.14927	0.08041	0.0707	0.08176
Item2Vec	0.00675	0.00888	0.00711	0.00372	0.00476	0.00386	0.07314	0.05499	0.06764	0.04217	0.03003	0.04191
GRU4Rec+	0.07389	0.07113	0.06656	0.0417	0.039	0.03681	0.10509	0.09799	0.10153	0.06275	0.05702	0.06085
NARM	0.11755	0.10832	0.10293	0.06809	0.05763	0.0533	0.17745	0.14916	0.19233	0.11323	0.09476	0.11792
STAMP	0.09382	0.08182	0.08124	0.05612	0.04888	0.04835	0.13195	0.11451	0.11052	0.07561	0.06793	0.06546
NextItNet	0.05903	0.03492	0.04622	0.03717	0.02088	0.02962	0.06747	0.05273	0.07173	0.0444	0.0375	0.04319
SRGNN	0.09736	0.09248	0.09067	0.05864	0.05774	0.05508	0.14138	0.13945	0.14271	0.08261	0.0888	0.0861
CSRM	0.08954	0.07502	0.07436	0.05184	0.04301	0.04267	0.16782	0.14886	0.16869	0.10206	0.08908	0.09692

Table A6. RQ5: The models' performances on different training set sizes.

	RECSYS								CIKMCUP							
	HR@5				MRR@5				HR@5				MRR@5			
	Large	**Medium**	**Small**	**V.Small**	**Large**	**Medium**	**Small**	**V.Small**	**Large**	**Medium**	**Small**	**V.Small**	**Large**	**Medium**	**Small**	**V.Small**
SPOP	0.123	0.23887	0.12375	0.13246	0.06885	0.17229	0.06909	0.07456	0.11678	0.12099	0.12219	0.13977	0.06926	0.07126	0.07293	0.08585
AR	0.2539	0.20174	0.15408	0.17881	0.15159	0.12179	0.09584	0.11112	0.13897	0.11553	0.10946	0.09884	0.07758	0.0616	0.05932	0.05875
SR	0.25294	0.23781	0.19775	0.14106	0.15411	0.14695	0.11946	0.08816	0.127	0.0957	0.08297	0.07022	0.07044	0.0536	0.04847	0.03977
VSKNN	0.11868	0.18405	0.15433	0.19799	0.08256	0.11959	0.10386	0.12862	0.12383	0.14896	0.17333	0.26981	0.07731	0.09177	0.10744	0.16946
SMF	0.26748	0.25059	0.1912	0.17083	0.13632	0.12911	0.09989	0.09317	0.14284	0.11634	0.11797	0.12938	0.07414	0.0622	0.05994	0.06912
Item2Vec	0.14973	0.13127	0.12048	0.10802	0.08674	0.07685	0.06474	0.06305	0.04802	0.05903	0.06116	0.02273	0.02534	0.03192	0.03153	0.01275
GRU4Rec+	0.2516	0.23867	0.18468	0.13167	0.13645	0.13098	0.1019	0.0704	0.10874	0.07274	0.05259	0.03795	0.05526	0.03821	0.0296	0.02357
NARM	0.3239	0.30572	0.2733	0.23365	0.17375	0.16939	0.16586	0.13512	0.16323	0.11283	0.11051	0.10799	0.08598	0.05754	0.0609	0.06637
STAMP	0.30545	0.28161	0.21915	0.16974	0.17852	0.16534	0.12933	0.09496	0.13441	0.08896	0.0727	0.06558	0.07124	0.04413	0.04041	0.03772
NextItNet	0.38028	0.35782	0.25321	0.14867	0.24234	0.22923	0.15534	0.08323	0.10202	0.03427	0.00893	0.01042	0.05662	0.01839	0.00417	0.00577
SRGNN	0.32953	0.31111	0.2493	0.17498	0.19709	0.18589	0.14318	0.0939	0.15672	0.13125	0.11545	0.04622	0.08382	0.06738	0.06007	0.02957
CSRM	0.37905	0.3521	0.29914	0.22736	0.22692	0.21471	0.17787	0.12587	0.15689	0.11646	0.10927	0.12969	0.08181	0.06025	0.0604	0.07177
	TMALL								**ROCKET**							
	HR@5				**MRR@5**				**HR@5**				**MRR@5**			
	Large	**Medium**	**Small**	**V.Small**	**Large**	**Medium**	**Small**	**V.Small**	**Large**	**Medium**	**Small**	**V.Small**	**Large**	**Medium**	**Small**	**V.Small**
SPOP	0.13858	0.14347	0.15952	0.17862	0.04681	0.04815	0.05564	0.06277	0.12635	0.12828	0.13363	0.13272	0.07727	0.08062	0.08246	0.08512
AR	0.03168	0.02975	0.02908	0.03367	0.01822	0.01706	0.01649	0.01893	0.12744	0.11242	0.10736	0.12166	0.07956	0.07291	0.06982	0.08674
SR	0.02863	0.02522	0.02274	0.02555	0.01686	0.01524	0.0134	0.01452	0.11887	0.0885	0.08934	0.08664	0.07449	0.05475	0.05791	0.0565
VSKNN	0.05019	0.05684	0.08435	0.16206	0.03527	0.03948	0.05871	0.11423	0.19361	0.24895	0.3021	0.42527	0.14324	0.18544	0.22099	0.30087
SMF	0.0454	0.04575	0.04833	0.05538	0.02371	0.02446	0.02693	0.03025	0.1503	0.1205	0.11787	0.11521	0.07871	0.06489	0.06786	0.06846
Item2Vec	0.00613	0.00772	0.01009	0.01226	0.0032	0.00417	0.00557	0.0067	0.05295	0.04703	0.03815	0.02259	0.0288	0.0247	0.02175	0.01393
GRU4Rec+	0.05466	0.04454	0.02633	0.02606	0.03049	0.02472	0.01523	0.01633	0.12649	0.08693	0.07232	0.07477	0.07074	0.04966	0.04399	0.05
NARM	0.07271	0.05639	0.04121	0.05831	0.04217	0.03128	0.0241	0.03684	0.19803	0.14715	0.13706	0.14397	0.10834	0.0993	0.08523	0.09854
STAMP	0.06511	0.07118	0.02788	0.02535	0.03723	0.04386	0.016	0.01394	0.13456	0.09238	0.09513	0.05937	0.07717	0.0556	0.06497	0.03788
NextItNet	0.02978	0.01732	0.00137	0.00299	0.01769	0.00964	0.00041	0.00116	0.14328	0.04464	0.00947	0.0125	0.09742	0.02718	0.00445	0.01146
SRGNN	0.07542	0.06309	0.04779	0.03849	0.04654	0.03966	0.03141	0.02333	0.18273	0.10603	0.04278	0.01215	0.11294	0.06721	0.02416	0.00645
CSRM	0.0609	0.04509	0.02679	0.051	0.03511	0.02653	0.01672	0.02995	0.19991	0.13938	0.1292	0.13333	0.1202	0.08582	0.07339	0.08299

Table A7. RQ6: The models' performances on different training set time spans.

	RECSYS								CIKMCUP							
	HR@5				MRR@5				HR@5				MRR@5			
	2days	**7d**	**14d**	**30d**	**2d**	**7d**	**14d**	**30d**	**2d**	**7d**	**14d**	**30d**	**2d**	**7d**	**14d**	**30d**
SPOP	0.13353	0.12558	0.12507	0.12356	0.07388	0.07079	0.07026	0.0694	0.24145	0.2112	0.12168	0.11996	0.17616	0.14873	0.07216	0.071
AR	0.29183	0.29309	0.2579	0.28838	0.18077	0.18059	0.15759	0.17865	0.13353	0.13191	0.13511	0.144	0.07294	0.07332	0.07576	0.07949
SR	0.10015	0.11067	0.11818	0.12579	0.05643	0.06239	0.06317	0.07129	0.11615	0.12723	0.13432	0.14264	0.06567	0.07218	0.07174	0.08067
VSKNN	0.30713	0.26796	0.23744	0.20827	0.17618	0.16437	0.16055	0.13191	0.23906	0.16883	0.14673	0.12945	0.14352	0.10287	0.08866	0.07838
SMF	0.28808	0.2933	0.2893	0.29315	0.15163	0.15178	0.14742	0.14412	0.14554	0.14331	0.14086	0.14701	0.07703	0.07536	0.07296	0.07247
Item2Vec	0.1744	0.19101	0.19994	0.21624	0.10721	0.11551	0.1206	0.13141	0.06232	0.07352	0.06952	0.0628	0.03264	0.03883	0.03944	0.03486
GRU4Rec+	0.24186	0.26189	0.2738	0.28003	0.13034	0.1456	0.14972	0.15457	0.06384	0.0759	0.08589	0.10225	0.03709	0.03979	0.0429	0.05204
NARM	0.37549	0.37502	0.37441	0.3797	0.20959	0.20882	0.20331	0.20724	0.12301	0.15468	0.15912	0.16702	0.06782	0.08064	0.08039	0.07969
STAMP	0.34461	0.26508	0.35345	0.35459	0.20564	0.20701	0.20933	0.21206	0.09408	0.11142	0.11363	0.14463	0.0501	0.06093	0.05983	0.07407
NextItNet	0.38438	0.41508	0.42338	0.42323	0.24487	0.26622	0.27193	0.26929	0.00586	0.02396	0.05127	0.09053	0.00353	0.01348	0.02996	0.05038
SRGNN	0.357	0.36588	0.37042	0.37334	0.21429	0.2189	0.22263	0.22445	0.07751	0.14726	0.15678	0.17404	0.04411	0.07632	0.08005	0.09169
CSRM	0.40702	0.41197	0.41463	0.41316	0.25111	0.25543	0.25653	0.25684	0.15244	0.14358	0.1523	0.15882	0.08049	0.07627	0.0803	0.079
	TMALL								**ROCKET**							
	HR@5				**MRR@5**				**HR@5**				**MRR@5**			
	2d	**7d**	**14d**	**30d**	**2d**	**7d**	**14d**	**30d**	**2d**	**7d**	**14d**	**30d**	**2d**	**7d**	**14d**	**30d**
SPOP	0.14327	0.13483	0.13224	0.1304	0.08368	0.07883	0.07723	0.07596	0.13537	0.13356	0.13098	0.12938	0.0859	0.0788	0.07753	0.07623
AR	0.03543	0.03935	0.04136	0.04229	0.02035	0.02281	0.02396	0.02469	0.16457	0.14144	0.13154	0.13486	0.10271	0.08651	0.0838	0.08305
SR	0.03091	0.0353	0.03906	0.04078	0.01836	0.02119	0.02312	0.02421	0.1294	0.11404	0.11643	0.12294	0.08567	0.07098	0.07125	0.07736
VSKNN	0.07883	0.05627	0.04589	0.04485	0.06261	0.04714	0.04843	0.0321	0.45548	0.30787	0.25038	0.21187	0.32211	0.22249	0.18107	0.15659
SMF	0.05477	0.05789	0.06187	0.06319	0.02908	0.02995	0.0317	0.03583	0.17386	0.13601	0.14442	0.15035	0.102	0.07642	0.07664	0.07966
Item2Vec	0.00908	0.00738	0.00717	0.00734	0.00485	0.00392	0.00376	0.00398	0.03919	0.06648	0.07421	0.06584	0.02243	0.03738	0.04152	0.039
GRU4Rec+	0.04943	0.06804	0.07992	0.08865	0.02743	0.03817	0.04443	0.05135	0.11427	0.10118	0.10097	0.11632	0.07417	0.06367	0.05977	0.06836
NARM	0.06912	0.10992	0.12225	0.12595	0.03813	0.06139	0.06845	0.06961	0.18371	0.16607	0.1933	0.17737	0.1256	0.10077	0.1144	0.10207
STAMP	0.09296	0.08084	0.08592	0.08078	0.05638	0.04875	0.05089	0.0482	0.11957	0.12281	0.12268	0.13372	0.06733	0.07711	0.07059	0.07849
NextItNet	0.01779	0.04797	0.06563	0.06847	0.01092	0.02927	0.03974	0.04313	0.03438	0.03585	0.05945	0.09115	0.02419	0.02197	0.03407	0.05971
SRGNN	0.07548	0.09185	0.09919	0.10416	0.04656	0.05561	0.05991	0.06294	0.0625	0.07201	0.10463	0.16004	0.03972	0.04425	0.06321	0.09356
CSRM	0.05328	0.07535	0.0923	0.10183	0.03097	0.04297	0.05358	0.06006	0.20612	0.16894	0.17307	0.1703	0.12712	0.10264	0.1052	0.10271

Table A8. Training using random (1/16, All, 1/16, All) portions of the full training set for (RECSYS, CIKMCUP, TMALL, ROCKET) datasets, respectively. Evaluation is performed on the original testing sets. The HR, MRR, coverage, and popularity metrics with different cut-off thresholds are reported. Highest performance is made in bold for each dataset.

RECSYS	**HR@**			**MRR@**			**COV@**			**POP@**		
	1	**5**	**20**	**1**	**5**	**20**	**1**	**5**	**20**	**1**	**5**	**20**
S-POP	0.03545	0.12288	0.15136	0.03545	0.06877	0.07218	0.12645	0.17678	0.18881	0.05014	0.36861	0.41405
AR	0.09326	0.23779	0.29666	0.09326	0.14436	0.15269	0.12925	0.31183	0.40665	0.05395	0.04773	0.05782
SR	0.09684	0.23781	0.38576	0.09684	0.14695	0.16186	0.13818	0.34088	0.55249	0.05339	0.04686	0.04941
VSKNN	0.08387	0.18405	0.28761	0.08387	0.11959	0.12962	0.06223	0.13235	0.18946	0.07252	0.06734	0.05485
SMF	0.06524	0.25059	0.48033	0.06524	0.12911	0.15239	0.13977	0.29848	0.50202	0.03742	0.03584	0.0342
Item2Vec	0.04703	0.13127	0.28418	0.04703	0.07685	0.09154	0.14115	0.36694	0.67071	0.01948	0.01507	0.01152
GRU4Rec+	0.07526	0.23867	0.47698	0.07526	0.13098	0.15489	0.15376	0.37594	0.65268	0.02175	0.02043	0.01689
NARM	0.09725	0.30572	0.6317	0.09725	0.16939	0.19892	0.14718	0.32267	0.61052	0.04394	0.04098	0.03585
STAMP	0.10244	0.28161	0.50613	0.10244	0.16534	0.18781	0.12094	0.24481	0.47112	0.01721	0.01738	0.01885
NextItNet	**0.15686**	**0.35782**	0.55857	**0.15686**	**0.22923**	**0.24973**	0.07244	0.18841	0.33614	0.05459	0.04872	0.04451
SRGNN	0.11629	0.31111	0.53981	0.11629	0.18589	0.20882	0.1339	0.284	0.45424	0.01883	0.01918	0.01939
CSRM	0.1403	0.3521	**0.60582**	0.1403	0.21471	0.24048	0.11977	0.26834	0.46558	0.04012	0.03706	0.03272
CIKMCUP	**HR@**			**MRR@**			**COV@**			**POP@**		
	1	**5**	**20**	**1**	**5**	**20**	**1**	**5**	**20**	**1**	**5**	**20**
S-POP	0.03795	0.1143	0.13193	0.03795	0.06759	0.06972	0.02571	0.04627	0.04935	0.09372	0.39809	0.56251
AR	0.0466	0.1481	0.20761	0.0466	0.08187	0.09005	0.03412	0.1143	0.18302	0.12387	0.10161	0.10267
SR	0.04357	0.13856	0.2733	0.04357	0.07612	0.08958	0.0376	0.1261	0.28699	0.11482	0.0969	0.08535
VSKNN	0.05088	0.11536	0.17571	0.05088	0.07443	0.08065	0.01756	0.05278	0.09645	0.13022	0.10130	0.07209
SMF	0.03755	0.1546	0.3716	0.03755	0.07686	0.09758	0.03739	0.11054	0.24409	0.08022	0.0707	0.05845
Item2Vec	0.00907	0.02949	0.08229	0.00907	0.01627	0.02119	0.03858	0.13398	0.32806	0.00579	0.00579	0.00585
GRU4Rec+	0.03408	0.13105	0.32018	0.03408	0.06633	0.08437	0.0434	0.14069	0.32344	0.04954	0.04892	0.04176
NARM	0.04829	**0.19298**	**0.43868**	0.04829	0.0952	**0.12134**	0.03353	0.10048	0.29964	0.08762	0.08455	0.06647
STAMP	0.04121	0.15149	0.35362	0.04121	0.07869	0.09799	0.03691	0.11405	0.25159	0.03807	0.03702	0.03414
NextItNet	0.04688	0.15714	0.32277	0.04688	0.08473	0.10058	0.01531	0.05877	0.14956	0.08685	0.08713	0.08283
SRGNN	**0.05536**	0.18393	0.39509	**0.05536**	**0.09874**	0.11929	0.03452	0.10384	0.23304	0.03984	0.03896	0.03583
CSRM	0.04086	0.18665	0.43015	0.04086	0.08921	0.11293	0.02281	0.07181	0.17438	0.07868	0.0783	0.06864
TMALL	**HR@**			**MRR@**			**COV@**			**POP@**		
	1	**5**	**20**	**1**	**5**	**20**	**1**	**5**	**20**	**1**	**5**	**20**
S-POP	**0.04815**	**0.14347**	**0.19308**	**0.04815**	**0.06313**	**0.07056**	0.03859	0.11621	0.19768	0.04018	0.20512	0.27367
AR	0.01039	0.02975	0.04064	0.01039	0.01706	0.01855	0.0706	0.23079	0.38594	0.04326	0.03423	0.0182
SR	0.00983	0.02522	0.04385	0.00983	0.01524	0.01716	0.09624	0.30236	0.53731	0.03828	0.0316	0.02423
VSKNN	0.03073	0.05684	0.08502	0.03073	0.03948	0.04306	0.07479	0.27285	0.57219	0.06251	0.03870	0.02520
SMF	0.01322	0.04575	0.09756	0.01322	0.02446	0.02945	0.08042	0.22742	0.44955	0.039	0.02898	0.02397
Item2Vec	0.00234	0.00772	0.01909	0.00234	0.00417	0.00525	0.08061	0.23045	0.41993	0.00467	0.00432	0.00415
GRU4Rec+	0.01421	0.04454	0.07763	0.01421	0.02472	0.02813	0.11887	0.30595	0.56886	0.02515	0.02	0.01577
NARM	0.01707	0.05639	0.13195	0.01707	0.03128	0.03805	0.15697	0.21753	0.30538	0.13894	0.13637	0.13393
STAMP	0.02826	0.07118	0.11514	0.02826	0.04386	0.04826	0.10807	0.22155	0.3633	0.01271	0.01334	0.01315
NextItNet	0.0055	0.01732	0.02926	0.0055	0.00964	0.01074	0.02945	0.07463	0.12817	0.06225	0.07357	0.08514
SRGNN	0.02639	0.06309	0.10492	0.02639	0.03966	0.04379	0.07253	0.18728	0.35322	0.01267	0.01332	0.01306
CSRM	0.01637	0.04509	0.08155	0.01637	0.02653	0.03011	0.05914	0.14149	0.27906	0.0374	0.03444	0.03008
ROCKET	**HR@**			**MRR@**			**COV@20**			**POP@20**		
	1	**5**	**20**	**1**	**5**	**20**	**1**	**5**	**20**	**1**	**5**	**20**
S-POP	0.04035	0.12477	0.14753	0.04035	0.0754	0.07784	0.01588	0.02265	0.02461	0.05994	0.40345	0.44083
AR	0.06166	0.14587	0.18291	0.06166	0.09149	0.09656	0.01992	0.07009	0.10894	0.09101	0.07402	0.08349
SR	0.06083	0.14546	0.22512	0.06083	0.09048	0.09888	0.02153	0.07716	0.17382	0.07569	0.06229	0.05799
VSKNN	**0.09787**	0.17365	0.21300	**0.09787**	0.12694	0.13113	0.01346	0.04020	0.07662	0.08880	0.07363	0.04940
SMF	0.0329	0.15911	0.31513	0.0329	0.07583	0.09209	0.02084	0.07498	0.18105	0.06033	0.05043	0.04155
Item2Vec	0.01584	0.04941	0.10507	0.01584	0.02798	0.0332	0.60701	0.23934	0.66277	0.00809	0.00687	0.00627
GRU4Rec+	0.05656	0.17949	0.31655	0.05656	0.09269	0.10776	0.02462	0.09038	0.23303	0.03512	0.03326	0.02954
NARM	0.09294	**0.22845**	**0.38783**	0.09294	**0.14104**	**0.15654**	0.02415	0.08073	0.25337	0.05659	0.05901	0.0462
STAMP	0.05199	0.15824	0.29445	0.05199	0.08886	0.10251	0.0192	0.05846	0.12198	0.02882	0.02912	0.02721
NextItNet	0.09208	0.21094	0.32199	0.09208	0.13507	0.14612	0.01288	0.05247	0.13681	0.07332	0.07103	0.06876
SRGNN	0.09231	0.21443	0.34087	0.09231	0.13575	0.14845	0.01764	0.06143	0.15321	0.02836	0.02839	0.02595
CSRM	0.08091	**0.22895**	0.37966	0.08091	0.13351	0.14886	0.01852	0.06544	0.16379	0.06413	0.05781	0.04674

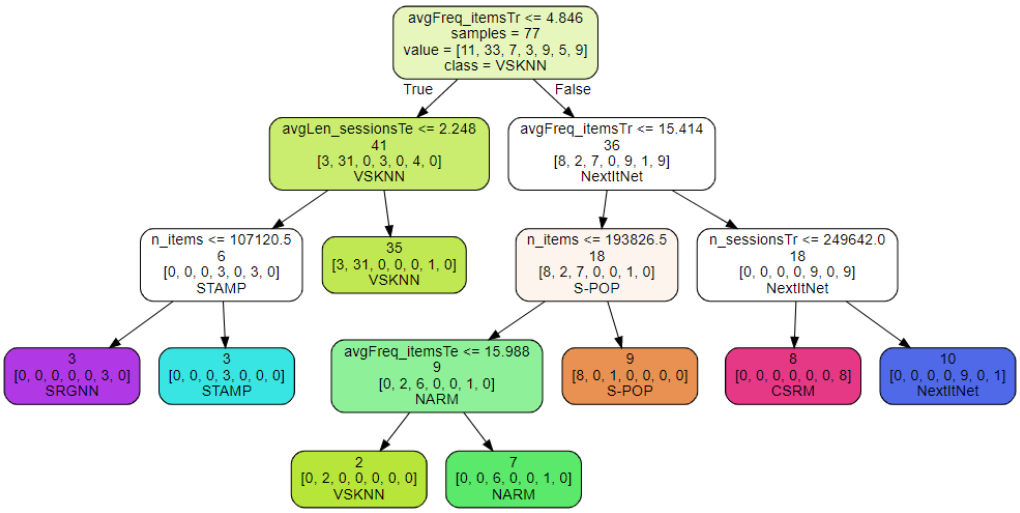

Figure A1. Decision tree model trained to predict the best model to perform based on dataset characteristics.

Appendix C. Hyper-Parameters' Ranges and Discretization Levels for Each Model

Table A9. List of tuned hyper-parameters ranges and discretization level for each model.

Model Name	Hyper-parameters					
S-POP	**TopN**					
	10–1000					
	Scale: Linear - Step: 10					
AR	**Pruning**					
	0–10					
	Scale: Linear-Step: 1					
SR	**Pruning**	**Weighting**				
	0–10	linear, same, div, log, quadratic				
	Scale: Linear-Step: 1					
VSKNN	**K**	**Sample**	**Sampling**	**Similarity Distance**	**Weighting**	**Weighting_Score**
	50–500	100–10,000	Random-Recent	jaccard, cosine, binary, tanimoto	linear, same, div, log, quadratic	linear, same, div, log, quadratic
	Scale: Linear-Step: 10	Scale: Linear - Step: 100				
SMF	**Learning Rate**	**Factors**	**Negative Samples**	**Momentum**	**Regularization**	**Dropout**
	0.001–0.1	50–200	100–4000	0.2	0.5	0.1–0.5
	$10^n, n \in \{-3, -2, -1\}$	Scale: Linear-Step: 10	Scale: Linear-Step: 100			Scale: Linear-Step: 0.1
Item2Vec	**Skip Probability**	**Batch Size**	**Max-Epochs**	**Loss Fun**		
	0.1	32	10	bpr-max		
	Starting Learning Rate	**Final Learning Rate**	**Window Size**	**Embedding Dim.**	**Neg. Sampling**	**Max-Epochs**
	0.01–0.05	0.00001–0.001	3–9	32–512	10–100	20
	Scale: Linear-Step: 0.01	$10^n, n \in \{-5, -4, -3\}$	Scale: Linear - Step: 1	$2^n, n \in \{5, 6, 7, 8, 9\}$	Scale: Linear-Step: 10	
	Freq. Threshold~	**SubSampling**				
	1	0.0001				
GRU4Rec+	**Learning Rate**	**Neurons/Layer**	**Hidden Layers**	**Dropout**	**Loss Fun**	**Optimizer**
	0.001–0.1	50–200	1	0–0.5	BPR-Max	Adagrad
	$10^n, n \in \{-3, -2, -1\}$	Scale: Linear-Step: 10		Scale: Linear-Step: 0.1		
	Momentum	**Activation**	**Batch Size**	**Max-Epochs**		
	0.1	tanh	32	10		
NARM	**Learning Rate**	**Batch Size**	**Hidden Units**	**Embedding Dim.**	**Optimizer**	**Max-Epochs**
	0.0001–0.01	512	50–200	50–200	Adam	10
	$10^n, n \in \{-4, -3, -2\}$		Scale: Linear - Step: 10	Scale: Linear-Step: 10		
STAMP	**Learning Rate**	**Hidden Size**	**Max Grad. Norm**	**Max-Epochs**	**Embedding Dim.**	**Std-dev**
	0.001–0.005	50–200	50–250	10	50–200	0.05
	Scale: Linear-Step: 0.001	Scale: Linear-Step: 10	Scale: Linear-Step: 10		Scale: Linear - Step: 10	
	Activation Fun.	**Batch Size~**				
	sigmoid	64				
NextItNet	**Learning Rate**	**Dilated Channels**	**Dilations**	**Batch Size**	**Max-Epochs**	**Neg. Sampling**
	0.0001–0.01	50–200	1,2,4	32	20	FALSE
	$10^n, n \in \{-4, -3, -2\}$	Scale: Linear-Step: 10				
SRGNN	**Learning Rate**	**Batch Size**	**Max-Epochs**	**Neurons/Layer**	**LR Decay**	**L2 Penalty**
	0.0001–0.01	50–200	10	50–200	0.1	0.00001
	$10^n, n \in \{-4, -3, -2\}$	Scale: Linear-Step: 10		Scale: Linear-Step: 10		
CSRM	**Learning Rate**	**Memory Size**	**Hidden Units**	**Max-Epochs**	**Embedding Dim.**	**Memory Dim.**
	0.0001–0.01	128–1024	50–200	20	50–200	50–200
	$10^n, n \in \{-4, -3, -2\}$	$2^n, n \in \{7, 8, 9, 10\}$	Scale: Linear - Step: 10		Scale: Linear - Step: 10	Scale: Linear - Step: 10
	Batch Size	**Shift Range**				
	512	1				

References

1. Lee, D.; Hosanagar, K. Impact of Recommender Systems on Sales Volume and Diversity 2014. Available online: https://www.semanticscholar.org/paper/Impact-of-Recommender-Systems-on-Sales-Volume-and-Lee-Hosanagar/8109f0a47f433b32979d9b3f2da9facee5eba4ad (accessed on 1 August 2022).
2. Liu, D.C.; Rogers, S.; Shiau, R.; Kislyuk, D.; Ma, K.C.; Zhong, Z.; Liu, J.; Jing, Y. Related pins at pinterest: The evolution of a real-world recommender system. In Proceedings of the 26th International Conference on World Wide Web Companion, Perth Australia 3–7 April 2017; International World Wide Web Conferences Steering Committee: Geneva, Switzerland, 2017; pp. 583–592.
3. Van den Oord, A.; Dieleman, S.; Schrauwen, B. Deep content-based music recommendation. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, Nevada, USA, 5–10 December 2013; pp. 2643–2651.
4. Deldjoo, Y.; Elahi, M.; Cremonesi, P.; Garzotto, F.; Piazzolla, P.; Quadrana, M. Content-based video recommendation system based on stylistic visual features. *J. Data Semant.* **2016**, *5*, 99–113.
5. Wang, H.; Zhang, F.; Xie, X.; Guo, M. DKN: Deep knowledge-aware network for news recommendation. In Proceedings of the 2018 World Wide Web Conference, Lyon, France, 23–27 April 2018; International World Wide Web Conferences Steering Committee: Geneva, Switzerland, 2018; pp. 1835–1844.
6. Koren, Y. Collaborative filtering with temporal dynamics. In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, 28 Jun–1 July 2009; ACM: New York, NY, USA, 2009; pp. 447–456.
7. Song, Y.; Elkahky, A.M.; He, X. Multi-rate deep learning for temporal recommendation. In Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval, Pisa, Italy, 17–21 July 2016; ACM: New York, NY, USA, 2016; pp. 909–912.
8. Fan, W.; Ma, Y.; Li, Q.; He, Y.; Zhao, Y.E.; Tang, J.; Yin, D. Graph Neural Networks for Social Recommendation. 2019. Available online: https://ieeexplore.ieee.org/document/9139346 (accessed on 1 August 2022).
9. Deng, S.; Huang, L.; Xu, G.; Wu, X.; Wu, Z. On deep learning for trust-aware recommendations in social networks. *IEEE Trans. Neural Netw. Learn. Syst.* **2016**, *28*, 1164–1177.
10. Mohallick, I.; De Moor, K.; Özgöbek, Ö.; Gulla, J.A. Towards New Privacy Regulations in Europe: Users' Privacy Perception in Recommender Systems. In Proceedings of the International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage, Melbourne, NSW, Australia, 11–13 December 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 319–330.
11. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436.
12. Wang, H.; Wang, N.; Yeung, D.Y. Collaborative deep learning for recommender systems. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, Australia, 10–13 August 2015; ACM: New York, NY, USA, 2015; pp. 1235–1244.
13. Kim, D.; Park, C.; Oh, J.; Lee, S.; Yu, H. Convolutional matrix factorization for document context-aware recommendation. In Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, 15–19 September 2016; ACM: New York, NY, USA, 2016; pp. 233–240.
14. Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; Tikk, D. Session-based recommendations with recurrent neural networks. *arXiv* **2015**, arXiv:1511.06939.
15. Kang, W.C.; McAuley, J. Self-attentive sequential recommendation. In Proceedings of the 2018 IEEE International Conference on Data Mining (ICDM), S ingapore, 17–20 November 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 197–206.
16. Liu, Q.; Zeng, Y.; Mokhosi, R.; Zhang, H. STAMP: Short-term attention/memory priority model for session-based recommendation. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; ACM: New York, NY, USA, 2018; pp. 1831–1839.
17. Wu, S.; Tang, Y.; Zhu, Y.; Wang, L.; Xie, X.; Tan, T. Session-based recommendation with graph neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 346–353.
18. Ludewig, M.; Jannach, D. Evaluation of session-based recommendation algorithms. *User Model. User-Adapt. Interact.* **2018**, *28*, 331–390.
19. Jannach, D.; Ludewig, M. When recurrent neural networks meet the neighborhood for session-based recommendation. In Proceedings of the Eleventh ACM Conference on Recommender Systems, Como, Italy, 27–31 August 2017; pp. 306–310.
20. Ludewig, M.; Mauro, N.; Latifi, S.; Jannach, D. Empirical Analysis of Session-Based Recommendation Algorithms. *arXiv* **2019**, arXiv:1910.12781.
21. Kamehkhosh, I.; Jannach, D.; Ludewig, M. A Comparison of Frequent Pattern Techniques and a Deep Learning Method for Session-Based Recommendation. In Proceedings of the RecTemp@ RecSys, Italy, 2017; pp. 50–56. Available online: https://www.semanticscholar.org/paper/A-Comparison-of-Frequent-Pattern-Techniques-and-a-Kamehkhosh-Jannach/5f6ff6990a1afac7e7b5616283b1d4e67a9d034f (accessed on 1 August 2022).
22. Quadrana, M.; Cremonesi, P.; Jannach, D. Sequence-aware recommender systems. *ACM Comput. Surv. (CSUR)* **2018**, *51*, 66.
23. Jannach, D.; Lerche, L.; Jugovac, M. Adaptation and evaluation of recommendations for short-term shopping goals. In Proceedings of the 9th ACM Conference on Recommender Systems, Vienna, Austria, 16–20 September 2015; ACM: New York, NY, USA, 2015; pp. 211–218.

24. Bonnin, G.; Jannach, D. Automated generation of music playlists: Survey and experiments. *ACM Comput. Surv. (CSUR)* **2015**, *47*, 26.
25. Garcin, F.; Dimitrakakis, C.; Faltings, B. Personalized news recommendation with context trees. *arXiv* **2013**, arXiv:1303.0665.
26. Hosseinzadeh, A.; Hariri, N.; Mobasher, B.; Burke, R. Adapting recommendations to contextual changes using hierarchical hidden markov models. In Proceedings of the 9th ACM Conference on Recommender Systems, Vienna, Austria, 16–20 September 2015; pp. 241–244.
27. Tavakol, M.; Brefeld, U. Factored MDPs for detecting topics of user sessions. In Proceedings of the 8th ACM Conference on Recommender Systems, Foster City, CA, USA, 6–10 October 2014; ACM: New York, NY, USA, 2014; pp. 33–40.
28. Cheng, C.; Yang, H.; Lyu, M.R.; King, I. Where you like to go next: Successive point-of-interest recommendation. In Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, Beijing, China, 3–9 August 2013.
29. He, R.; McAuley, J. Fusing similarity models with markov chains for sparse sequential recommendation. In Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM), Barcelona, Spain, 12–15 December 2016; IEEE: Piscataway, MJ, USA, 2016; pp. 191–200.
30. Hidasi, B.; Karatzoglou, A. Recurrent neural networks with top-k gains for session-based recommendations. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, Torino, Italy, 22–26 October 2018. ACM: New York, NY, USA, 2018; pp. 843–852.
31. Hidasi, B.; Quadrana, M.; Karatzoglou, A.; Tikk, D. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, 15–19 September 2016; ACM: New York, NY, USA, 2016; pp. 241–248.
32. Quadrana, M.; Karatzoglou, A.; Hidasi, B.; Cremonesi, P. Personalizing session-based recommendations with hierarchical recurrent neural networks. In Proceedings of the Eleventh ACM Conference on Recommender Systems, Como, Italy, 27–31 August 2017; ACM: New York, NY, USA, 2017; pp. 130–137.
33. Ruocco, M.; Skrede, O.S.L.; Langseth, H. Inter-session modeling for session-based recommendation. In Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems, Como, Italy, 27 August 2017; ACM: New York, NY, USA, 2017; pp. 24–31.
34. Tuan, T.X.; Phuong, T.M. 3D convolutional networks for session-based recommendation with content features. In Proceedings of the Eleventh ACM Conference on Recommender Systems, Como, Italy, 27–31 August 2017; ACM: New York, NY, USA, 2017; pp. 138–146.
35. Yuan, F.; Karatzoglou, A.; Arapakis, I.; Jose, J.M.; He, X. A Simple Convolutional Generative Network for Next Item Recommendation. In Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, Melbourne, VIC, Australia, 11–15 February 2019; ACM: New York, NY, USA, 2019; pp. 582–590.
36. Wang, M.; Ren, P.; Mei, L.; Chen, Z.; Ma, J.; de Rijke, M. A Collaborative Session-based Recommendation Approach with Parallel Memory Modules 2019. Available online: https://dl.acm.org/doi/10.1145/3331184.3331210#:~:text=We%20propose%20a%20Collaborative%20Session-based%20Recommendation%20Machine%20%28CSRM%29%2C,Encoder%20%28IME%29%20and%20an%20Outer%20Memory%20Encoder%20%28OME%29. (accessed on 1 August 2022).
37. Graves, A.; Wayne, G.; Danihelka, I. Neural turing machines. *arXiv* **2014**, arXiv:1410.5401.
38. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 5998–6008.
39. Li, J.; Ren, P.; Chen, Z.; Ren, Z.; Lian, T.; Ma, J. Neural attentive session-based recommendation. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, Singapore, 6–10 November 2017; ACM New York, NY, USA, 2017; pp. 1419–1428.
40. Chen, W.; Cai, F.; Chen, H.; de Rijke, M. A Dynamic Co-attention Network for Session-based Recommendation. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019; pp. 1461–1470.
41. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
42. Sun, F.; Liu, J.; Wu, J.; Pei, C.; Lin, X.; Ou, W.; Jiang, P. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. *arXiv* **2019**, arXiv:1904.06690.
43. Barkan, O.; Koenigstein, N. Item2Vec: Neural Item Embedding for Collaborative Filtering. *arXiv* **2016**, arXiv:1603.04259.
44. Tang, J.; Wang, K. Personalized top-n sequential recommendation via convolutional sequence embedding. In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, Marina Del Rey, CA, USA, 5–9 February 2018; pp. 565–573.
45. Adomavicius, G.; Tuzhilin, A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.* **2005**, *17*, 734–749.
46. Agrawal, R.; Imieliński, T.; Swami, A. Mining association rules between sets of items in large databases. *ACM Sigmod Rec.* **1993**, *22*, 207–216.
47. Steck, H. Item popularity and recommendation accuracy. In Proceedings of the Fifth ACM conference on Recommender Systems, Chicago, IL, USA, 23–27 October 2011; pp. 125–132.

48. Wen, Z. Recommendation System Based on Collaborative Filtering. CS229 Lecture Notes, 12 December 2008. Available online: https://www.dominodatalab.com/blog/recommender-systems-collaborative-filtering (accessed on 1 August 2022).
49. Kabbur, S.; Ning, X.; Karypis, G. Fism: Factored item similarity models for top-n recommender systems. In Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, IL, USA 11–14 August 2013; ACM: New York, NY, USA, 2013; pp. 659–667.
50. Hidasi, B.; Karatzoglou, A. Recurrent Neural Networks with Top-k Gains for Session-based Recommendations. *arXiv* **2018**, arXiv:1706.03847 .
51. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. *arXiv* **2013**, arXiv:1301.3781.
52. Shani, G.; Gunawardana, A. Evaluating recommendation systems. In *Recommender Systems Handbook*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 257–297.
53. Kim, H.K.; Kim, H.; Cho, S. Bag-of-concepts: Comprehending document representation through clustering words in distributed representation. *Neurocomputing* **2017**, *266*, 336–352.
54. Barry-Straume, J.; Tschannen, A.; Engels, D.W.; Fine, E. An evaluation of training size impact on validation accuracy for optimized convolutional neural networks. *SMU Data Sci. Rev.* **2018**, *1*, 12.
55. Jannach, D.; Ludewig, M.; Lerche, L. Session-based item recommendation in e-commerce: On short-term intents, reminders, trends and discounts. *User Model. User-Adapt. Interact.* **2017**, *27*, 351–392.
56. Kumar, S.; Zhang, X.; Leskovec, J. Predicting dynamic embedding trajectory in temporal interaction networks. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; ACM: New York, NY, USA, 2019; pp. 1269–1278.
57. Hutter, F.; Babic, D.; Hoos, H.H.; Hu, A.J. Boosting verification by automatic tuning of decision procedures. In Proceedings of the Formal Methods in Computer Aided Design (FMCAD'07), Austin, TX, USA, 11–14 November 2007; IEEE: Piscataway, NJ, USA, 2007; pp. 27–34.
58. Hutter, F.; Kotthoff, L.; Vanschoren, J. *Automated Machine Learning*; Springer: Berlin/Heidelberg, Germany, 2019.
59. Elshawi, R.; Maher, M.; Sakr, S. Automated Machine Learning: State-of-The-Art and Open Challenges. *arXiv* **2019**, arXiv:1906.02287.