UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Technology

Perseverance Munga Ngoy

Hyper-parameter Optimization of Session-based Recommendation Systems

Bachelor's Thesis (12 ECTS)

Curriculum Science and Technology

Supervisor(s):	Mohamed Maher, MSc
	Assoc. Prof. Cagri Ozcinar
	Prof. Gholamreza Anbarjafari

Tartu 2021

Hyper-parameter optimization of session-based recommendation systems

Abstract: This thesis proposes a comprehensive framework to recommend session-based algorithms and tune their *hyperparameters*. In the first part of this study, we present a comprehensive evaluation of the state-of-the-art deep learning approaches used in the session-based recommendation. Furthermore, we present an evaluation of neural-based models' performance using the *AutoML Neural Network Intelligence framework*. In session-based recommendation, the system counts on the sequence of events made by a user within the same session to predict and endorse other more likely items to correlate with his 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). The first evaluation shows that advanced neural network models outperform the baseline techniques in most of the scenarios. However, we found that these models suffer more in long sessions when there is a drift in user interests, and there is not enough data to correctly model different items during training. The first study suggests that using hybrid models of different approaches combined with baseline algorithms could result in session-based recommendations based on dataset characteristics. The second evaluation shows that *AutoML* improved the performance of neural-based models by using a state-of-the-art algorithm such as *Bayesian Optimization and hyperband (BOHB)* to replace the random selection of hyper-parameters.

Keywords: Machine Learning, Deep Learning, Automated Machine Learning, hyper-parameter optimization, Recommendation systems.

CERCS: T111 Imaging, image processing; T121 Signal processing;

Hüperparameetrite optimeerimine sessioonipõhistele soovitussüsteemidele

abstrakt: See lõputöö pakub välja tervikliku raamistiku seansipõhiste algoritmide soovitamiseks ja nende hüperparameetrite häälestamiseks. Selle uuringu esimeses osas esitame põhjaliku hinnangu tipptasemel süvaõppe lähenemisviisidele, mida kasutatakse sessioonipõhistes soovitustes. Lisaks esitame hinnangu närvipõhiste mudelite toimivusele, kasutades AutoML Neural Network Intelligence raamistikku. Sessioonipõhistes soovituses arvestab süsteem sama seansi jooksul kasutaja poolt tehtud sündmuste jadaga, et ennustada ja soovitada muid tõenäolisemaid tooteid, mis on tema eelistustega korrelatsioonis. Meie ulatuslikud katsed uurivad lähtemeetodeid (nt lähimad naabrid ja mustrihanke algoritmid) ja süvaõppe lähenemisi (nt rekurrentsed närvivõrgud, graafipõhised närvivõrgud ja tähelepanupõhised võrgustikud). Esmane hinnang näitab, et uudsed närvivõrgu mudelid ületavad enamikes stsenaariumites lähtemeetodeid. Siiski leidsime, et need mudelid kannatavad rohkem pikkadel seanssidel, kui kasutajate huvi on triivinud ja pole piisavalt andmeid, et treeningu vältel erinevaid esemeid õigesti näidata. Esimene uuring viitab sellele, et erinevates lähenemisviisides hübriidmudelite kasutamine koos algtaseme algoritmidega, võib andmekogumi omaduste põhjal tekitada sessioonipõhise soovituse. Teine hinnang näitab, et AutoML parandas närvipõhiste mudelite toimivust, kasutades hüperparameetrite juhusliku valiku asendamiseks tipptehnoloogilist algoritmi nagu Bayesi optimeerimine ja hüperriba (BOHB).

Märksõnad: Masinõpe, süvaõpe, automatiseeritud masinõpe, hüperparameetrite optimeerimine, soovitussüsteemid

CERCS:T111 Pilditehnika; T121 Signaalitöötlus;

Acknowledgments

I would like to thank my thesis supervisors Mohamed Maher Msc, Assoc. Prof. Cagri Ozcinar, Prof. Gholamreza Anbarjafari for the guidance, encouragement, and advice they have provided to me throughout a full academic year. It would not have been possible for me to defend if it was not with their help and support. They always had the patience to listen to my ideas, provide fruitful feedback and, steer me in the right direction whenever I needed it the most. I am particularly very grateful to Prof. Gholamreza Anbarjafari for the many opportunities he gave me, like joining the iCV Lab at the Institute of Technology, University of Tartu, where I gained various research skills, learned about many interesting topics in the field of Machine Learning and enjoyed the exciting discussions among my colleagues in the group.

I would also like to express my profound gratitude to my parents, brothers, sisters, and friends for their endless support and encouragement throughout all years of my studies. Finally, I dedicate my work to the memory of my father, Floribert Kakudji Ngoy, who always believed in my ability to be successful in the academic arena.

Perseverance Munga Ngoy
May, 2021

Contents

Abbreviation	Definition
KDE	Kernel density estimation
DNN	Deep neural networks
DNNs	Deep Neural Networks
CNNs	Convolutional Neural Networks

Table 1. List of abbreviations

Symbol	Description
$\mathcal{I}$	Set of available items
N	Total number of available items = $\|\mathcal{I}\|$
A	Number of hyperparameters = $\|A\|$
I_n	Item with index $n \in \mathcal{I}$ where $n \in N$
x_{i_t}	The i^{th} 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 $\{x1_t, x2_t, ..., x_{L_t}\}$
$1_{EQ}(x, y)$	= 1 if $x = y$, and 0 otherwise.
S_D, S_{TR}, 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 $\mathbf{Y}$, and 0 otherwise
$r_{IN}(x, \mathbf{Y})$	= rank of x if it is one of elements in vector $\mathbf{Y}$, 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
$argmax_K(\hat{Y})$	Index of items with top K predicted scores.
$\mathcal{U}(Y)$	Set of unique items in vector $\mathbf{Y}$
$\mathcal{F}(Y)$	Frequencies of items in vector $\mathbf{Y}$ from the training set

Table 2. Notation used in mathematical formulas described throughout the thesis.

1 Introduction

Recommender systems (RS) are software applications that help users to find the item of interest based on their navigation behavior, usually with the aim of helping them in order to overcome information overload or make informed choices. This process is typically based on the assumption that long-term preference information about the individual users is available to the system, most commonly in the form of a user-item rating matrix [51].

Recommender systems have become a significant piece of any effective business that helps meet the user's needs and boost the business sales volume. On classic online sites, different sorts of user activity can be recorded. For example, a user views an item or makes a buy. A few actions made by the user could be enough to identify items of interest to him. These recorded activities and the recognized patterns are then used to figure out recommendations that match an individual user's preferences. Recommender system can be beneficial, serve different purposes, and add more value for both users and service providers. For example, by helping the users to discover supplementary items of interest or by helping providers to boost specific sectors of their item range. In addition, in the e-commerce domain, several other applications have been a subject of research, particularly media streaming[17], music, travel, web page navigation, tourism [5], and even news recommendations [54].

1.1 Importance of session-based recommendation systems in electronic commerce

Due to the recent introduction of the General Data Protection Regulation (GDPR)[1] rules regulating the collection of personalized data about the user in order to protect their privacy, more consideration has been given to session-based recommendation relying on the user navigation behavior and the sequence of action and clicks on different items exclusively to suggest the next item that matches the user's interest. Furthermore, most academic research is concerned about methodologies that personalize the recommendations according to long-term user profiles [39]. In some real-world applications, although such long-term profiles regularly do not exist and recommendations in this manner must be made exclusively dependent on a user's observed behavior during an ongoing session. Given the high practical significance of the problem, an expanded interest in this problem can be observed lately, prompting various propositions for session-based recommendation architecture that ordinarily intend to predict the user's quick next actions.

1.2 Definition of session-based recommendation

Session-based recommendation is the task of recommending the next item based on previously recorded user interactions. This task focuses on sequential mining compared to conventional user-item settings, and only restricted user interactions within a short time frame can be exploited. Figure1 shows an example of a session-based recommendation where the user has a stream of

[1]https://gdpr-info.eu/

click events on multiple items. The recommendation system tries to predict the next items to be viewed by the same user based on the information made available in a single session only.

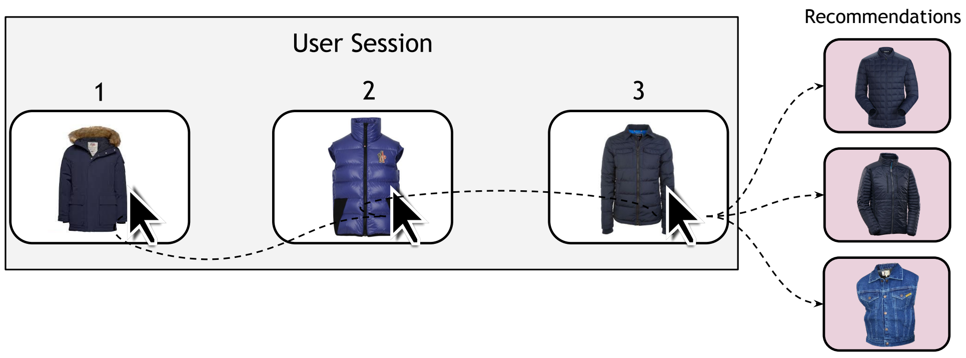


Figure 1. An example of session-based recommendation. 3 different items are clicked by the user. The recommendation system predicts other candidate items that can be viewed next by the same user.

1.3 Evaluating current recommendation systems

Few studies were performed to evaluate the session-based algorithms. Jannach et al. contrasted a heuristics-based nearest neighbour baseline with basic recurrent neural network algorithms (RNN) [47]. This study showed that deep learning techniques, such as neighbour-hood methods, still lag behind basic algorithms. However, in a session-based recommendation, several developments have been suggested in the use of deep learning over the last few years, leading to the emergence of many neural-based architectures such as Gated recurrent neural networks for session-based recommendation (GRU4Rec) [10] and Neural item to vector (Item2Vec) [11]. For example, a study was conducted to compare several session-based recommendation baseline algorithms using four e-commerce datasets and four others in music and playlists' recommendation. We depend on this study by Ludewig et al [48] to select the most promising baseline techniques to investigate their performance in comparison with deep learning methods. Also, in our work, we rely on this study to select the most promising neural-based methods to explore their efficiency by automatically improving the selection of hyperparameters by using an Automated Machine Learning (AutoML) framework named Neural Network Intelligence[2] instead of the random selection of hyperparameters.

However, the study by Ludewig et al [48] included only a single neural-based model and lacks the evaluations with various deep-learning approaches. Furthermore, this research has recently been expanded to include state-of-the-art models of deep learning in session-based recommendation. An empirical evaluation was carried out by training models on complete

[2]https://nni.readthedocs.io/en/stable/index.html

datasets in [48], [42], and the results in these studies showed that no model outperforms the others in all datasets. However, there was no enough reasoning behind why exactly a particular model outperforms the others on a specific dataset.

Furthermore, each algorithm/neural-network model has many hyper-parameters that can take tons of values. Training these models can be computationally challenging and obtaining good results requires extensive parameter tuning. For example, training GRU4Rec with one parameter can take an average of 6-7 minutes on a GPU to find the best number of samples according to the study made by [34]. In the same study, they trained the architecture on two parameters (different loss functions and the number of samples). The results showed that the training required, on average, 30 minutes to find the best performance for a particular set of these parameters. Also, according to the study made by [48]. For the same architecture using different hyperparameter configurations. The parameters for this algorithm were optimized for each dataset. Due to the computational complexity of the method, they restricted the layer size for GRU4Rec and used a randomized search method with 100 iterations for the remaining parameters as described by [Hidasi and Karatzoglou 2017]. In each iteration, the learning rate, the drop-out factor, the momentum and the loss function were determined in a randomized process to find the maximum hit rate for a list length of 20. GRU4Rec needed at least 6 hours on GPU to learn a model for a single data split..

In this thesis, the previous assessment studies [48], [42], [40] evaluating the overall performance of various models on a few datasets are expanded by adding the following key contributions:

- In session-based recommendations, a thorough evaluation and benchmarking of the state-of-the-art neural-based methods, including recurrent neural networks, convolutional neural networks, and attention mechanisms, is carried out along with a group of the most common base-line techniques in the field of recommendation such as nearest-neighbors, frequent pattern mining and matrix factorization.

- We improved the performance of the state-of-the-art neural-based models in session-based recommendation by turning their hyper-parameters using automated machine learning (AutoML) concepts via the Neural Network Intelligence (NNI) framework[3].

- The various approaches' performance is evaluated based on different train and test dataset split obtained from four different e-commerce benchmark datasets, namely RECSYS[4], CIKMCUP[5], RETAIL ROCKET[6], TMALL[7].

- Based on different dataset attributes, our experiments expand on the evaluation process. Therefore, we divide the datasets according to values of different characteristics such as session length, item frequency, and data sizes, which may expose why certain models

[3]https://nni.readthedocs.io/en/latest/index.html

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

[5]https://cikm2016.cs.iupui.edu/cikm-cup/

[6]https://www.kaggle.com/retailrocket/ecommerce-dataset

[7]https://www.tmall.com/

are poorly performing and open up new research horizons on what kind of changes were required for each model that were difficult to observe from previous studies.

- An interpretable decision tree model was used to correctly suggest the best performing model according to the dataset features.
- Present limitations of session-based recommendation systems are addressed with suggested ways to address these problems that in certain areas achieve positive results.
- A final assemble method is proposed to include all the experiments cited above by having a framework to recommend session based recommendation and tuning its hyperparameters.

1.4 Thesis structure

In the following Section 2, a survey of session-based recommendation systems has been discussed. Section 3 presents a detailed description of different algorithms and models evaluated in our experiment setup and the research questions to be answered. Furthermore, it presents a detailed description of the Neural Network Intelligence (NNI). Section 5 describes the experimental setup and the research question to be answered, and Section 4 shows the results and discussion of the evaluation experiments. Finally, Section 6 represents the central insights of the final assemble to include all the experiments. Figure 2 shows a summary of the thesis content with an overview of its flow and organization.

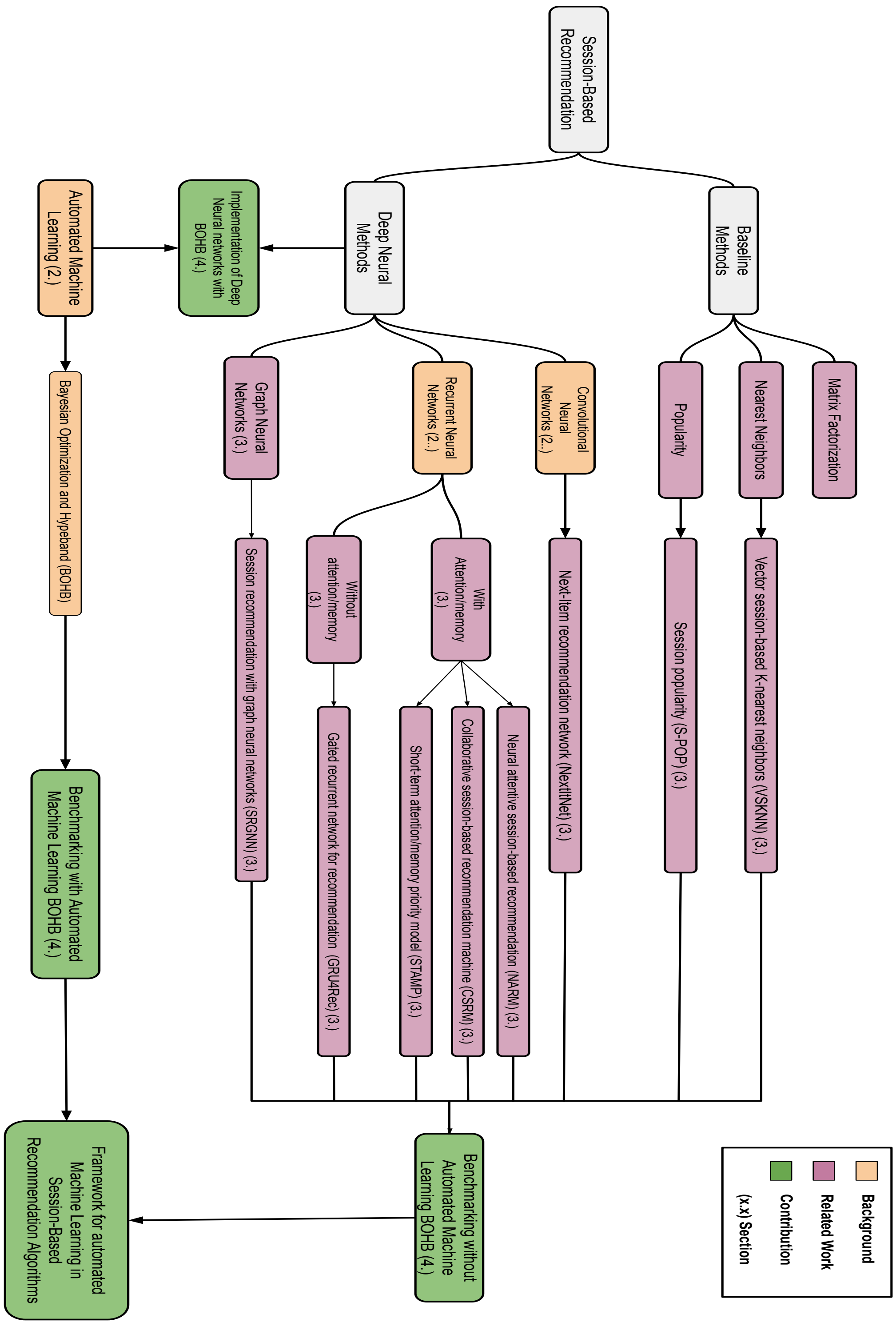

Session-Based Recommendation
Baseline Methods
Deep Neural Methods
Matrix Factorization
Nearest Neighbors
Popularity
Convolutional Neural Networks (2..)
Recurrent Neural Networks (2..)
Graph Neural Networks (3.)
Implementation of Deep Neural networks with BOHB (4.)
Automated Machine Learning (2.)
Vector session-based K-nearest neighbors (VSKNN) (3.)
Session popularity (S-POP) (3.)
Next-Item recommendation network (NextItNet) (3.)
With Attention/memory (3.)
Without attention/memory (3.)
Session recommendation with graph neural networks (SRGNN) (3.)
Bayesian Optimization and Hypeband (BOHB)
Neural attentive session-based recommendation (NARM) (3.)
Collaborative session-based recommendation machine (CSRM) (3.)
Short-term attention/memory priority model (STAMP) (3.)
Gated recurrent network for recommendation (GRU4Rec) (3.)
Benchmarking with Automated Machine Learning BOHB (4.)
Benchmarking without Automated Machine Learning BOHB (4.)
Framework for automated Machine Learning in Session-Based Recommendation Algorithms
🟧 Background
🟪 Related Work
🟩 Contribution
(x.x) Section

Figure 2. A high-level overview of the thesis flow and organization in the following section.

2 Background

In the following sections, we start by introducing the notation being used in our study and an overview of the necessary background information, including different types of Recommendation systems in e-commerce and a detailed explanation of the session-based recommendation and its importance. Then, we introduce deep neural networks (DNN) and discuss different types of DNN and their use in modern applications. After that, Attention in neural networks is being discussed. Finally, we present several Automated Machine Learning frameworks and how they have been used to automate processes and to improve deep neural network performance.

2.1 Types of recommendation systems in e-commerce

Early approaches tackling session-based recommendation problems in literature were based on nearest neighbors and frequent pattern mining techniques using the support of the occurrence of some items together [15]. However, these methods are categorized as instance-based algorithms that usually take much time during making predictions, which is not suitable for real-time use cases. Later on, other research works were proposed using more advanced techniques. For example, Markov chain models were proposed in [28], [35] to be utilized in sequence modeling. The problem of the state-space explosion in Markov models has been treated by using some items attributes to predict the characteristics of the next items to be recommended [61]. Additionally, classical matrix factorization techniques were combined with Markov chains in different variations that were applied in a wide range of domains [19], [31].

Over the last few years, different deep learning approaches have been used in the session-based recommendation. The main advantage of deep learning approaches is their ability to automatically extract features, allowing learning complex functions to be mapped from the input space to the output space without the need for much human intervention [12]. For example, Gated Recurrent Units (GRUs) were used in RNNs to predict the next item to be clicked by the user [10]. The model was trained using pairwise losses by comparing the target item score with the negative examples' maximal score. The likelihood of these negative samples is taken into account in proportion with the target item's maximal score. These loss functions showed good performance by correctly ranking the predicted items and overcoming the vanishing gradient problem in RNNs.

Furthermore, GRUs architecture was extended by using a modified version of originally negative sampling approach where the likelihood score of the next recommended item was calculated for a subset of items only as it would be impractical to do it for the whole list of items [34]. The new sampling method used additional negative samples shared by all the session sequences within the same mini-batch and used to update a small percentage of the weights rather than all of them for each mini-batch to make the training process faster. These samples were chosen based on items' popularity, giving more chances to include most of the high-scoring negative examples. This approach leads to a substantial improvement in model performance. The same architecture was also 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 significant improvement [33]. 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 for a higher level RNN for the next session of the same user [53]. A hybrid architecture of two RNNs was proposed for a personalized session-based recommendation that aims mainly in targeting session cold-start problem by learning from the user's recent sessions [56]. Convolutional neural networks (CNNs) were also used in the session-based recommendation. In particular, Tuan et al. used 3D-CNN with character level encoding to combine session clicks with items content textual descriptions to generate recommendations [62]. Similarly, a generative CNN was proposed in [67] by embedding past clicked items in a 2-dimensional matrix, and treated as images as input to the CNN. Recently, graph neural networks were used to capture complex transitions among items after modeling the session sequence events as graph-structured data even without adequate user behavior in a session [26]. Also, Wang et al. proposed a novel framework using two parallel memory encoders to make use of collaborative neighborhood session information in addition to the current session information followed by selective fusion of both encoders output [66].

After the reveal of the attention concept in neural networks that leads to great improvement in neural machine translation tasks [29], [63]. Recently, attention networks were widely adopted in a session-based recommendation [46], [44], [18]. In the work done by Li *et al.* [44], a hybrid encoder was used with attention to model user sequential behavior. This was a clear problem in long-term memory models like GRU4Rec [44]. Also, a short-term attention priority model was introduced such that attention weights are computed from the full session context and enhanced by current user interests represented by the last clicked item [46]. In addition, Sun et al. [20] adopted the current state-of-the-art BERT transformer network, used mainly in natural language processing domain neural session-based recommendation architectures in personalized session-based recommendation [59]. Most of the neural solutions in session-based recommendation generate a static representation for users' long-term interests. However, such representation might be a problem as its importance for predicting the next recommended item can be dynamic and related to the short-term preferences. Thus, a co-attention network was proposed to recognize the dynamic interaction between the user's long and short-term interests and generate a co-dependent representation of the users' interests [18].

2.2 Detailed explanation of the session-based recommendation and its importance

Sequence-aware recommendation is a general class of recommendation systems where the input of these systems is a chronologically ordered set of user actions linked with a set of different items. Various types of attributes can characterize users, items, and actions. The output of these systems is a ranking of different scores based on the likelihood that user preferences match these items [52].

Session-based recommendation systems (SBRS) are special kind of sequence-aware recommendation that are interested more in the short-term user intent represented by the very last events in a short period of interaction time between the user and a website defined by a session. Despite that e-commerce is the most important application for the session-based recommendation, many other applications exist for a session-based recommendation like music playlist recommendation, video recommendation, news recommendation [28], online course recommendation, etc. [38]. SBRS aims to predict either the unknown part (e.g., an item or a batch of items) of a session given the known part, or the future session (e.g., the next-basket) given the historical sessions via learning the intra- or inter-session dependencies usually primarily rely on the co-occurrence of interactions inside a session and they may be sequential or non-sequential [36]. In principle, an SBRS does not necessarily rely on the order information inside sessions, but for ordered sessions, the naturally existing sequential dependencies can be utilized for recommendations. For example, in comparison to a Sequential Recommender Systems(SRSs), which predicts the successive symbols given a sequence of historical ones by learning the sequential dependencies among them [49].

2.3 Deep Neural Networks

Deep neural networks are a subset of machine learning technologies that have gained tremendous interest in the past decade. DNN targets the problem of automatic learning of good representation for the input data. The smallest building block of a neural network is an abstraction of a biological neuron, which takes different inputs, adds them up with different weights, and passes the output to the other nodes. These neurons are grouped into layers based on the network architecture, and their weights are learned during the training process. Simple representations can be obtained from the first layers of the network, while the complexity of these representations increases as it goes through the deeper layers. A wide range of network types and thousands of architectures have been proposed like convolutional neural network that are suitable for computer vision tasks, and recurrent neural networks for time-series analysis [32].

Deep learning techniques have achieved outstanding results in a wide range of domains, including natural language processing, medical diagnosis, speech recognition, and computer vision. In practice, their ability to automatically extract features is the key benefit of deep learning over conventional machine learning techniques, allowing complex functions to be mapped from the input space to the output space without human intervention [12]. Some approaches to using deep neural networks in recommendation systems have recently been suggested [65], [43]. In particular, for modeling the sequence of user navigation actions in online services to be used in the next item recommendation, various deep learning models were used[10] [57]. These works demonstrated a competitive benefit in efficiency compared to methods used such as nearest neighbor algorithms, sequential pattern mining techniques and traditional Markov models [48].

2.3.1 Convolutional Networks

Convolutional neural networks (CNNs) are a specific architecture of neural networks that are extremely effective at processing data that has a grid-like topology of sequential filters for image processing. Each filter/kernel is considered as a 2-dimensional grid of pixels that slide horizontally and vertically over the input image spatially, yielding the sum of the element-wise multiplication of corresponding elements. Therefore, every operation leads to the output of a single-pixel.[14] Each kernel can output a feature map and all the feature maps are concatenated together; this is also known as a convolutional layer and it is the core component in a CNN. During the past decade, several advancements have occurred in CNN architectures which lead to a substantial improvement in image classification. Especially, CNNs have made many breakthroughs on large scale image datasets such as the ImageNet challenge [22], AlexNet [22], GoogleNet [23].

2.3.2 Recurrent Networks

Recurrent neural networks, also known as RNNs, are deep models of choice when dealing with sequential data [45]. They are a class of neural networks that allow previous outputs to be used as inputs while having hidden states. At each time step, the hidden state of the RNN is computed from the current input in the sequence and the hidden state from the last step. The hidden state is then used to predict the probability of the next items in the sequence. The recurrent feedback mechanism memorizes the influence of each past data sample in the hidden state of the RNN, hence overcoming the fundamental limitation of sequence modeling methods such as Markov Models [49]. RNNs have been used in a wide range of applications such as Speech recognition, music generation, sentiment classification, DNA sequence analysis, machine translation, video activity recognition, and much more.

2.3.3 Attention in Neural Networks

Attention models, or attention mechanisms, are input processing techniques for neural networks that allow the network to focus on specific aspects of a complex input, one at a time until the entire dataset is categorized. Attention is a layer of calculation that lets a model focus on the most important part of the sequence for each step [64]. Queries, values, and keys represent the encoders' and decoders' hidden states and are used to retrieve information inside the attention layer by calculating similarities between the decoder queries and the encoder key-value pairs. The attention mechanism uses encoded representations of both the input or the encoder hidden states and the outputs or the decoder hidden states, as shown in Figure 3. The keys and values are in pairs. Both of dimensions N, where N is the input sequence length and comes from the encoder hidden states.

Keys and values have their own respective matrices, but the matrices have the same shape and are often the same. At the same time, the queries come from the decoder's hidden states. For example, while translating English into German. We can represent the word embeddings in the English language as keys and values. The queries will then be the German equivalent. We can then calculate the dot product between the query and the key. Let us note that similar vectors have higher dot products and non-similar vectors will have lower dot products. The intuition here is that we want to identify the corresponding words in the queries that are similar to the keys. This would allow the model to look or focus on the right place when translating each word.

2.4 Automated Machine Learning

The past decade has seen many accomplishments in Machine Learning (ML) research and applications: especially, deep learning methods have enabled key advances in many application domains,

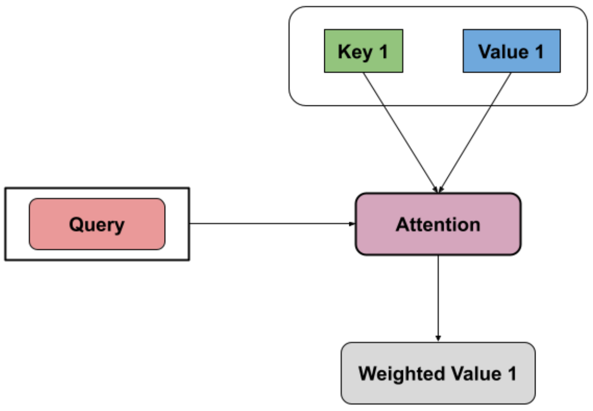


Figure 3. Attention mechanism representation.

such as computer vision, speech processing, and game playing. However, the performance of many machine learning methods is susceptible to a plethora of design decisions, which constitutes a considerable obstacle for new users. This is particularly true in the growing field of deep learning, where human engineers need to select the right neural architectures, training procedures, regularization methods, and hyperparameters of all of these components in order to make their networks to do what they are supposed to do with sufficient performance. This process has to be repeated for every application. Even experts are often left with exhausting trial and error episodes until they identify a good set of choices for a particular dataset.

2.4.1 What is AutoML? Why is it important?

Automated Machine learning provides methods and processes to make Machine Learning available for non-Machine Learning specialists to enhance machine learning performance and accelerate research on Machine Learning. Besides, manual ML design is a tedious task, slowing down new application of ML. Consequently, the field of automated machine learning [27, 21] has emerged to support researchers and developers in developing high-performance ML pipelines. Popular AutoML systems such as Auto-WEKA [16], hyperopt-sklearn [8], Auto-Keras [30] are immediately available and allow for using ML with fewer lines of code. Neural Network

Intelligence (NNI)[8]provides tools for tuning hyperparameters and managing training experiments.

While most AutoML frameworks initially focused on traditional machine learning algorithms, deep learning (DL) has become more popular over the last few years due to its robust performance and ability to automatically learn useful representations from data. However, choosing the correct hyperparameters for a task is of significant importance for DL [27]; the optimal choice of neural architecture also varies between tasks and datasets [3]. This has lead to the recent rise of neural architecture search (NAS) methods [9]. Since there are interaction effects between the best architecture and the best hyperparameter configuration for it, AutoML systems have jointly optimized both [4].

2.4.2 State-of-the-art AutoML and the tools

Hyperparameter optimization (HPO) is a method that helps solve the challenge of tuning hyperparameters of machine learning algorithms. Several approaches have been used, such as Bayesian optimization (BO), which provides a principled technique based on Bayes Theorem to direct search of a global optimization problem that is efficient and effective. Successive halving is a type of bandit-based algorithm that attempts to give the most budget to the most promising methods. It assumes all configurations could be stopped early and a validation score could be obtained.

Modern deep learning models are very sensitive to many hyperparameters, and due to the long training times of the state-of-the-art methods, vanilla Bayesian hyperparameter [25] optimization is typically computationally infeasible. Alternatively, bandit-based configuration evaluation approaches based on random search lack guidance and do not converge to the best configurations as quickly. Here, we introduce several hyperparameters optimization methods such as Bayesian optimization, Hyperband (HB), and the combined state-of-the-art BOHB (Bayesian optimization and Hyperband) used in our work which consistently outperforms both BO and HB on a wide range of problem types, including support vector machines, feed-forward neural networks, high-dimensional toy functions, Bayesian neural networks, deep reinforcement learning, and convolutional neural networks [25]. Furthermore, we briefly introduce Neural Network Intelligence (NNI) toolkit used in our work to tune hyperparameters of session-based recommendation models.

2.4.2.1 Bayesian optimization

Bayesian optimization (BO) is a state-of-the-art optimization algorithm for the global optimization of expensive BlackBox functions. It has been successfully applied to optimize hyper-parameters of neural networks for speech recognition [1], image classification [55], and neural language modeling [50] and by demonstrating wide applicability to different problem settings.

In each iteration i, BO uses a probabilistic model $p(f|D)$ to model the objective function f based on the already observed data points $D = \{(x_o, y_o), ..., (x_{i-1}, y_{i-1})\}$. BO uses an acquisition function $a : \mathcal{X} \to \mathcal{R}$ based on the current model $(f|D)$ that exchanges exploration and exploitation. Based on the model and the acquisition function, it iterates the following three step: (1) choose the point that maximizes the acquisition function $X_{new} = argmax_{x \in \S} a(x)$,

[8]https://nni.readthedocs.io/en/latest/index.html

(2) estimate the objective function $y_{new} = f(x_{new}) + \in$, and (3) augment the data $D \leftarrow D \cup (x_{new}, y_{new})$ and refit the model. A common acquisition function is the expected improvement (EI) over the currently best observed value $\alpha = min\{y_o, ..., y_n\}$:

$$a(x) = \int max(0, \alpha - f(x))dp(f|D).$$

2.4.2.2 Hyperband

Hyperband [2] is a bandit strategy that dynamically allocates resources to a set of random configurations vs. budget for each, and uses successive halving [37] as a subroutine to stop poorly performing configurations. Because of the hedging strategy which involves running some configurations only on the maximal budget, in the worst case, HyperBand takes at most a constant factor more time than random search on the maximal budget. In practice, due to its use of cheap low-fidelity evaluations, Hyperband has been shown to improve over vanilla random search and BlackBox BO for data subsets, feature subsets and iterative algorithms, such as stochastic gradient descent for deep neural networks

i	S = 4		S = 3		S = 2		S = 1		S = 0	
	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i	n_i	r_i
0	81	1	27	3	9	9	6	27	5	81
1	27	3	9	9	3	27	2	81		
2	9	9	3	27	1	81				
3	3	27	1	81						
4	1	81								

Figure 4. Example of resource allocation of Hyperband configurations. s means bucket, n means the number of configurations that are generated, the corresponding r means how many budgets these configurations run. i means round, for example, bucket 3 has 4 rounds, bucket 2 has 3 rounds.

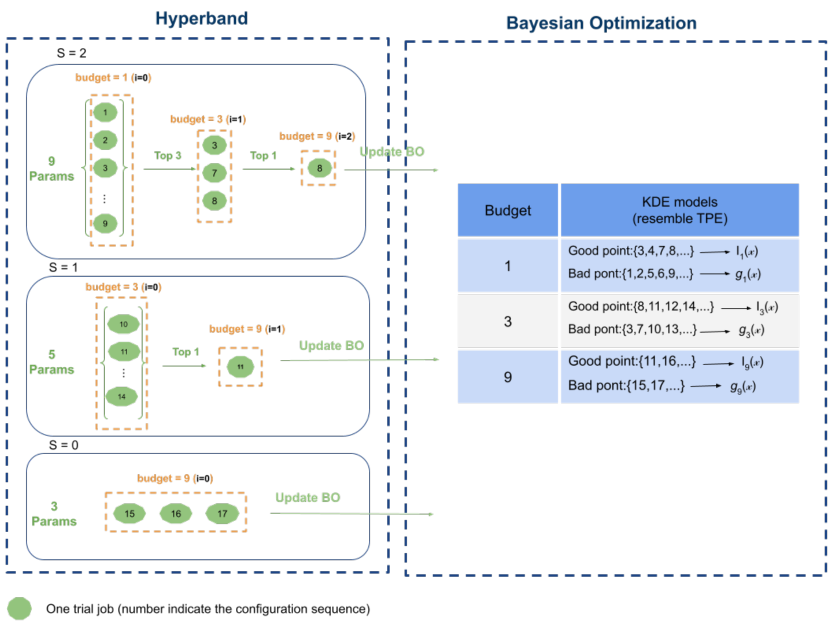
Hyperband requires two inputs (1) R, the maximum amount of resource that can be allocated to a single configuration, and (2) n, an input that regulates the proportion of configurations dropped in each round of successive halving as shown in Figure 4. The two inputs dictate how many different brackets are considered; specifically, $s_{max}+1$ different values for n are considered with $s_{max} = [log_n(R)]$. Hyperband starts with the most aggressive bracket $s = s_{max}$, which sets n to maximize exploration, subject to the constraint that at least one configuration is allocated R resources. Each subsequent bracket reduces n by a factor of approximately n until the last bracket, $s = 0$, in which each configuration is allotted R resources (this bracket simply performs classical random search). Hence, hyperband performs a geometric search in the average budget per configuration and removes the need to select n for a fixed budget at the cost of approximately $s_{max} + 1$ times more work than running successive halving for a single value of n. By doing so, hyperband is able to exploit situations in which adaptive allocation works well while protecting itself in situations where more conservative allocations are required.

2.4.2.3 Bayesian optimization Hyperband (BOHB)

Notwithstanding Hyperband's success for DNN, its convergence to the global optimum is limited by its dependence on the random selection of configurations, and with large budgets, its advantage over random search typically decreases. To overcome this weakness, the recent approach BOHB [24] combines Bayesian optimization and Hyperband to achieve the best of both algorithms: strong anytime performance (yield good configurations with a small budget, fast improvements in the beginning by using low fidelities in Hyperband), and strong final performance (good performance with a given larger budget by substituting hyperband's random search by Bayesian optimization). BOHB deals with problem domains ranging from a few to many dozen hyper-parameters. It also uses parallel resources effectively. The BO part of BOHB closely resembles the Tree Parzen Estimator (TPE) [13]. TPE is a method that sequentially constructs models to approximate the performance of hyper-parameters based on historical measurements, and then subsequently chooses new hyper-parameters to test based on this model. Even though it resembles to TPE however, BO part of the BOHB uses a single multidimensional kernel density estimators (KDE). It only fits a model on the highest fidelity for which at least $|A| + 1$ evaluations have been performed (the number of hyper-parameters, plus one). BOHB's first model is therefore fitted on the lowest fidelity, and overtime models trained on higher fidelities take over while still using the lower fidelities in successive halving. Empirically, BOHB was shown to outperform several state-of-the-art hyper-parameters optimization methods for tuning support vector machines, neural networks and reinforcement learning algorithms [25]. Figure 5 illustrates the workflow of BOHB

2.4.2.4 Neural network Intelligence

Neural network Intelligence (NNI) toolkit is used in this work to tune deep learning models of session based-recommendation. It provides the scalability as NNI as shown in Figure 6. Tuner receives search space and generates configurations. These configurations will be submitted to training platforms, such as the local machine, remote machines, or training clusters. Their performances are reported back to Tuner. Then, new configurations are generated and submitted. For every experiment, the user only needs to define a search space and update a few lines of



Budget	KDE models (resemble TPE)
1	Good point:{3,4,7,8,...} $\longrightarrow l_1(x)$ Bad pont:{1,2,5,6,9,...} $\longrightarrow g_1(x)$
3	Good point:{8,11,12,14,...} $\longrightarrow l_3(x)$ Bad pont:{3,7,10,13,...} $\longrightarrow g_3(x)$
9	Good point:{11,16,...} $\longrightarrow l_9(x)$ Bad pont:{15,17,...} $\longrightarrow g_9(x)$

Figure 5. Workflow of BOHB. The maxbudget=9, minbudget=1, eta = 3, others as default. In this case, smax=2, so it will continuously run the $\{s = 2, s = 1, s = 0, s = 2, s = 1, s = 0, ...\}$ cycle. In each stage of Successive Halving (the orange box), we will choose the top 1/eta configurations and run them again with more budget, repeating the Successive Halving stage until the end of this iteration. Simultaneously, we collect the configurations, budgets and metrics of each trial and use these to build a multidimensional KDE model with the key *"budget"*

code and then leverage NNI built-in Tuner/Assessor and training platforms to search the best hyper-parameters and/or neural architecture. Furthermore, NNI can entirely leverage various computation resources, such as remote machines, training platforms (e.g., Kubernetes). Hundreds of trials could run in parallel by depending on the capacity of the configured training platforms. Besides rich built-in algorithms, its adaptability allows users to customize various hyper-parameter tuning algorithms, neural architecture search algorithms, early stopping algorithms, etc. Moreover, NNI can connect to external environments to tune special applications/models on them.

Key Concepts

- *Experiment:* one job is described as finding out the best neural network architecture, finding out the best hyper-parameters of a model. It consists of trials and AutoML algorithms.
- *Search Space:* The possible region for tuning the model. For example, the value range of each hyper-parameter.
- *Configuration:* An instance from the search space where each hyper-parameter has a

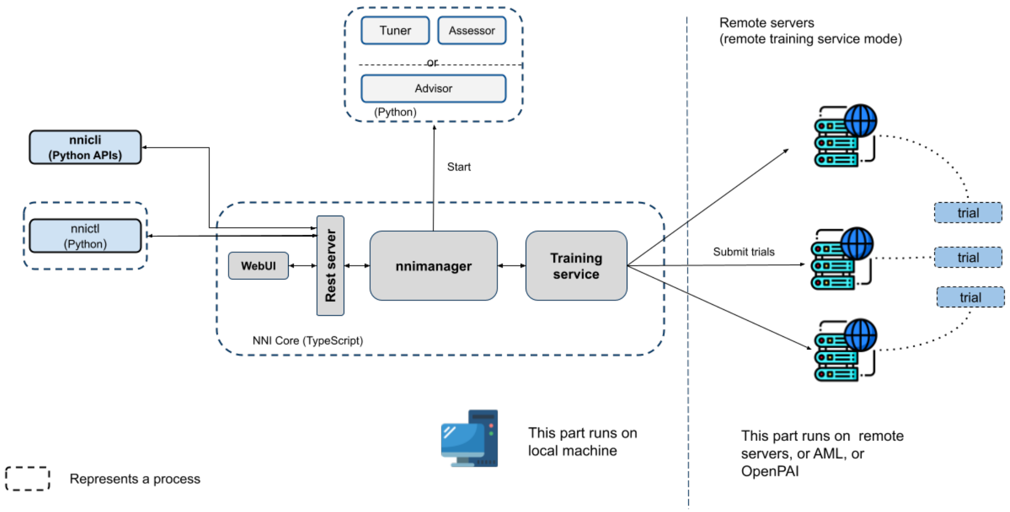


Figure 6. A high-level architecture of NNI.

specific value

- *Trial*: An individual attempt at applying a new configuration (e.g., a set of hyper-parameter values, a specific neural architecture, etc.). Trial code should be able to run with the given configuration.
- *Tuner:* An AutoML algorithm, which generates a new configuration for the next try. A new trial will run with this configuration.
- *Assessor:* Investigate a trial's intermediate results (e.g., periodically evaluated loss or accuracy on test dataset) to tell whether this trial can be early stopped or not
- *Training Platform:* where trials are executed. Depending on the experiment's configuration, it could be the local machine, or remote servers.

3 Related work

In this section, we give a detailed explanation of the previous literature work. Also, we explain different approaches used for session-based recommendations.

3.1 Traditional Recommendation Algorithm

Based on the study made by Ludewig *et al.*, we have selected a set of 3 baseline algorithms to be included in this work. The selection was made based on two different criteria. First, we selected at least one method from each family of algorithms of a good performance at the session-based recommendation in [48]. Second, we have chosen the method with the highest performance in general compared to all different methods within the same family. The chosen algorithms are session-based popular products (S-POP) as a simple heuristic algorithm, Vector session-based K-nearest neighbors (VSKNN) as the nearest neighbors approach, and session-based matrix factorization (SMF) was chosen as one of the factorization based methods.

3.1.1 Session-Based Popular Products

S-POP is considered as the most well-known baseline recommendation algorithm [6]. Despite its simplicity, it is widely used in recommendation systems [58], and provides a good choice of item recommendation based on the most frequent item in the current session of items viewed by the user. In summary, if an item is viewed by the user many times during the same session, this shows a clear sign of the user's interest in that particular item and recommending that the item to the user again is a reasonable choice. In some cases, the recommendation process in S-POP is limited to the top popular K items while the rest of the items are ignored. This ensures that the predicted items belong to the most popular items among all users generally instead of being specific to the current user.
Score of item I_n in session S_t is computed as follows:

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

3.1.2 Vector Multiplication Session-based K-Nearest Neighbors

Despite their simplicity, nearest neighbor algorithms showed good performance in session-based recommendation[40]. Additionally, it has many different variant schemes that can be applied according to domain type, like item-based nearest neighbors [7], which depends on predicting similar items to the last one viewed by the user. On the other hand, session-based nearest neighbors consider the whole session viewed items and try to find neighboring sessions with similar items to be used in predicting the next recommended items [15]. Ludewig *et al.* evaluated multiple variants of nearest neighbor algorithms. It has been shown that vector multiplication session-based K-nearest neighbors (VSKNN) has outperformed pattern mining and matrix factorization methods in most datasets. It even has a competitive performance to RNNs. VSKNN is considered

as one of the session-based nearest neighbors algorithms where recent items clicked by the user take higher weights than older items which in turn tries to give more emphasis on recent events made by the user.
Score of item I_n to be recommended next is computed as follows:

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

where $sim(I_n, S_t)$ can be set to the cosine similarity distance, and $W_t(S_t)$ is set to the reciprocal of the item position in the session to give higher value to recently clicked items such that the last element x_{L_t} takes value 1 and $x_{(L-1)_t}$ takes 2, etc [48].

3.1.3 Session-based Matrix Factorization

SMF is a matrix factorization-based approach designed for the task of session-based recommendation by [48], and inspired by factorized personalized Markov chains [31], [41] which were proposed for sequential recommendation tasks. In SMF, a hybrid approach combining between classical matrix factorization and factorized Markov chains. A special treatment for factorization methods is used in SMF by replacing the latent user vector with an embedding vector representing the current session. During making predictions, the score of a candidate item is computed as the weighted sum of the whole session preferences in addition to the sequential dynamics representing the probability of transition 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., where some tricks were adopted from GRU4Rec network like skip-rate and drop-out factor, and the same loss function. The algorithm showed a better performance than other factorization-based methods over multiple datasets [48].

3.2 Neural-Based Models

Many deep learning architectures were proposed in the research literature for session-based recommendation. These architectures are classified depending on their types like Tuan *et al.* [62], and Yuan et al. [67], where convolutional neural networks (CNNs) were used. In particular, Hidasi *et al.* [10] was the first work that use RNNs with GRUs in session-based recommendation. Recently, Wu et al. [57] exploited graph neural networks in session-based recommendation and Liu *et al.* [46], [44] proposed different attention mechanisms to enhance the performance of RNNs.

In this work, we limited our selection to include current state-of-art and well-cited architectures proposed in the range of last four years published in top tier venues, and have open-source implementation. Additionally, we refined our list to select model that can be used in making generalized (non-personal) predictions without the need of collecting personal data for each user for the reasons mentioned in section I. The final list of the chosen architectures includes an extended version of GRUs neural networks (GRU4Rec +) by Hidasi *et al.* [10], graph neural network proposed by Wu *et al.* [57], short-term attention priority network (STAMP) by Lieu et al. [46] as well as convolutional generative network for session-based recommendation (NextItNet)

[67], and collaborative neural network with parallel memory modules (CSRM) proposed by Wang et al. [66]

3.2.1 Gated recurrent neural networks for session-based recommendation

One of the early successful approaches of using RNNs in the recommendation domain was proposed by Hidasi et al. [10], such that a RNN with GRUs was used for the session-based recommendation (GRU4Rec). In GRU4Rec model, a novel training mechanism called session-parallel mini-batches was proposed as shown in Figure 7. 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 trained with the session beginning and used to predict the subsequent events. 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 the space of all items 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 ranking of items to be predicted. Moreover, 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 score of a positive item with several sampled negative ones in the loss value, and TOP1 is the regularized approximation of the relative rank of the relevant item loss. Hidasi et al. extended their work by modifying the two loss functions introduced previously by solving the issues of vanishing gradient faced by TOP1 and BPR when the negative samples have 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 a clear superiority over the older version network. Thus, we have included the modified GRU4Rec network, which is denoted by GRU4Rec+ in the rest of this work, in our evaluation study using the extended version of losses [34].

3.2.2 Simple generative convolution network

NextItNet was proposed to use CNNs in the session-based recommendation. The session made by a user is converted into a 2-dimensional latent matrix and fed into a convolutional neural network like images [67]. NextItNet is considered as an extension over the recent convolutional sequence embedding recommendation model (Caser) by Tang et al., who were the first to use CNNs in the domain of session-based recommendation [60]. However, NextItNet addresses 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. On the other hand, using large filters with the width of filter similar to the image width of an item inside the full session sequence

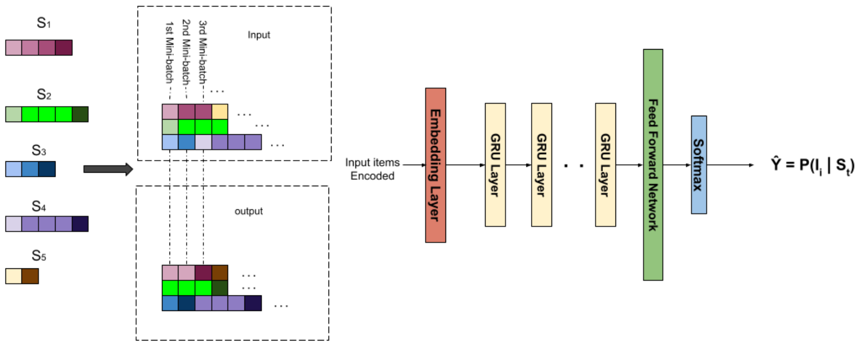


Figure 7. GRU4Rec model architecture adapted from Hidasi *et al* [10]

followed by max-pooling layers 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, the huge number of inefficient convolutional filters are replaced with a series of 1-dimensional dilated convolution layers that are responsible for increasing the receptive field and deal with different session lengths instead of standard 2D convolution layers. Thus, the max-pooling layers are omitted as they cannot distinguish the essential features in the map if they occur once or multiple times with ignoring the position of these features. Additionally, NextItNet makes use of residual blocks effectively in recommendation systems, which can ease the optimization for much deeper networks than the shallow convolutional network in Caser which cannot model complex relations between items in a user session.

3.2.3 Short-term attention/memory priority model

STAMP is one of the approaches that replaces complex recurrent computations in RNNs with self-attention layers [46]. 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 has the advantage of capturing the user interest drifts, especially during long sessions and outperforms other approaches like GRU4Rec [10] that uses long-term memory but still not efficient in capturing user drifts. The model architecture in Figure 8 shows that the input is two embedding vectors (E_L, E_{S_t}) denoting 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 interest, and the general user's interest through the entire session clicked items respectively. E_{S_t} vector is computed by averaging the items embedding vectors throughout the whole session memory $(x_1, x_2, ..., x_L)$. An attention layer is used to produce a real-valued vector Ea where this layer is responsible for computing the attention weights corresponding to each item in the current session such that we avoid treating each item in the session as equally important and paying more attention to only related items, which improves capturing of user interest drift. E_a and E_L flow into two multi-layer perceptron networks 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 each of the available items to be clicked next by the user and to be used in the recommendation process.

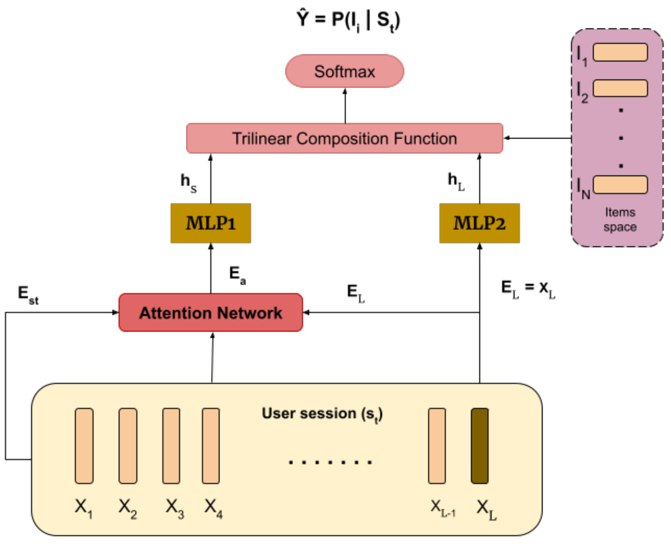


Figure 8. STAMP model architecture adapted from Liu *et al*[18]

3.2.4 Collaborative session-based recommendation machine

A hybrid framework applying collaborative neighborhood information to the session-based recommendation was proposed by Wang *et al.* [66] who hypothesized that neighborhood sessions to the current session, even made by different users, can contain useful information in improving the recommendation system predictions. The architecture implementation, shown in Figure 9, includes two main encoders. First, the inner memory encoder models the user behavior during his current session using an RNN with an attention mechanism that is fed with hidden state of the network from previous layer h_{t-1}, and current session St items. This encoder outputs two concatenated vector embeddings C^{Inner} of the current session behavior representing the whole session items and key items clicked in the session, respectively. Second, the outer memory encoder looks for the neighborhood sessions that contain similar patterns to the current session out of a subset of the recently stored sessions $(S_{t-1}, S_{t-2}, ...)$ which are used in enhancing 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 on the current session. The final current session representation C_t is formed by a selective fusion between both encoders output, and finally, the output scores are predicted for all items using a bi-linear decoding scheme between the item I_i embedding and the current session final representation vector C_t followed by a softmax layer. 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 representing item embedding vector, which can enhance the recommendations.

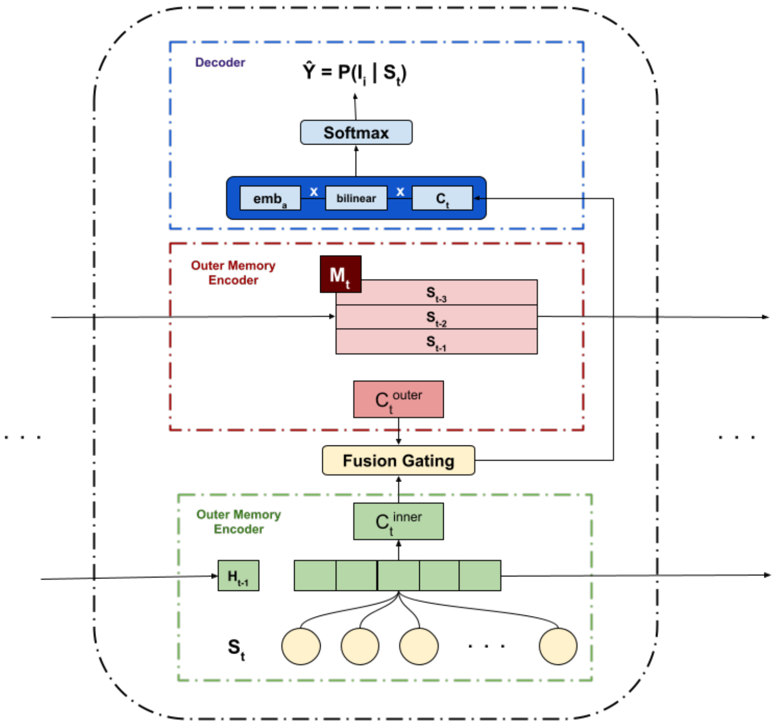


Figure 9. One unit of the CSRM architecture adapted from Wang *et al* [66]

4 Methodology

Next, we aim to explain the steps of our proposed automated machine learning framework, AutoiCV4Rec, for recommending SB algorithms based on dataset characteristics and tuning its hyper-parameters.

4.1 Experiments Description

Our study was divided into two sections; the first section aimed to evaluate and benchmark the models from their original paper. This study was divided into eight different experiments repeated for each model on all datasets. In the second section, the best neural-based models were selected and evaluated by tuning their hyper-parameters on a small portion of all datasets from the first study. The target of these experiments is to answer the following set of research questions (RQs):

- RQ1: Different Training Session Lengths: we aim to evaluate which models can perform well even with shorts sessions in length during training the models and which ones can make use of long training sessions better to specify the user's interest accurately. To answer this RQ, we divided each training dataset into three different splits according to their length. We keep only sessions of length < 5 in the first split, $\geq 5\& < 10$ in the second split, and ≥ 10 for the third one. All models are trained on each subset of training sessions and evaluated on the same validation set.

- RQ2: Different Validation Session Lengths: In this experiment, the models' performance is measured with different session lengths during inference. The main target of this experiment is to observe the model performance during the start of the session and after having an adequate number of interactions from the user. This could help in figuring out which models can't perform well at long sessions that usually include user drifts in preferences. On the contrary to the previous RQ, we fixed the same training dataset and divided the validation 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 validation set split.

- RQ3: Prediction of items with different popularity in the training set: instead of evaluating the performance against session length, we want to evaluate which models can learn well from less frequent items in the training set, and predict them accurately in the validation set. The goal of this experiment is to measure how models are biased towards more popular items. In this experiment, we divided the validation sets of each dataset to keep only items whose frequencies exceed a specific threshold in the training set. The frequency threshold used for different splits were (50, 100,200,300, > 300) for RECSYS/TMALL, and (10, 30, 60, 100, > 100) for CIKMCUP/ROCKET. All models are trained on the same training set and evaluated on each validation set split.

- RQ4: Effect of Data Recency: in this experiment, we divided our training sets into almost equal three data portions, the first represents the most recent collected data sessions, the second represents the eldest sessions, and the last one is a mix between newer half of portion 1 and the older half of portion 2. Then, all models are evaluated on the same validation set. We aim in this experiment to show empirically whether it is important to have a dynamic times series modeling taken into account during fitting the different models or seasonal changes can't make a significant drift in user preferences. For example, in fashion, e-commerce, users tend to look for light clothes during summer, making it unlikely to learn from data collected in the winter, where users usually have more preference for heavy clothes. Additionally, it is essential to determine how data recency could lead to data leakage problems resulting in deceiving model accuracy during training.

- RQ5: Effect of Training Data Size: we aim by the experiment to observe the models' performance on different dataset sizes. Answering this RQ can help more in understanding the suitable dataset size corresponding to the number of available items in a dataset such that the model performance is not affected heavily. Consequently, saving more computational resources without impairment in the model's performance. We have selected randomly different splits of the original training datasets such that these splits sizes are equal to 1/P of the original training set size where P (2, 8, 16, 64, 256)
- RQ6: Effect of Training Data Time Span: moreover, to check whether results obtained from RQ4, and RQ5 can be generalized or not, we run a similar experiment to RQ5; however, instead of selecting random portions of trainings sets, we divided them according to the period of time taken in data collection. For example, we used only the most recent data collected during the last m days before the period used for collecting the validation set instances to train the model. In this experiment, we used m (2, 7, 14, 30), and we aim to know what is the time span required to train different models and achieve the best performance according to the different dataset properties like 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 a fixed dataset splits? What are the coverage and popularity of the items of each model on a fixed dataset splits? Given the predictions made by each model, computing the item predictions coverage out of the total number of unique items, and their popularity can be a good indication if models tend to predict the most frequent items only or they can cover the space of items to a good extent in their predictions. We used full training and validation sets of CIKMCUP and the most recent sessions collected in 2 days for RECSYS in evaluating the coverage and popularity of the items.
- RQ8: Computational Resources: what are the required time and memory resources for each model during both training and prediction phases? In this experiment, we report 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. Besides, we aim to find the suitability of different models to be used practically in making real-time predictions.
- RQ9: Hyper-parameter optimization: we aim by the experiment to observe the neural-based models' performance after tuning their hyper-parameters using the most promising

HPO algorithm on single dataset size. Answering this RQ can help more in understanding the potential of automated machine learning frameworks in replacing the random selection of hyper-parameters. We have selected randomly different splits of the original training datasets such that the size of the split is equal to I/P of the original training set size where P (64).

In the first section of our study, for all the experiments, we filter out items from the validation split that do not exist in the training splits. The properties of all train/validation data splits used in each experiment are summarized in Table 3. 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 run a 20 iterations random search for all models on each dataset to tune the different hyper-parameters suggested to be tuned by their authors to be used in each neural network model. However, we kept the rest of the network's hyper-parameters as default, as mentioned in their corresponding papers. We selected the hyper-parameter settings achieving the highest HR@20 for each dataset. The list of used tuned hyper-parameters for each model can be found in Section 6.

The first section of the study was carried out in part in the high performance computing center of the University of Tartu[9] In case that graphical processing unit (GPU) could be used for neural network models, we used NVIDIA Tesla P100 GPU. Memory size was limited to 20GB RAM, Intel(R) Xeon(R) CPU E52660 v2 @ 2.20GHz processors with up to 30 cores were allocated from the computing center to run the models that don't support GPU. During reporting/validation time and memory consumption in our results, we haven't used any GPUs to make the comparison fair among all models. However, indeed all neural-based models support the usage of GPUs which is a highly important advantage over other algorithms. Source codes used in this study in addition to results logs can be fount at[10]

4.2 Datasets

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

1. YOOCHOOSE: The first dataset is collected by YOOCHOOSE[11] incorporation and published in RecSys Challenge 2015[12]. 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 was collected during $\approx$ 6 months in the year of 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 items in the dataset. Following the previous literature, we used the last day sessions as a validation set and the rest sessions as a training set. This dataset is referred to as RECSYS [10], [34].

[9]https://hpc.ut.ee

[10]https://github.com/mmaher22/iCV-SBR

[11]https://www.yoochoose.com/

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

2. DIGINETICA: Diginetica dataset was used in CIKM Cup 2016[13] for the personalized e-commerce search challenge. The dataset was provided by DIGINETICA[14] corporation containing anonymized search and browsing logs, product data, and anonymized transactions collected for 5 months from E-commerce websites. Only the transaction data was used in our experiments. Similar to RecSys, we used the last day sessions as a validation set and the rest sessions as the training set. We use the name CIKMCUP to refer to this dataset in the rest of the thesis.

3. TMALL: TMall is a large dataset consists of interaction logs from the e-commerce TMall website[15] . The data was collected for 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 significant longer sessions than other datasets. Due to constraints in 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 (1st of November) as the validation set. This dataset is referred to as TMALL.

4. RETAIL ROCKET: the Retail-Rocket dataset was collected and published by retail-rocket e-commerce personalization company[16] aiming to motivate researches in the field of recommender systems. The dataset includes user behavioral data from a real-world e-commerce website throughout ≈ 4.5 months like views, add to carts, and transactions in addition to items identifiers and their properties in a hashed format. Only the views, and add to carts events where used in our experiments while transactions were discarded. This dataset, in addition to CIKMCUP, are characterized by a low number of clicks compared to the number of existing unique items in addition to fewer sessions that both TMALL and RECSYS. We used the last two days as the validation set while the rest of the data as the training set. In the rest of this paper, we refer to this dataset as ROCKET.

	RECSYS	CIKMCUP	TMALL	ROCKET
Number of Items	37.48K	122.53K	618.77K	134.71
Number of Sessions	7.98M	310.33K	1.58M	367.59K
Number of Clicks	27.68M	1.16M	10.83M	1.06M
Timespan in Days	175	152	62	138
Average Item Frequency	738.5	9.5	17.5	7.9
Average Session Length	3.74	3.75	6.86	2.88

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

[13]https://cikm2016.cs.iupui.edu/cikm-cup/
[14]http://diginetica.com/
[15]https://www.tmall.com/
[16]https://retailrocket.net/

We filtered out sessions of length one during the preprocessing of all datasets as they do not include enough information for evaluation. Additionally, we filter out the clicked items in the validation set that were not found in the training set in all our different experiments. Also, multiple consecutive clicks on the same item in one session are replaced by a single click on that item as it does not make sense to recommend the same item that is currently clicked viewed by the user. Final statistics of the datasets are summarized in Table 3

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 O otherwise. The formula of HR@K for a dataset D is shown in Equation 1:

$$HR@K = \frac{1}{|S_D|} \sum_{i}^{|S_D|} 1_{IN}(I_{target}, argmax_K(\hat{Y_i}))$$

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 [10]. Equation 2 explains the computation of MRR@K

$$MRR@K = \frac{1}{|S_D|} \sum_{i}^{|S_D|} \frac{1}{r_{IN}(I_{target}, argmax_K(\hat{Y_i}))}$$

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 (Eq. 3) is the ratio of the number of unique items predicted by the model to the total number of items in the training set, where K is the number of top predictions to be considered from the model for each session.

$$COV@K = \frac{|\mathcal{U}(\{argmax_K(\hat{Y_i}) : i \in \{1, 2, ..., |S_{TE}|\}\})|}{|\mathcal{U}(S_{Tr})|}$$

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 user's preferences. The item popularity is the ratio between the average of the frequency of predicted items to the frequency of the most popular item in the training set, where K is the number of top predictions to be considered from the model for each session.

$$POP@K = \frac{\frac{1}{|S_{TE}|} \sum_{i}^{S_{TE}} \sum_{I \in argmax_K(\hat{Y_i})} \mathcal{F}(\mathcal{I})}{max(\mathcal{F}(\mathcal{S}_{\mathcal{TR}}))}$$

5 Experiment results

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

5.1 RQ1: Different training session lengths

In our plots and diagrams, the results for HR and MRR are reported for each model on all datasets. Contrary to some previous work like Ludewig et al., Who used predictions cut-off of 20 [48], we have chosen to set the number of predictions cut-off to five as it is more reasonable to recommend around five items to the user. At the same time, twenty seems to be a considerable number to be used in real-life e-commerce scenarios. However, complete results for different prediction cut-offs thresholds (1,3,5,10,20) can be found in Section 6 for long, intermediate, and short sessions, respectively. As shown in figure 10, most of the neural models out-perform

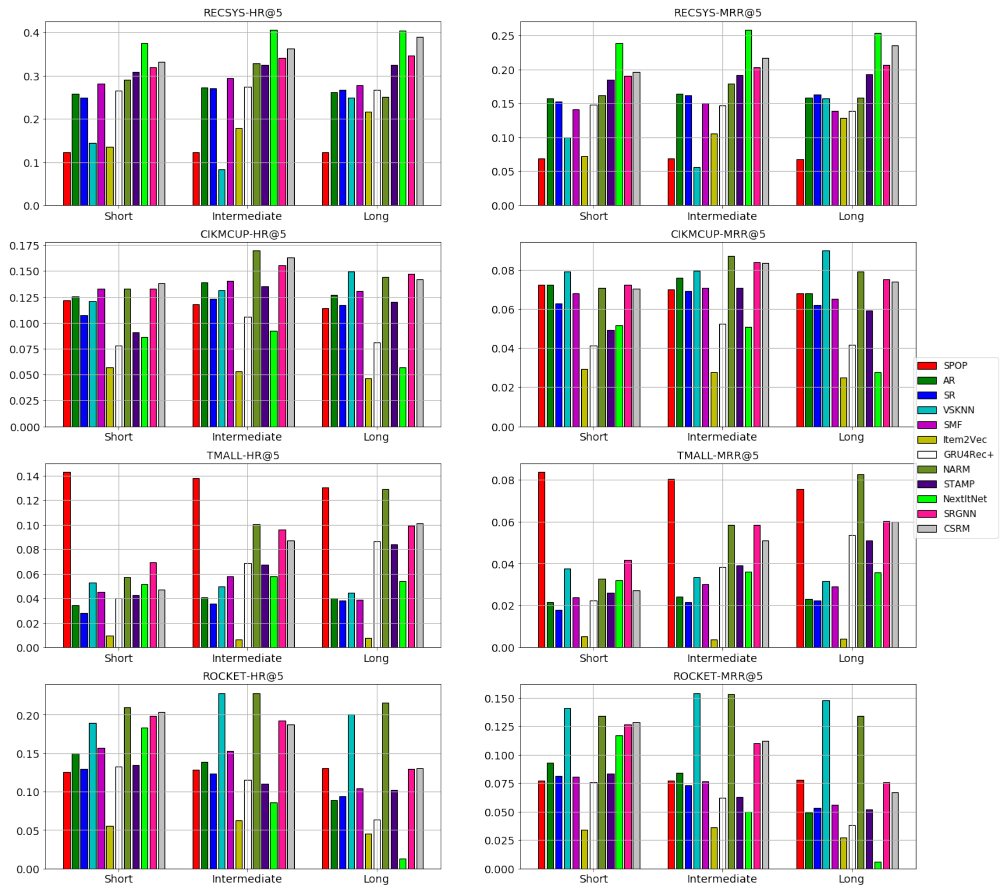


Figure 10. Effect of using different training session length on algorithms performance

the non-neural baseline models significantly except for S-POP, which is the best model in the TMALL dataset due to the nature of the dataset where items are repeated frequently within sessions. NextItNet achieves the highest performance in the RECSYS dataset only as this dataset is characterized by high average item frequency, which is suitable more with convolutional networks that require a large amount of data to model different items correctly. Other neural models have consistently high performance over all datasets. NARM, however, there is a small decrease in the performance of CSRM, SRGNN, and STAMP when training using short session length, which is clear in RECSYS, and TMALL, which have mainly intermediate to long sessions in corresponding validation sets. On the other hand, GRU4Rec+ has the highest performance when training using intermediate and short session length. In contrast, this performance degrades on long sessions due to the effect of vanishing gradient while using GRUs in these networks, which is evident in TMALL, and ROCKET datasets. Although NARM is also using the same network as GRU4Rec+, it does not suffer similarly from the vanishing gradient effect thanks to the attention layers used.

Regarding baseline models like S-POP, AR, and SR, there is a significant and consistent change in their performance while changing the training session's length, especially in the RECSYS dataset. However, it is clear that these models have comparable performance to neural models in datasets where the average item frequency is minor and not enough for learning good item representations from the data like in CIKMCUP and short sessions of ROCKET and TMALL datasets

5.2 RQ2: Different validation session lengths

While training using different session lengths gives good insights into the performance of models, this performance is still highly correlated with the length of the validation sessions. For example, suppose the model is trained using short sessions while most of the validation set sessions are long. In that case, it will not make accurate predictions. The full results of these experiments are tabulated in Section 6.

Figure 11 shows the performance of different models using the same training set for all of them while choosing a subset of sessions as a validation set according to their length. S-POP has an increasing performance while the session goes longer as it reflects a higher probability for the user to click on a previous item that he liked again. SR, and VSKNN performance degrade consistently and by a large degree for all the datasets as the session length increases. This is due to the small window of interest that both algorithms look at while computing the frequent patterns of items, which means that they will not make use out 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 for long sessions. Moreover, 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 apparent in Item2Vec, GRU4Rec+ among all datasets. Simultaneously, it is less observable for CSRM, STAMP, and NARM, which can be due to the memory and attention mechanisms applied in these models.

Overall, it seems that achieving good performance in long sessions is still a challenging

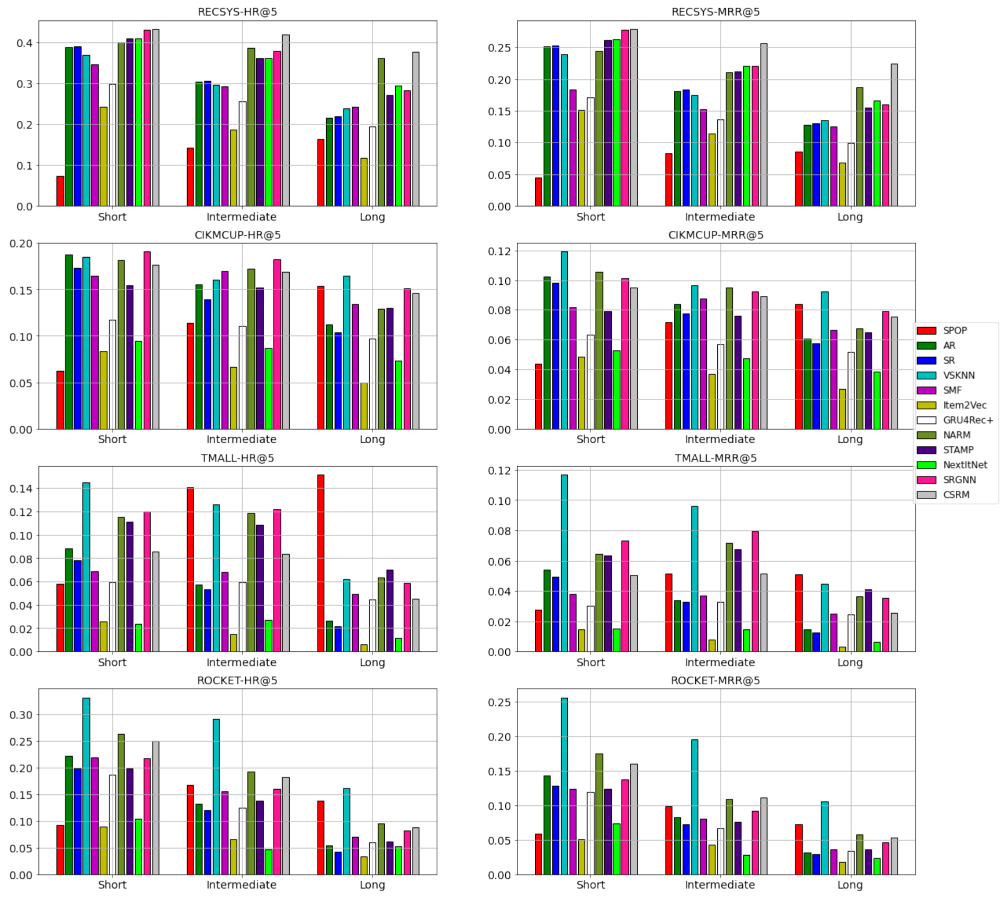


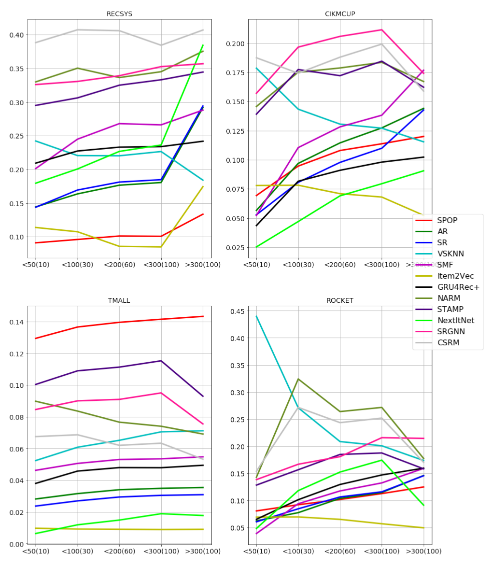
Figure 11. Models Performance on different validation session lengths

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

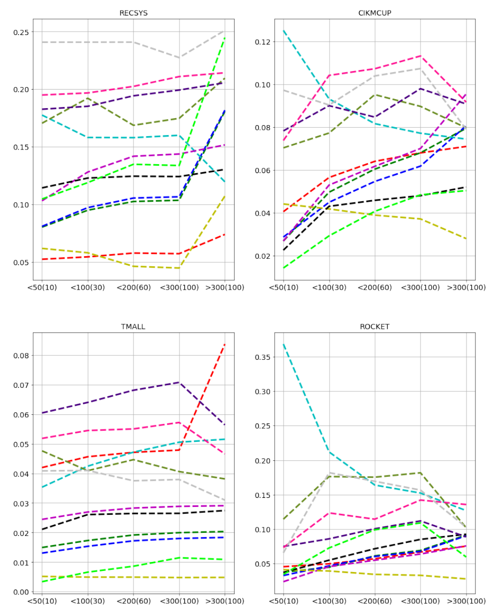
5.3 RQ3: Prediction of items with different abundance in the training set

Figure 12 shows the performance of different models when predicting an item with below a specific threshold of occurrences in the training set. This experiment shows how the model performance is affected by the presence of items many times during fitting. For this experiment, we used frequency threshold of (< 50, < 100, < 200, < 300) for RECSYS, and TMALL datasets which have higher average items frequency, and (< 10, < 30, < 60, < 100) for CIKM-CUP, and ROCKET datasets. In the tables in Section 6, we denote these different frequency thresholds (very low, low, medium, high, all).

SR, SMF, GRU4Rec+ performance is constantly improved significantly by increasing the frequency threshold among all datasets. To a less extent, NextItNet, and SRGNN has a slight gain in performance by increasing frequency threshold where this gain is stopped at very high frequencies like TMALL, 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 their occurrences in the training set to be able to model these items accurately, like AR and SR. On the contrary, some other models do not need a high frequency of occurrences, and it is enough to be represented only a few times in the training set like NARM, STAMP, and CSRM, which are all having various attention mechanisms.



(a) HR@5



(b) MRR@5

Figure 12. Performance of different models using different items frequency values in training set (< 50, < 100, < 200, < 300) for RECSYS, and TMALL datasets which have higher average items frequency, and (< 10, < 30, < 60, < 100) for CIKM-CUP, and ROCKET

5.4 RQ4: Effect of data recency

Using session-based recommendation models in e-commerce always requires being up-to-date with enough recent data to model the current users' trends. In this experiment, we tested our hypothesis using the most recent five days, the eldest five days, and a mix between half of recent and half of the old chosen splits[17]. The test set was fixed for all these different training splits. In Figure 13, 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 significant difference between models' performance on recent and mixed splits, especially in RECSYS and ROCKET datasets, there is still a slight difference in favor of recent splits for CSRM, SRGNN, and NARM in CIKMCUP, and TMALL datasets. Surprisingly, VSKNN is the only model that has higher performance on old splits in three out of the four datasets; however, this can be interpreted as the algorithm only cares about neighboring sessions of exactly similar items as those clicked by users, which indeed results in better recommendations if matching sessions were found.

These results confirm that there is a good use case to account for time-series dynamic modeling in the session-based recommendation to model general trends in users' preferences. Besides, in case the collected data is much old, there is a high chance that the nearest neighbor algorithms outperform other models.

5.5 RQ5: Effect of training data size

In this experiment, we aim to know the suitable training data sizes corresponding to the different datasets with various items. Additionally, investigating how the evaluated models perform 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 $(\frac{1}{8}, \frac{1}{16}, \frac{1}{64}, \frac{1}{256})$ split as they are larger ones. We refer to these splits as (large, medium, small, very small) respectively.

Figure 14 shows a heat map for HR@5 and MRR5 of different models while increasing the training size. S-POP and VSKNN are the only algorithms that do not get benefit out of larger data sizes. It can be easily observed that VSKNN achieved its highest performance along with the four datasets when using 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+ consistently get better when increasing training data sizes along all the datasets, while NARM, STAMP, and CSRM are improved a little bit less than the former models. Although there is a slight improvement in AR, SR, and SMF performance in RECSYS datasets, this improvement is not clear enough in other datasets to generalize this observation.

[17]In ROCKET, we used 10 days instead of 5 as the dataset is smaller than the rest

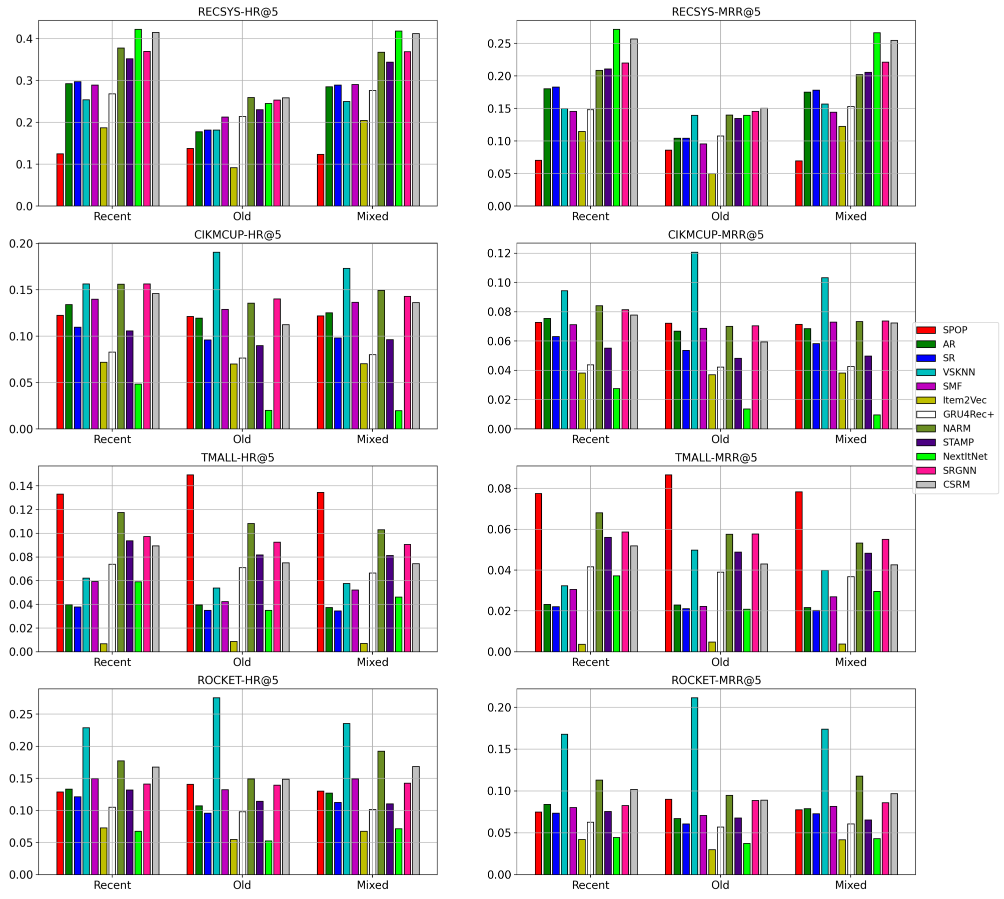


Figure 13. Effect of data recency on models performance for different datasets.

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 some information about the length of time required to collect training data sessions. Similar to RQ5, Figure 15 illustrates a heat-map of evaluated HR@5 and MRR@5 when training using splits of most recent x days from the whole training set in each dataset where x = (2, 7, 14, 30). similar to what is shown in Figure 14, 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 when increasing the dataset time-span; however, in RECSYS, the model improvement is almost ceased as this dataset has a small number of items, and a number of sessions in a 2-days time-span are pretty enough to model different items to be used in drawing accurate session representations.

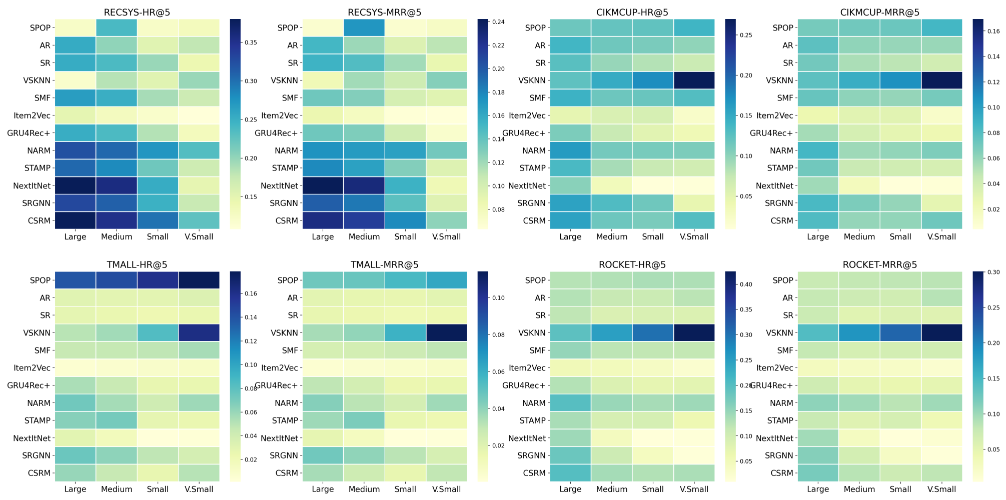


Figure 14. Models Performance on different training set sizes.

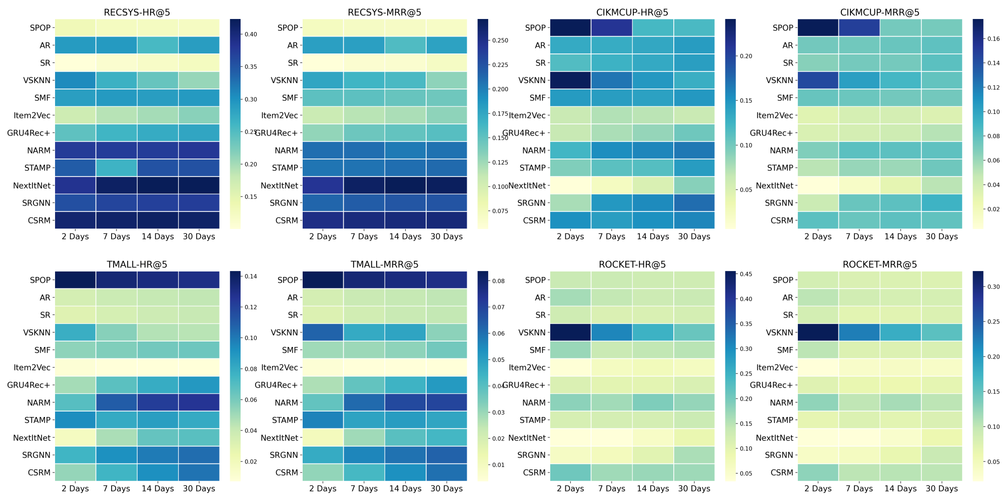


Figure 15. Models Performance on different training set time spans.

5.7 RQ7: Items coverage and popularity

Item coverage and popularity are good indications of how models tend to cover different items found and learned from the training set to be used in making recommendations. A high coverage value shows that the model is not biased towards the recommendation of a small set of items like popular or frequent items in the training set, which is shown by the second metric of average predictions popularity normalized to the most popular item in the training set. Figure 16 shows the natural logarithm of items coverage (COV@5) and popularity (POP@5) using the same training and validation sets. As expected, S-POP has the lowest coverage and highest popularity since it predicts only the most frequent items. On the other hand. Item2Vec has the highest coverage and lowest popularity; however, it is worth mentioning that this model has the lowest performance regarding the accuracy metrics.

Regarding baselines, AR, SR, and VSKNN are having quite similar coverage and popularity except for CIKMCUP, where VSKNN has higher item coverage. Then, SMF has relatively smaller both item coverage and popularity. GRU4Rec+ has the highest coverage among its predictions regarding the neural-based models, followed by NARM, STAMP, SRGNN, and CSRM in that order. However, the differences in item coverage of these models are not significant. Finally, NextItNet has the lowest coverage with relative popularity to NARM and CSRM, suggesting that this model is more likely to be over-fitted to a small subset of items and needs special regularization enhancements to be applied to the model. SRGNN and STAMP have relatively smaller average popularity across their predictions than other neural models consistently along with all the datasets, suggesting that using these models could be preferable if they have similar accuracy performance measurements with other models as they cover more unpopular items in their recommendations.

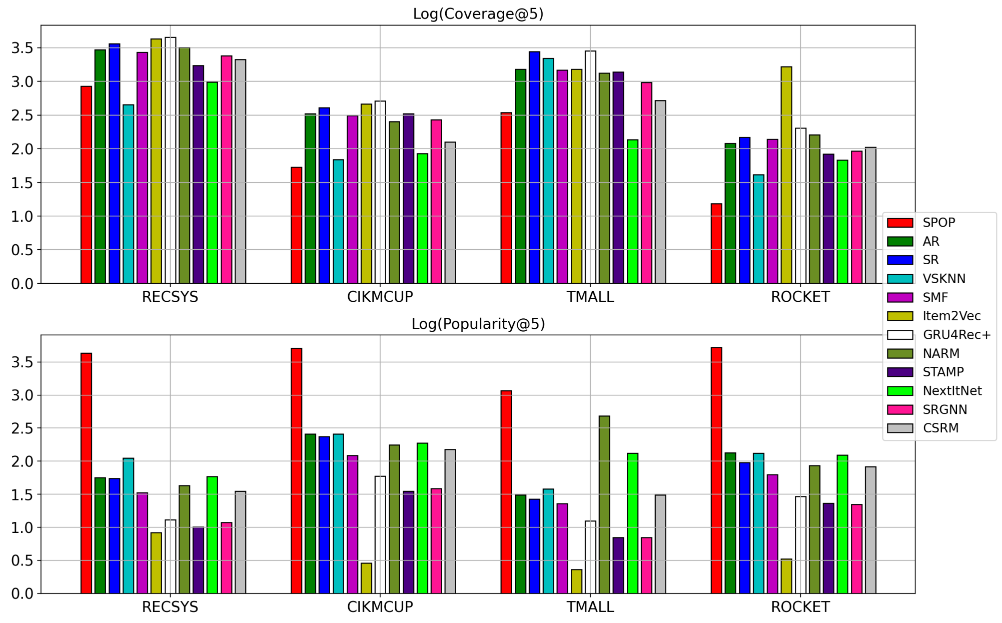


Figure 16. Items Coverage and Popularity of Models Predictions

5.8 RQ8: Computational resources

It is pretty essential to have a short validation time to make predictions quickly as it is required to provide the user with recommendations in a few seconds after making a specific action. At the same time, training computational complexity is essential in terms of the scalability of the model and the ability to train it easily every short period of time. Figure 17 summarizes the computational complexity of different models during both training and validation phases in RECSYS and CIKMCUP datasets. S-POP, AR, SR, and VSKNN are considered instance-based algorithms. The learning process occurs based on the validation session that we want to make predictions for it. Consequently, for these models, the computational training resources are almost neglected. On the other hand, they have significantly large validation time while making predictions, which means they are not suitable for making real-time predictions. Item2Vec and SMF are having both quite long training and validation time. Regarding neural models, all of them have significant large consumption training time and memory resources; however, they are still characterized by the adequate time when making predictions in the validation phase. STAMP has the lowest both training and validation time consumption, followed by GRU4Rec+, CSRM, then NARM, SRGNN, and NextItNet in an ascending order. However, NextItNet has a smaller validation time than NARM, SRGNN, and CSRM. At the same time, it is the highest memory consumer during the training and validation phases.

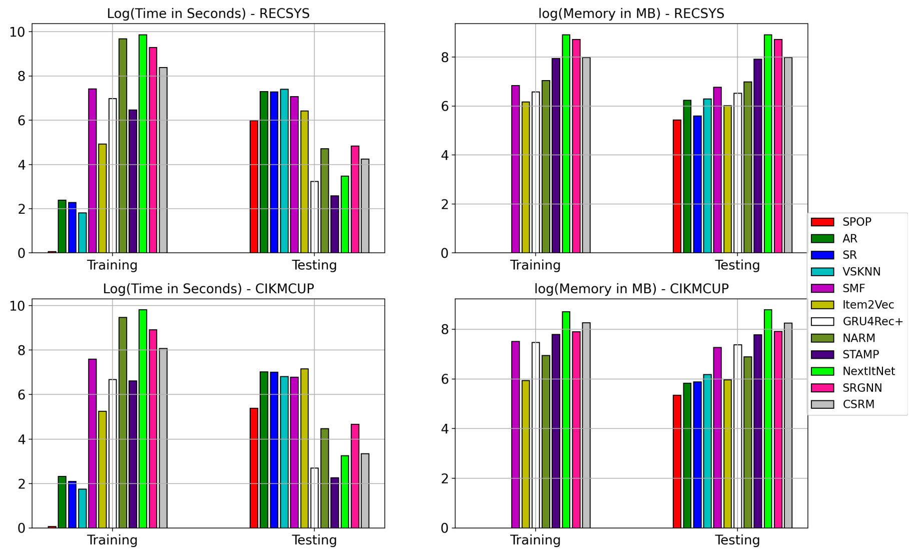


Figure 17. Models time and memory consumption on RECSYS and CIKMCUP datasets during training and validation.

5.9 RQ9: Hyper-parameter optimization

In this section, we tuned a set of neural-based model' hyper-parameters using BOHB as a tuner. The used parameters are *quniform* and *uniform*. The *uniform* keyword provides values uniformly distributed, which refers to a type of probability distribution in which all outcomes are equally likely. The sampling strategies are defined in the *Search Space*, which then generates a set of configurations where each hyper-parameter has a specific value. These values are then sent to the model to be trained. The process is repeated until all the trials are finished. The sampling strategies supported by the tuner were:

- $\{"type" : "quniform", "value" : [low, high, q]\}$: Type "quniform" will take three values [low, high, q] where [low, high] defines a range and "q" defines the interval. It will be sampled so that the first sampled value is "low" and each of the following values in "interval" larger than the value in front of it. This is suitable for a discrete value with respect to which the objective is still somewhat "smooth" but should be bounded above and below.[18].
- $\{"type" : "uniform", "value" : [low, high]\}$: the variable value is uniformly sampled between low and high. When optimizing, this variable is constrained to a two-sided interval[19].

5.9.1 NARM

5.9.1.1 Experiment setup

This experiment was carried on a computer with an NVIDIA GPU Corporation Quadro M4000 with a memory size of 8GB GDDR5. Three parameters were added to the search space for the NNI to tune: hidden units (type: quniform, value range [50, 200, 10]), batch size (type: quniform, value range [64, 1024, 10]) and the learning rate (type: uniform, value range: [0.0001, 0.1]).

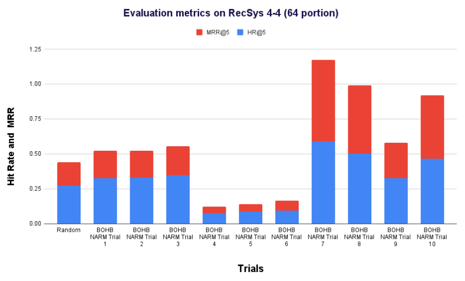
5.9.1.2 Results and discussion

Figure 18 shows the performance comparison of NARM using the BOHB tuner and random search. On the RECSYS dataset, most of the configurations generated by the tuner outperform the random search. The best architectural configuration generated by the tuner had more hidden units than the one proposed in the random search, suggesting that increasing the hidden units within a layer with regularization techniques increased the accuracy of the prediction. This performance is obvious on the $\frac{1}{16}$ portions of the RECSYS dataset where the NARM model reached its best performance, with the Hit Rate reaching 1, which means that with those configurations, the model was able to match the true item clicked by the user with any of the lists of predictions made by the model, which suggest that the target item was present among the top five predictions. However, in RECSYS $\frac{1}{16}$ portions, only three trials were completed. With the large size of the dataset, the

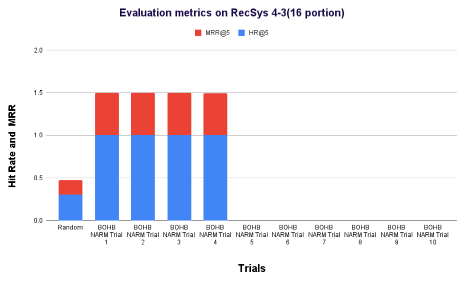
[18]https://nni.readthedocs.io/en/stable/Tutorial

[19]https://nni.readthedocs.io/en/stable/Tutorial

experiment will require a more powerful computer to accelerate experiments. All the results for this model can be found in Section 6.



(a) RECSYS 64 PORTION



(b) RECSYS 16 PORTION

Figure 18. Performance of NARM model

5.9.2 CSRM

5.9.2.1 Experiment setup

This experiment was carried on a computer with an NVIDIA ASUS Dual GeForce RTX™ 2080 EVO with a memory size of 8GB GDDR6. One parameter was added to the search space for the NNI to tune: the learning rate (type: uniform, value range: [0.0001, 0.1]).

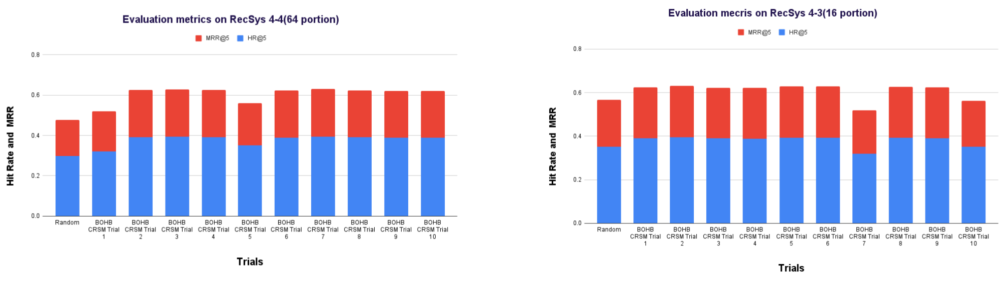
5.9.2.2 Results and discussion

Figures 19 show the performance comparison of the CSRM model using the BOHB tuner and random search. Overall the performance of the generated configurations from the tuner outperforms the random search configurations in RECSYS $\frac{1}{64}$, and $\frac{1}{16}$ portions datasets. Our results showed that the best architectural configuration (hidden units) generated by the tuner was the same as the one provided in the original paper. Other produced values resulted in "trial stopped," suggesting that having more or less hidden units was not improving the model's performance, resulting in the trial failing. Moreover, tuning the learning rate alone improved the model's performance. All the results done on each dataset for this experiment can be found in Section 6.

5.9.3 NextItNet

5.9.3.1 Experiment setup

This experiment was carried on a computer with a 3840 NVIDIA Titan Xp with a memory size of 12GB of GDDR5X. Two parameters were added to the search space for the NNI to tune: batch size (type: quniform, value range [30, 200, 10]), and the learning rate (type: uniform, value range: [0.0001, 0.1]).



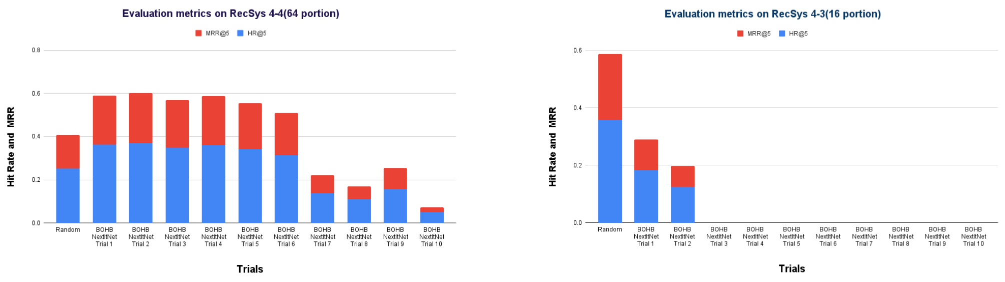
(a) RECSYS 64 PORTION

(b) RECSYS 16 PORTION

Figure 19. Performance of CSRM model

5.9.3.2 Results and discussion

Figures 20 show the performance comparison of the NextItNet model using the BOHB tuner and the random search. Similar to the results from the CSRM model, the best architectural configuration (dilated channels) generated by the tuner was the same as the one provided in the original paper resulting in similar behavior. Several configurations generated by the tuner outperform the random search in the RECSYS $\frac{1}{64}$ portion datasets. However, due to the size of RECSYS $\frac{1}{16}$ portion dataset and the complexity of the model [60], none of the trials matched the performance of the random search, suggesting that the tuner will probably need more time to find the best configurations for this dataset. All the results done on each dataset for this experiment can be found in Section 6.



(a) RECSYS 64 PORTION

(b) RECSYS 16 PORTION

Figure 20. Performance of NextItNet model

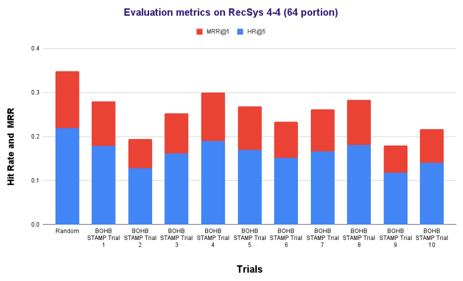
5.9.4 STAMP

5.9.4.1 Experiment setup

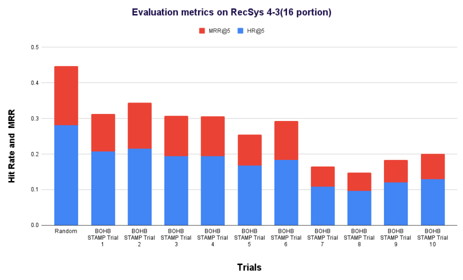
This experiment was carried on a computer with an NVIDIA ASUS Dual GeForce RTX™ 2080 EVO with a memory size of 8GB GDDR6. Two parameters were added to the search space for the NNI to tune: batch size (type: quniform, value range [50, 300, 10]) and the learning rate (type: uniform, value range: [0.0001, 0.1]).

5.9.4.2 Results and discussion

Figures 21 show the performance comparison of the STAMP model using the BOHB tuner and the random search. The overall performance is inadequate for the generated configurations from the tuner, especially for the RECSYS dataset, suggesting that the configurations generated from the same (learning rate and batch size) range used for all the experiments were not suitable for the STAMP model. Furthermore, Our results showed that the best architectural configuration (hidden size) generated by the tuner was the same as the one provided in the original paper. Other produced values resulted in "trial stopped," suggesting that having more or less hidden units was not improving the model's performance, resulting in the trial failing. A possible solution could be to reduce the learning rate range according to the model-specific characteristics, allowing the tuner to work more efficiently. Nevertheless, the only good performance was found using the same portion of the RETAIL ROCKET dataset. However, due to this dataset's small size, not much conclusion can be drawn. All the results done on each dataset for this experiment can be found in Section 6.



(a) RECSYS 64 PORTION



(b) RECSYS 16 PORTION

Figure 21. Performance of STAMP model

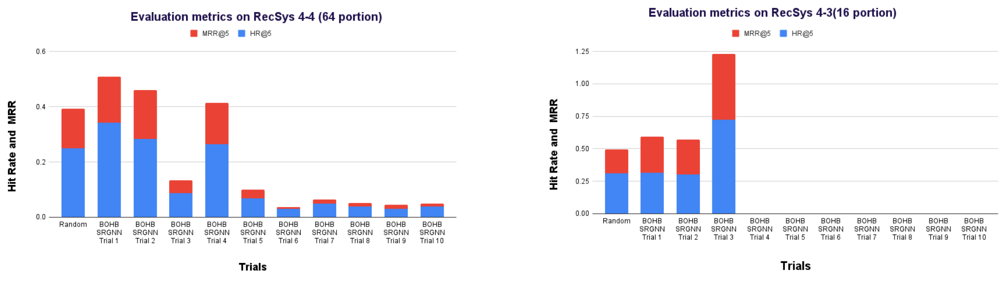
5.9.5 SRGNN

5.9.5.1 Experiment setup

This experiment was carried on a computer with an NVIDIA ASUS Dual GeForce RTX™ 2080 EVO with a memory size of 8GB GDDR6. Two parameters were added to the search space for the NNI to tune: batch size (type: quniform, value range [100, 200, 10]) and the learning rate (type: uniform, value range: [0.0001, 0.1]).

5.9.5.2 Results and discussion

Figures 22 show the performance comparison of SRGNN using the BOHB tuner and the random search. Compared to other models' architectural configurations generated by the tuner, the best architectural (Hidden size) configuration for the SRGNN model was slightly different from the one proposed in random search 110, 100 hidden layers, respectively. Our results showed that the configurations generated by the tuner outperform the random selection in all the datasets. However, in RECSYS $\frac{1}{16}$ portions, only three trials were completed. With the large size of the dataset, the experiment will require a more powerful computer to accelerate experiments. All the results for this model can be found in Section 6.



(a) RECSYS 64 PORTION (b) RECSYS 16 PORTION

Figure 22. Performance of SRGNN model

5.10 Interpretable meta-model for best model predictions

Based on our empirical study, we trained an interpretable decision tree model that can predict which model outperforms the others based on dataset characteristics. We used all the experiments that we have carried out to construct our new dataset. The features list in the new dataset includes the number of sessions, average session length, average item frequency both in the training and validation set. We set the target variable as the best performing model out of the whole list of models according to the MRR@5 evaluation metric. Our dataset was divided into 90-10% training and hold-out validation splits. A decision tree was trained to achieve an average of 81.8% and 88.89% for training and validation accuracy. The visualization of this decision tree model can be found in Figure 23. the most important features used in determining the outperforming model turn out to be the following order:

1. the average item frequency in the training set
2. the number of sessions in the training set
3. the total number of items and average session length of the validation set

This meta-model tree supports our previous findings of how different dataset characteristics can affect different models' performance and choose the best one. Our study suggests that using different models according to different datasets' characteristics could substantially result in the session-based recommendation. Using a similar approach to our decision-tree meta-model, we can easily predict which models can perform well with different dataset properties, and this information can help in combining predictions out of multiple candidate models, which will consequently improve the set of recommendations. Also, our dataset can be extended easily with more e-commerce datasets, which help increase the meta-model accuracy and reliability of predicting the best models.

Finally, this meta-model is used in our proposed framework to automatically recommend the best models according to the properties extracted from the uploaded dataset.

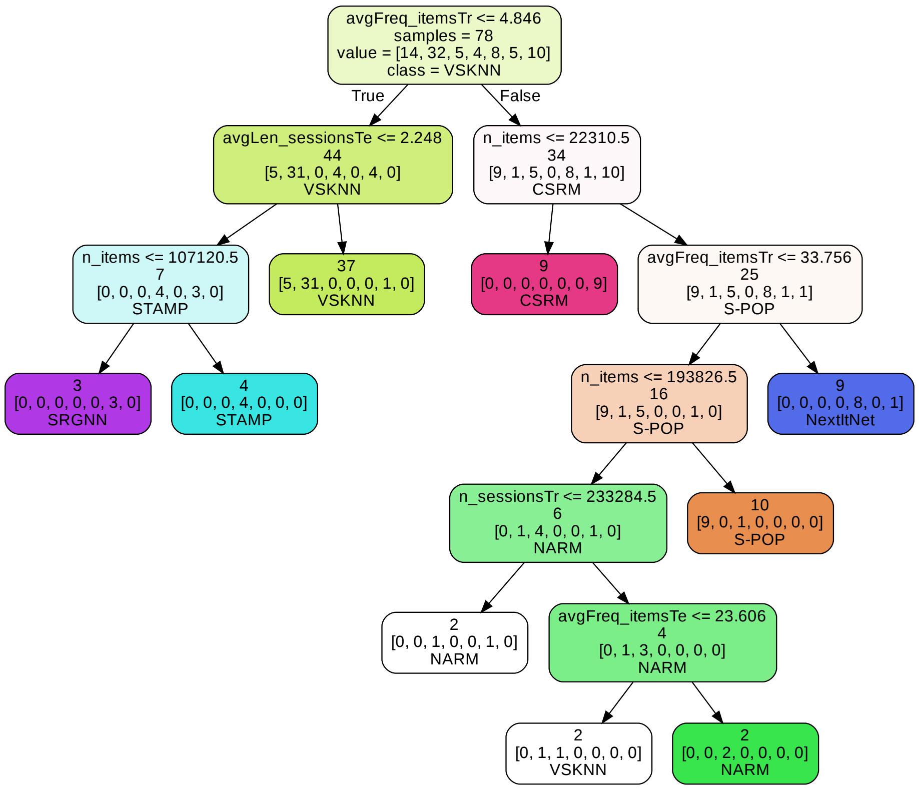
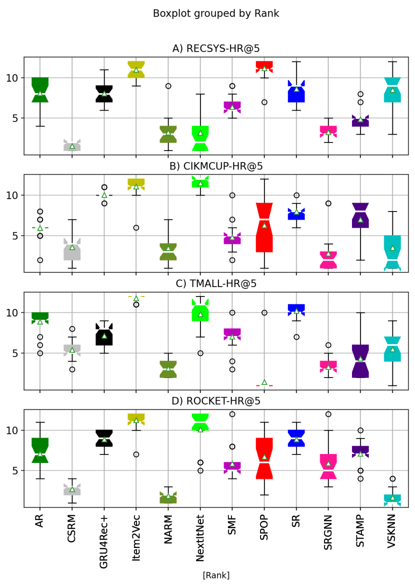


Figure 23. Decision Tree model trained to predict the best model to perform based on dataset characteristics.

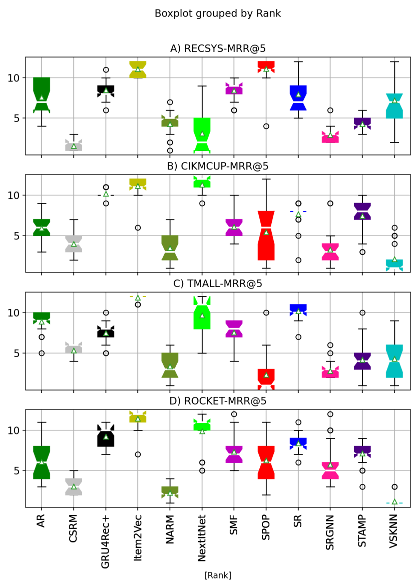
5.11 Overall performance

To judge the overall performance of different models, we used a box-plot in Figure 24 to represent the average ranking of the models in all the held out experiments along with the different datasets.

In general, NARM, SRGNN, and CSRM are the top three models for both HR@5 and MRR@5 in all datasets. VSKNN has a good performance in the ROCKET dataset, characterized by the smallest average session length among all datasets. Besides, S-POP has the best performance in the TMALL dataset, which has the largest number of items and average session length. NextItNet has a good performance just in the RECSYS dataset, which has the smallest number of items and the largest average item frequency in the training set, which means that most items are well-represented in training set by a large number of times.



(a) HR@5



(b) MRR@5

Figure 24. Ranking of different models on each dataset in all the experiments (The lowest rank is the better)

5.12 AutoiCV4Rec description

The user can upload a dataset in the proposed final assembly framework, as shown in Figure 26. After submitting the dataset, the preprocessing of the dataset will be done on it, followed by the features extraction. These features are then used in the meta-model to predict the model according to the uploaded dataset characteristics. When this process is completed, the proposed model's hyper-parameters are then tuned to find the best configuration that achieves the best evaluation metrics. Finally, the model can be downloaded for further training or used for prediction, as shown in the Figure 25.

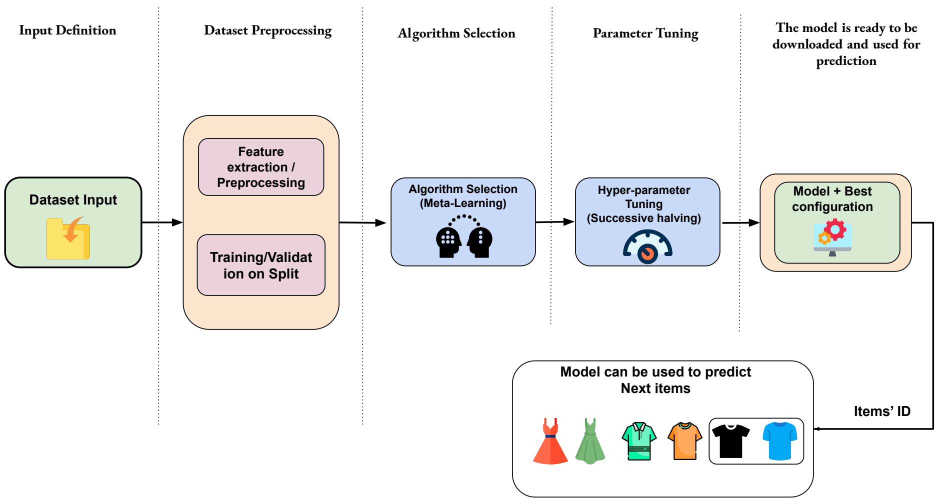
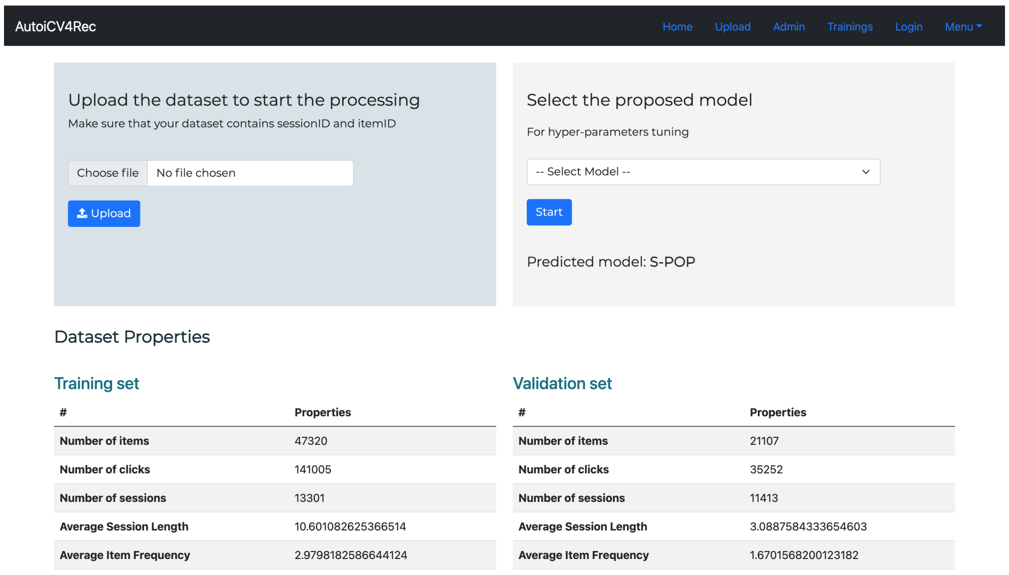

Input Definition
Dataset Preprocessing
Algorithm Selection
Parameter Tuning
The model is ready to be downloaded and used for prediction
Dataset Input
Feature extraction / Preprocessing
Training/Validation on Split
Algorithm Selection (Meta-Learning)
Hyper-parameter Tuning (Successive halving)
Model + Best configuration
Model can be used to predict Next items
Items' ID

Figure 25. AutoiCV4Rec: Framework Architecture


AutoiCV4Rec
Home
Upload
Admin
Trainings
Login
Menu
Upload the dataset to start the processing
Make sure that your dataset contains sessionID and itemID
Choose file
No file chosen
Upload
Select the proposed model
For hyper-parameters tuning
-- Select Model --
Start

Predicted model: S-POP

Dataset Properties

Training set

#	Properties
Number of items	47320
Number of clicks	141005
Number of sessions	13301
Average Session Length	10.601082625366514
Average Item Frequency	2.9798182586644124

Validation set

#	Properties
Number of items	21107
Number of clicks	35252
Number of sessions	11413
Average Session Length	3.0887584333654603
Average Item Frequency	1.6701568200123182

Figure 26. AutoiCV4Rec: User interface platform

6 Conclusion

Recommender systems have become a significant piece of any effective business that helps meets the user's needs and boost the business sales volume. However, choosing the best architecture for any particular business remains a challenge for scientists. Furthermore, the performance of recommender systems is susceptible to a plethora of design decisions such as choosing the suitable neural architectures, training procedures, regularization methods, and hyperparameters, which constitute a considerable obstacle for new users.

In this thesis, we have investigated the current state-of-the-art neural models and other baseline algorithms for the session-based recommendation. Additionally, we used the AutoML framework to tune the neural based-models hyperparameters to observe their performance. Different experiments have been carried out trying to answer a set of research questions covering the different dataset properties used during both training and inference phases. We used different evaluation metrics covering the accuracy measurement of recommendations made by the models, also the coverage of different items, and the average popularity of predicted items. Moreover, the consumption of computational resources during training and testing phases has been discussed in terms of the suitability of being applied in real-life e-commerce portals.

In general, neural-based models with attention mechanisms like NARM and CSRM are the most top-performing along with different dataset characteristics with appropriate training and prediction time consumption. Baseline models like nearest neighbors are still outperforming all algorithms when having relatively small training sizes and sessions of short average length. Our results suggest that training data recency and sizes substantially affect the accuracy measurements during inference time. In e-commerce, it is clear that dynamic time modeling is a crucial part of being further investigated and included in session-based algorithms to model general trends through different periods of time. Additionally, dataset characteristics like average session length, average item frequency, and the total number of sessions can significantly impact the chosen model performance. Moreover, most of the models' performance degrades heavily on very long sessions, which suggests the need to improve the model's performance on these cases and accurately detect drifts of user preferences while making use of old data in session efficiently. The second experiment showed that AutoML could improve the neural-based models by automatically selecting the best configurations with less budget. Our results showed that the overall budget that most researchers can afford during development is usually not much higher than that of fully training some of the models. Hence, practical HPO methods go beyond that to already yield good configurations with such a small budget. Moreover, the results suggest that HPO methods can easily handle problems varying from just a few to many dozens of hyperparameters (categorical, binary, integer, and continuous).

The possible future work could be the addition to our proposed framework the evaluation of session-based recommendation in domains other than e-commerce like music playlist recommendation, which has not yet been investigated. As different domain characteristics can affect the properties of data collected and the performance of different models, it is essential to answer similar research questions to those investigated here in other domains.

Appendix

I. Source Code

Source code for implementation and all the results of different methods proposed in this thesis are located in the following GitHub repository and links:

- `https://github.com/Blessing92/autoiCV`
- `https://docs.google.com/spreadsheets/d/1jlwHY4oCs9xXQdykgREsJxR3Eos0w_812zDCjQZt_5U/edit?usp=sharing`
- `https://drive.google.com/file/d/1m8aFpGa5IXS9MI0l8vXF8qoYT3DE4rlG/view?usp=sharing`

II. Licence

References

[1] S. Falkner A Zela, A. Klein and F. Hutter. "improving deep neural networks for lvcsr using rectified linear units and dropout.". *In: Adams, M., Zhao, V. (eds) International Conference on Acoustics, Speech and Signal Processing (ICASSP 13)*, pages 8609–8613, 2013.

[2] S. Falkner A Zela, A. Klein and F. Hutter. "hyperband: A novel bandit-based approach to hyperparameter optimization.". *Journal of Machine Learning Research 18(185)*, 185(1-52), 2018.

[3] S. Falkner A Zela, A. Klein and F. Hutter. "neural architecture search with reinforcement learning". *In Proceedings of the international Conference on Learning Representations (ICLR'17)*, 2018.

[4] S. Falkner A Zela, A. Klein and F. Hutter. "towards automated deep learning: Efficient joint neural architecture and hyperparameter search". *AutoML Workshop*, 20(55:1-55:21), 2018.

[5] Benjamin Schrauwen Aaron van den Oord, Sander Dieleman. Deep content-based music recommendation. *In Advances in neural information processing systems*, pp. 2643-2651, 2013.

[6] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[7] Mahdieh Atashkar and Faramarz Safi. Item-based recommender systems applying social-economic indicators. *SN Computer Science*, 1, 04 2020.

[8] J. Bergstra B. Komer and C. Eliasmith. "hyperopt-sklearn: Automatic hyperparameter configuration for scikit-learn". *workshop on Automated Machine Learning (AutoML workshop 2014)*, 2014.

[9] J. Bergstra B. Komer and C. Eliasmith. "neural architecture search: A survey". *J. Mach. Learn. Res*, 2019.

[10] B. Hidasi A. Karatzoglou L. Baltrunas and D. Tikk. "session-based recommendations with recurrent neural networks". *arXiv preprint*, arXiv: 1511.06939, 2015.

[11] Oren Barkan and Noam Koenigstein. Item2vec: Neural item embedding for collaborative filtering. *2016 IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6, 2016.

[12] Y. LeCun Y. Bengio and G. Hinton. "deep learning". *nature*, 521(no. 7553):p.436, 2015.

[13] Bengio Y Bergstra J, Bardenet R and Kegl B. "algorithms for hyper-parameter optimization.". *NeurIPS'11*, pages 2546–2554, 2011.

[14] Giuseppe Bonaccorso. *Machine Learning Algorithms: A Reference Guide to Popular Algorithms for Data Science and Machine Learning*. Packt Publishing, 2017.

[15] Geoffray Bonnin and Dietmar Jannach. Automated generation of music playlists: Survey and experiments. *ACM Computing Surveys*, 47:1–35, 01 2014.

[16] H. Hoos C. Thornton, F.Hutter and K. Leyton-Brown. "auto-weka: combined selection and hyperparameter optimization of classification algorithms". *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'13)*, pages 847–855, 2013.

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

[18] Wanyu Chen, Fei Cai, Honghui Chen, and Maarten de Rijke. A dynamic co-attention network for session-based recommendation. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 1461–1470, New York, NY, USA, 2019. Association for Computing Machinery.

[19] Chen Cheng, Haiqin Yang, Michael R. Lyu, and Irwin King. Where you like to go next: Successive point-of-interest recommendation. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, IJCAI '13, page 2605–2611. AAAI Press, 2013.

[20] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[21] Radwa Elshawi, Mohamed Maher, and Sherif Sakr. Automated machine learning: State-of-the-art and open challenges. *arXiv preprint arXiv:1906.02287*, 2019.

[22] Russakovsky O. Deng J. Su H. Krause J. Satheesh S. Ma S. et al. "imagenet large scale visual recognition challenge". *International Journal of Computer vision*, 115(3):pp.211–252, 2015.

[23] Szededy C. Liu W. Jia Y. Sermanet P. Reed S. Anguelov D. et al. "going deeper with convolutions". *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages pp.1–9, 2015.

[24] Klein A Falkner, S. and Hutter F. "bohb: Robust and efficient hyperparameter optimization at scale". *PMLR*, pages 1437–1446, 2018.

[25] Stefan Falkner, Aaron Klein, and Frank Hutter. BOHB: robust and efficient hyperparameter optimization at scale. *CoRR*, abs/1807.01774, 2018.

[26] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. Graph neural networks for social recommendation, 2019.

[27] L. Kottholff F.Hutter and J. Vanschoren. "automatic machine learning: Methods, system, challenges". *Challenges in Machine Learning*, 2019.

[28] Florent Garcin, Christos Dimitrakakis, and Boi Faltings. Personalized news recommendation with context trees. In *Proceedings of the 7th ACM Conference on Recommender Systems*, RecSys '13, page 105–112, New York, NY, USA, 2013. Association for Computing Machinery.

[29] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines, 2014.

[30] Q. Song H. Jin and X.Hu. "auto-keras: An efficient neural architecture search system". *In A. Teredesai, V. kumar, Y. Li, R. Rosales, E. Terzi, and G. Karypis, editors, Proeedings of the 25th ACM SIGKDD International Conference on knowledge Discovery and Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8*, pages 1946–1956, 2019.

[31] R. He and J. McAuley. Fusing similarity models with markov chains for sparse sequential recommendation. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 191–200, Los Alamitos, CA, USA, dec 2016. IEEE Computer Society.

[32] Hansika Hewamalage, Christoph Bergmeir, and Kasun Bandara. Recurrent neural networks for time series forecasting: Current status and future directions. *CoRR*, abs/1909.00590, 2019.

[33] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, page 241–248, New York, NY, USA, 2016. Association for Computing Machinery.

[34] Balázs Hidasi and Alexandros Karatzoglou. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 843–852, 10 2018.

[35] Mehdi Hosseinzadeh Aghdam, Negar Hariri, Bamshad Mobasher, and Robin Burke. Adapting recommendations to contextual changes using hierarchical hidden markov models. In *Proceedings of the 9th ACM Conference on Recommender Systems*, RecSys '15, page 241–244, New York, NY, USA, 2015. Association for Computing Machinery.

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

[37] K. Jamieson and A Talwalkar. Non-stochastic best arm identification and hyperparamter optimization. *AISTATS*, 2016.

[38] Dietmar Jannach, Lukas Lerche, and Michael Jugovac. Adaptation and evaluation of recommendations for short-term shopping goals. In *Proceedings of the 9th ACM Conference on*

Recommender Systems, RecSys '15, page 211–218, New York, NY, USA, 2015. Association for Computing Machinery.

[39] Dietmar Jannach, Bamshad Mobasher, and Shlomo Berkovsky. Research directions in session-based and sequential recommendation: A preface to the special issue. *User Modeling and User-Adapted Interaction*, 30, 08 2020.

[40] I. Kamehkhosh D. Jannach and M. Ludewig. "a comparison of frequent pattern techniques and a deep learning method for session-based recommendation". *in RecTemp@ RecSys*, pages pp.50–56, 2017.

[41] Santosh Kabbur, Xia Ning, and George Karypis. 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*, KDD '13, page 659–667, New York, NY, USA, 2013. Association for Computing Machinery.

[42] M. Ludewig N. Mauro S. Latifi and D. Jannach. "empirical analysis of session-based recommendation algorithms". *ArXiv preprint*, arXiv: 1910.12781, 2019.

[43] D. Kim C. Park. J. Oh S. Lee and H. "convolutional matrix factorization for document context-aware recommendation". *Proceedings of the 10th ACM Conference on Recommender Systems*, pages pp.233–240, 2016.

[44] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, page 1419–1428, New York, NY, USA, 2017. Association for Computing Machinery.

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

[46] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 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*, pages 1831–1839, 07 2018.

[47] M. Ludewig and D. Jannach. "when recurrent neural networks meet the neighborhood for session-based recommendation". *In Proceedings of the Eleventh ACM Conference on Recommender Systems*, pages pp.306–310, 2017.

[48] M. Ludewig and D. Jannach. "evaluation of session-based recommendation algorithms". *User Modeling and User-Adapted Interaction*, 28(4-5):pp.331–390, 2018.

[49] Paolo Cremonesi Massimo Quadrana and Dietmar Jannach. "sequence-aware recommender systems". *ACM Comput. Surv.1 1*, page 35, 2018.

[50] Hilario M Nguyen P and Kalousis A. "using meta-mining to support data mining workflow planning and optimization.". *Journal of Artificial Intelligence Research 51*, pages 605–644, 2014.

[51] Mehrbakhsh Nilashi, Karamollah Bagherifard, Assoc Prof. Dr. Othman Ibrahim, Hamid Alizadeh, Ayodele Lasisi, and Nazanin Roozegar. Collaborative filtering recommender systems. *Research Journal of Applied Sciences, Engineering and Technology*, 5:4168–4182, 04 2013.

[52] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. Sequence-aware recommender systems, 2018.

[53] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, page 130–137, New York, NY, USA, 2017. Association for Computing Machinery.

[54] Retailrocket. Dkn: Deep knowledge-aware network for news recommendation. *In Proceedings of the 2018 World Wide Web Conference*, pp.1835-1844, 2018.

[55] Michael T. Rosenstein, Zvika Marx, Leslie Pack Kaelbling, and Thomas G. Dietterich. To transfer or not to transfer. In *In NIPS'05 Workshop, Inductive Transfer: 10 Years Later*, 2005.

[56] Massimiliano Ruocco, Ole Steinar Lillestøl Skrede, and Helge Langseth. Inter-session modeling for session-based recommendation, 2017.

[57] Y. Zhu L. Wang X. Xie S. Wu Y. Tang and T. Tan. "session-based recommendation with graph neural networks". *In Proceedings of the AAAI Conference on Artificial intelligence*, 33:pp.346–353, 2019.

[58] Harald Steck. Item popularity and recommendation accuracy. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, page 125–132, New York, NY, USA, 2011. Association for Computing Machinery.

[59] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer, 2019.

[60] Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 565–573, 02 2018.

[61] Maryam Tavakol and Ulf Brefeld. Factored mdps for detecting topics of user sessions. In *Proceedings of the 8th ACM Conference on Recommender Systems*, RecSys '14, page 33–40, New York, NY, USA, 2014. Association for Computing Machinery.

[62] Trinh Xuan Tuan and Tu Minh Phuong. 3d convolutional networks for session-based recommendation with content features. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, RecSys '17, page 138–146, New York, NY, USA, 2017. Association for Computing Machinery.

[63] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[64] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[65] H. Wang N. Wang and D. Y Yeung. "collaborative deep learning for recommender systems". *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages pp.1235–1244, 2015.

[66] Meirui Wang, Pengjie Ren, Lei Mei, Zhumin Chen, Jun Ma, and Maarten Rijke. A collaborative session-based recommendation approach with parallel memory modules. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 345–354, 07 2019.

[67] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M. Jose, and Xiangnan He. A simple convolutional generative network for next item recommendation. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, WSDM '19, page 582–590, New York, NY, USA, 2019. Association for Computing Machinery.