Deep Learning for Sequential Recommendation: Algorithms, Influential Factors, and Evaluations

HUI FANG, RIIS & SIME, Shanghai University of Finance and Economics, China
DANNING ZHANG*, SIME, Shanghai University of Finance and Economics, China
YIHENG SHU, Software College, Northeastern University, China
GUIBING GUO, Software College, Northeastern University, China

In the field of sequential recommendation, deep learning (DL)-based methods have received a lot of attention in the past few years and surpassed traditional models such as Markov chain-based and factorization-based ones. However, there is little systematic study on DL-based methods, especially regarding how to design an effective DL model for sequential recommendation. In this view, this survey focuses on DL-based sequential recommender systems by taking the aforementioned issues into consideration. Specifically, we illustrate the concept of sequential recommendation, propose a categorization of existing algorithms in terms of three types of behavioral sequences, summarize the key factors affecting the performance of DL-based models, and conduct corresponding evaluations to showcase and demonstrate the effects of these factors. We conclude this survey by systematically outlining future directions and challenges in this field.

CCS Concepts: • **Information systems** → **Recommender systems**.

Additional Key Words and Phrases: sequential recommendation, session-based recommendation, deep learning, influential factors, survey, evaluations

ACM Reference Format:
Hui Fang, Danning Zhang, Yiheng Shu, and Guibing Guo. 2020. Deep Learning for Sequential Recommendation: Algorithms, Influential Factors, and Evaluations. *ACM Transactions on Information Systems* 1, 1, Article 1 (January 2020), 41 pages. https://doi.org/10.1145/3426723

1 INTRODUCTION

With the prevalence of information technology (IT), recommender system has long been acknowledged as an effective and powerful tool for addressing information overload problem. It makes users easily filter and locate information in terms of their preferences, and allows online platforms to widely publicize the information they produce. Most traditional recommender systems are content-based and collaborative filtering based ones. They strive to model users' preferences towards items on the basis of either explicit or implicit interactions between users and items. Specifically, they incline to utilize a user's historical interactions to learn his/her static preference with the assumption that all user-item interactions in the historical sequences are equally important.

*Corresponding author

Authors' addresses: Hui Fang, RIIS & SIME, Shanghai University of Finance and Economics, China, fang.hui@mail.shufe.edu.cn; Danning Zhang, SIME, Shanghai University of Finance and Economics, China, zhangdanning5@gmail.com; Yiheng Shu, Software College, Northeastern University, China, shuyiheng29@gmail.com; Guibing Guo, Software College, Northeastern University, China, guogb@swc.neu.edu.cn.

1046-8188/2020/1-ART1 $15.00
https://doi.org/10.1145/3426723

However, this might not hold in real-world scenarios, where the user's next behavior not only depends on the static long-term preference, but also relies on the current intent to a large extent, which might be probably inferred and influenced by a small set of the most recent interactions. On the other side, the conventional approaches always ignore to consider the sequential dependencies among the user's interactions, leading to inaccurate modeling of the user's preferences. Therefore, sequential recommendation has become increasingly popular in academic research and practical applications.

Sequential recommendation (identical to sequence-aware recommendation in [77]) is also related to session-based, or session-aware recommendation. Considering that the latter two terms can be viewed as the sub-types of sequential recommendation [77], we thus use the much broader term *sequential recommendation* to describe the task that explores the sequential data.

For sequential recommendation, besides capturing users' long-term preferences across different sessions as the conventional recommendation does, it is also extremely important to simultaneously model users' short-term interest in a session (or a short sequence) for accurate recommendation. Regarding the time dependency among different interactions in a session as well as the correlation of behavior patterns among different sessions, traditional sequential recommender systems are particularly interested in employing appropriate and effective machine learning (ML) approaches to model sequential data, such as Markov Chain [21] and session-based KNN [31, 39], which are criticized by their incomplete modeling problem, as they fail to thoroughly model users' long-term patterns by combining different sessions.

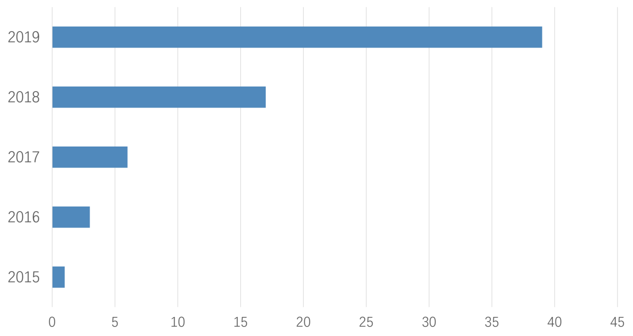


Fig. 1. The number of arXiv articles on DL-based sequential recommendation in 2015-2019.

In recent years, deep learning (DL) techniques, such as recurrent neural network (RNN), obtain tremendous achievements in natural language processing (NLP), demonstrating their effectiveness in processing sequential data. Thus, they have attracted increasing interest in the sequential recommendation, and many DL-based models have achieved state-of-the-art performance [147]. The number of relevant arXiv articles posted in 2015-2019 is shown in Figure 1[1], where we can see that the interest in DL-based sequential recommendation has increased phenomenally. Besides, common application domains of sequential recommendation include e-commerce (e.g., RecSys

[1]We searched on arXiv.org for articles using the related keywords as well as their combinations, such as *sequential recommendation*, *deep learning* and *session-based recommendation* in June 2020. We did not report the data of 2020 in Figure 1 due to its incomplete. Figure 2 considers all articles.

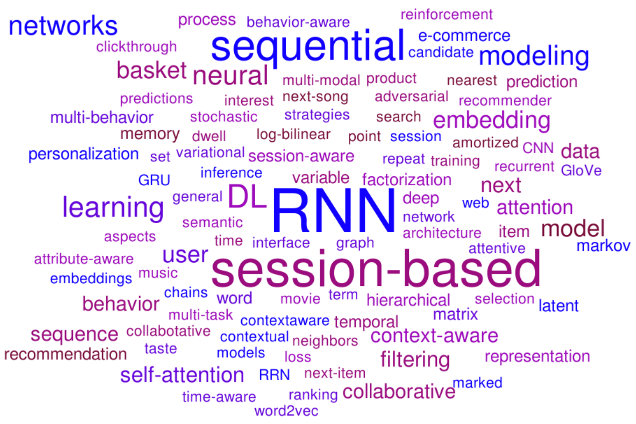


Fig. 2. Word cloud of the keywords for sequential recommendation related articles.

Challenge 2015[2]), POI (Point-of-Interest), music (e.g., Last.fm[3]), and movie/video (e.g., MovieLens[4]). Figure 2 depicts the word cloud of keywords in the sequential recommendation related articles, which to some extent reflects the hot topics in the field of sequential recommendation.

More and more new techniques and improved structures have been applied to facilitate the DL-based sequential recommendation. However, we find that the DL-based models still have some common drawbacks. For example, many existing works do not take items and users into consideration at the same important level, i.e., they largely emphasize item representation, but lack of a careful design on user representation. Besides, they merely consider all interactions into one type, instead of distinguishing different types. More importantly, although more advanced techniques have been increasingly adopted, it is still unclear the real progress for this area, as only complex DL structures could not possibly yield better performance [19]. In this case, it becomes extremely necessary to unveil the influential factors leading to a useful DL-based sequential recommender systems. Therefore, we tend to thoroughly discuss the improved techniques and the aforementioned issues in this survey. The contributions of this survey are concluded as follows:

- We provide a comprehensive overview of the sequential recommendation in terms of the deployed DL techniques. To the best of our knowledge, it is the first survey article on DL-based sequential recommendation.
- We propose an original classification framework for the sequential recommendation, corresponding to three different recommendation scenarios, which can contribute to the existing taxonomies as regards sequential recommendation.
- We summarize the influential factors for typical DL-based sequential recommendation and demonstrate their impacts on the sequential recommendation with regard to recommendation accuracy via a well-designed empirical study, which can serve as a guidance for sequential recommendation research and practices. Besides, our experimental settings and experimental repositories (including the source codes and datasets) can be viewed as a valuable testbed for future research.

[2]www.kaggle.com/chadgostopp/recsys-challenge-2015.
[3]labrosa.ee.columbia.edu/millionsong/lastfm.
[4]movielens.org.

- We summarize quite a few open issues in existing DL-based sequential recommendation and outline future directions.

1.1 Related Survey

There have been some surveys on either DL-based recommendation or sequential recommendation. For DL-based recommendation, Singhal et al. [92] summarized DL-based recommender systems and categorized them into three types: collaborative filtering, content-based, and hybrid ones. Batmaz et al. [8] classified and summarized the DL-based recommendation from the perspectives of DL techniques and recommendation issues, and also gave a brief introduction of the session-based recommendations. Zhang et al. [147] further discussed the state-of-the-art DL-based recommender systems, including several RNN-based sequential recommendation algorithms. For sequential recommendation, Quadrana et al. [77] proposed a categorization of the recommendation tasks and goals, and summarized existing solutions. Wang et al. [121] summarized the key challenges, progress and future directions for sequential recommender systems. In a more comprehensive manner [119], they further illustrated the value and significance of the session-based recommender systems (SBRS), and proposed a hierarchical framework to categorize issues and methods, including some DL-based ones.

However, to the best of our knowledge, our survey is the first to specifically and systematically summarize and explore DL-based sequential recommendation, and discuss the common influential factors using a thorough demonstration of experimental evaluations on several real datasets. The experiment results and conclusions can further guide the future research on how to design an effective DL model for the sequential recommendation.

1.2 Structure of This Survey

The rest of the survey is organized as follows. In Section 2, we provide a comprehensive overview of DL-based sequential recommender systems, including a careful refinement of sequential recommendation tasks. In Section 3, we present the details of the representative algorithms for each recommendation task. In Section 4, we summarize the influential factors for existing DL-based sequential recommendation followed by a thorough evaluation on real datasets in Section 5. Finally, we conclude this survey by presenting open issues and future research directions of DL-based sequential recommendation in Section 6.

2 OVERVIEW OF SEQUENTIAL RECOMMENDATION

In this section, we provide a comprehensive overview of the sequential recommendation. First, we clarify the related concepts, and then formally describe the sequential recommendation tasks. Finally, we elaborate and compare the traditional ML and DL techniques for the sequential recommendation.

2.1 Concept Definitions

To facilitate the understanding, we first formally define *behavior object* and *behavior type* to distinguish different user behaviors in sequential data.

Definition 2.1. behavior object refers to the items or services that a user chooses to interact with, which is usually presented as an ID of an item or a set of items. It may be also associated with other information including text descriptions, images and interaction time. For simplicity, we often use *item*(s) to describe behavior object(s) in the following sections.

Definition 2.2. behavior type refers to the way that a user interacts with items or services, including *search*, *click*, *add-to-cart*, *buy*, *share*, etc.

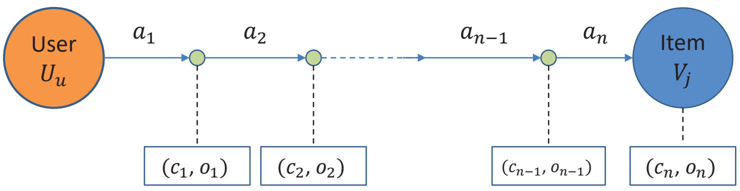


Fig. 3. A schematic diagram of the sequential recommendation. c_i: behavior type, o_i: behavior object. A behavior a_i is represented by a 2-tuple, i.e., $a_i = (c_i, o_i)$. A behavior sequence (i.e., behavior trajectory) is a list of 2-tuples in the order of time.

Given these concepts, a **behavior** can be considered as a combination of a behavior type and a behavior object, i.e., a user interacting with a behavior object by a behavior type. A **behavior trajectory** can be thus defined as a *behavior sequence* (or behavior session) consisting of multiple user behaviors. A typical behavior sequence is shown in Figure 3. Specifically, a behavior (a_i) is represented by a 2-tuple (c_i, o_i), i.e., a behavior type c_i and behavior object o_i. A user who generates the sequence can either be anonymous or identified by his/her ID. The behaviors in the sequence are sorted in time order. When a single behavior involves with multiple objects (e.g., items recorded in a shopping basket), objects within the basket may not be ordered by time, and then multiple baskets together form a behavior sequence. It should be noted that *sequence* and *session* are interchangeably used in this paper.

Thus, a **sequential recommender system** is referred to a system which takes a user's behavior trajectories as input, and then adopts recommendation algorithms to recommend appropriate items or services to the user. The input behavior sequence $\{a_1, a_2, a_3, ..., a_t\}$ is polymorphic, which can thus be divided into three types[5]: *experience-based*, *transaction-based* and *interaction-based* behavior sequence, and the details are elaborated as follows:

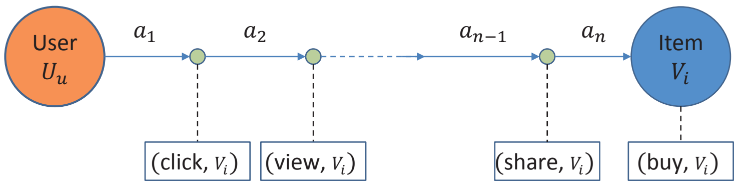


Fig. 4. Experience-based behavior sequence.

Experience-based behavior sequence. In an experience-based behavior sequence (see Figure 4), a user may interact with a *same object* (e.g., item v_i) multiple times by *different behavior types*. For example, a user's interaction history with an item might be as follows: first *searches* related keywords, then *clicks* the item of interest on the result pages followed by *viewing* the details of the item. Finally, the user may *share* the item with his/her friends and *add it to cart* if he/she likes it. Different behavior types as well as their orders might indicate users' different intentions. For instance, *click* and *view* can only show a user's interest of a low degree, while *share* behavior appears before (or after) *purchase* might imply a user's strong desire (or satisfaction) to obtain (or have) the item. *For this type of behavior sequence, a model is expected to capture a user's underlying*

[5]We discuss the sequence in a finer granularity in the purpose of better understanding the sequential recommendation task. We name the three types mainly according to the behavior types and objects involved in a sequence. We argue that it can promote better designs of network structures to process the corresponding sequences for sequential recommendation.

intentions indicated by different behavior types. The goal here is to predict the next behavior type that the user will exert given an item.

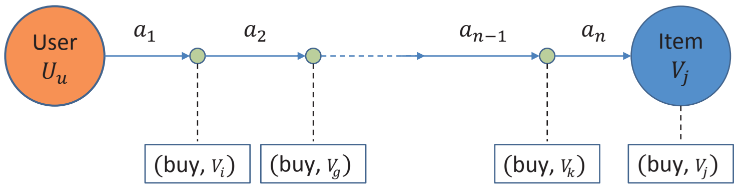


Fig. 5. Transaction-based behavior sequence.

Transaction-based behavior sequence. A transaction-based behavior sequence (see Figure 5) records a series of *different behavior objects* that a user interacts with, but with a *same behavior type* (i.e., *buy*). In practice, *buy* is the most concerned one for online sellers. Therefore, *with the transaction-based behavior sequence as input, the goal of a sequential recommender system is to recommend the next object (item) that a user will buy in view of the historical transactions of the user.*

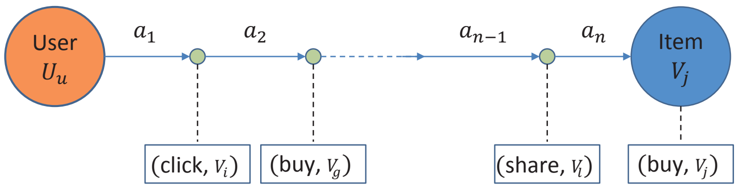


Fig. 6. Interaction-based behavior sequence.

Interaction-based behavior sequence. An interaction-based behavior sequence could be viewed as a mixture of experience-based and transaction-based behavior sequences (see Figure 6), i.e., a generalization of previous two types and much closer to the real scenarios. That is to say, it consists of *different behavior objects* and *different behavior types* simultaneously. *In interaction-based behavioral sequence modeling, a recommender system is expected to understand user preferences more realistically, including different user intents expressed by different behavior types and preferences implied by different behavior objects. Its major goal is to predict the next behavior object that a user will interact with.*

2.2 Sequential Recommendation Tasks

Before formally defining the sequential recommendation tasks, we firstly summarize the two representative tasks in the literature (as depicted in Figure 7): *next-item recommendation* and *next-basket recommendation*. In **next-item recommendation**, a behavior contains only one object (i.e., item), which could be a product, a song, a movie, or a location. In contrast, in **next-basket recommendation**, a behavior contains more than one object.

However, although the input of the aforementioned recommendation tasks is varied, their goals are mostly identical. Specifically, both of them strive to predict the next item(s) for a user, whilst the most popular form of the output is the top-N ranked item list. The rank could be determined by probabilities, absolute values or relative rankings, while in most cases *softmax function* is adopted to generate the output. Tan et al. [103] further proposed an embedding version of the softmax output for fast prediction to accommodate the large volume of items in recommendation.

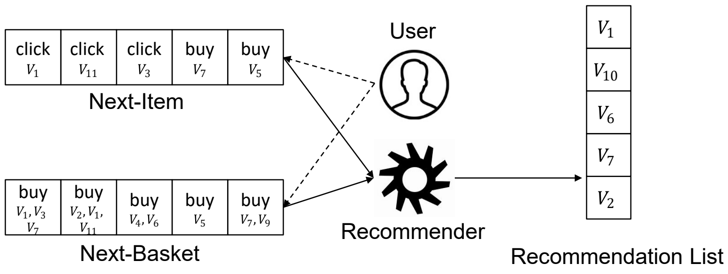


Fig. 7. Next-item and next-basket recommendation.

In this paper, we consider the task of sequential recommendation as to generate a personalized ranked item list on the basis of the three types of user behavior sequences (input), which can be formally defined as:

$$(p_1, p_2, p_3, ..., p_I) = f(a_1, a_2, a_3, ..., a_t, u) \tag{1}$$

where the input is the behavior sequence $\{a_1, a_2, a_3, ..., a_t\}$, u refers to the corresponding user of the sequence, and p_i denotes the probability that item i will be liked by user u at time $t+1$. I represents the number of candidate items. In other words, the sequential recommendation task is to learn a complex function f for accurately predicting the probability that user u will choose each item i at time $t + 1$ based on the input behavior sequence and the user profile.

According to the definition, and given the three types of behavior sequences, we thus divide the sequential recommendation tasks into three categories: *experience-based sequential recommendation*, *transaction-based sequential recommendation*, and *interaction-based sequential recommendation*. We will comprehensively discuss these tasks as well as the specific DL-based recommendation models in Section 3.

2.3 Related Models

In this subsection, we first review the traditional ML methods applied to the sequential recommendation and also briefly discuss their advantages and disadvantages. Second, we summarize related DL techniques for the sequential recommendation and elaborate how they overcome the issues involved in traditional methods.

2.3.1 Traditional Methods. Conventional popular methods for the sequential recommendation include frequent pattern mining, K-nearest neighbors, Markov chains, matrix factorization, and reinforcement learning [77]. They generally adopt matrix factorization for addressing users' long-term preferences across different sequences, whilst use first-order Markov Chains for capturing users' short-term interest within a sequence [31]. We next introduce the traditional methods as well as the representative algorithms for the sequential recommendation.

Frequent pattern mining. As we know, association rule [64] strives to use frequent pattern mining to mine frequent patterns with sufficient support and confidence. In the sequential recommendation, patterns refer to the sets of items which are frequently co-occurred within a sequence, and then are deployed to make recommendations. Although these approaches are easy to implement, and relatively explicable for users, they suffer from the limited scalability problem as matching patterns for recommendation is extremely strict and time-consuming.

Besides, determining suitable thresholds for support and confidence is also challenging, where a low minimum support or confidence value will lead to too many identified patterns, while a large value will merely mine co-occurred items with very high frequency, resulting in that only few items can be recommended or few users could get effective recommendation.

K-nearest neighbors (KNN). It includes item-based KNN and session-based KNN for the sequential recommendation. Item-based KNN [21, 57] only considers the last behavior in a given session and recommends items that are most similar to its behavior object (item), where the similarities are usually calculated via the cosine similarity or other advanced measurements [97].

In contrast, session-based KNN [42, 51, 57] compares the entire existing session with all the past sessions to recommend items via calculating similarities using Jaccard index or cosine similarity on binary vectors over the item space. KNN methods can generate highly explainable recommendation. Besides, as the similarities can be pre-calculated, KNN-based recommender systems could generate recommendations promptly. However, this kind of algorithms generally fails to consider the sequential dependency among items.

Markov chains (MC). In the sequential recommendation, Markov models assume that future user behaviors only depend on the last or last few behaviors. For example, [30] merely considered the last behavior with first-order MC, while [29, 31] adopted high-order MCs, which take the dependencies with more previous behaviors into account. Considering only the last behavior or several behaviors makes the MC-based models unable to leverage the dependencies among behaviors in a relatively long sequence and thus fails to capture intricate dynamics of more complex scenarios. Besides, they might also suffer from data sparsity problems.

Factorization-based methods. Matrix factorization (MF) tries to decompose the user-item interaction matrix into two low-rank matrices. For example, BPR-MF [82] optimizes a pairwise ranking objective function via stochastic gradient descent (SGD). Twardowski [108] proposed a MF-based sequential recommender system (a simplified version of Factorization Machines [84]), where only the interaction between a session and a candidate item is considered for generating recommendations. FPMC [83] is a representative baseline for next-basket recommendation, which integrates the MF with first-order MCs. FISM [44] conducts matrix factorization on an item-item matrix, and thus no explicit user representation is learned. On the basis of FISM, FOSSIL [31] tackles the sequential recommendation task by combining similarity-based methods and high-order Markov Chains. It performs better on sparse datasets in comparison with the traditional MC methods and FPMC. The main drawbacks of MF-based methods lie in: 1) most of them only consider the low-order interactions (i.e., first-order and second-order) among latent factors, but ignore the possible high-order interactions; and 2) excepts for a handful of algorithms considering temporal information (e.g., TimeSVD++ [47]), they generally ignore the time dependency among behaviors both within a session and across different sessions.

Reinforcement learning (RL). The essence of RL methods is to update recommendations according to the interactions between users and the recommender systems. When a system recommends an item to a user, a positive reward is assigned if the user expresses his/her interest on the item (via behaviors such as click or view). It is usually formulated as a Markov decision process (MDP) with the goal of maximizing the cumulative rewards in a set of interactions [90, 151]. With RL frameworks, sequential recommender systems can dynamically adapt to users (changing) preferences. However, similar to DL-based approaches, this kind of works is also lack of interpretability. Besides, more importantly, there is few appropriate platforms or resources for developing and testing RL-based methods in academia .

2.3.2 Deep Learning Techniques. In this subsection, we summarize the DL models (e.g., RNN and CNN) that have been adopted in the sequential recommendation in the literature.

Recurrent neural networks (RNNs). The effectiveness of RNNs in sequence modeling have been widely demonstrated in the field of natural language processing (NLP). In the sequential recommendation, RNN-based models are in the majority of DL-based models [17]. In comparison with the traditional models, RNN-based sequential recommendation models can well capture the dependencies among items within a session or across different sessions. The main limitation of RNNs for the sequential recommendation is that it is relatively difficult to model dependencies in a longer sequence (although could be somehow mitigated by other techniques), and training is burdened with the high cost especially with the increase of sequence length.

Convolutional neural networks (CNNs). CNN is commonly applied to process time series data (e.g., signals) and image data, where a typical structure consists of convolution layers, pooling layers, and feed-forward full-connected layers. It is suitable to capture the dependent relationship across local information (e.g., the correlation between pixels in a certain part of an image or the dependencies between several adjacent words in a sentence). In the sequential recommendation, CNN-based models can well capture local features within a session, and also could take the time information into consideration in the input layer [105, 107].

Multi-layer perceptrons (MLPs). MLPs refer to feed-forward neural networks with multiple hidden layers, which can thus well learn the nonlinear relationship between the input and output via nonlinear activation functions (e.g., tanh and ReLU). Therefore, MLP-based sequential recommendation models are expected to well capture the complex and nonlinear relationships among users behaviors [128].

Attention mechanisms. Attention mechanism in deep learning is intuited from visual attentions of human-beings (incline to be attracted by more important parts of a target object). It is originated from the work of Bahdanau et al. [4], which proposes an attention mechanism in neural machine translation task to focus on modeling the importance of different parts of the input sentence on the output word. Grounded on the work, *vanilla attention* is proposed by applying the work as a decoder of the RNN, and has been widely used in the sequential recommendation [53]. On the other hand, *self-attention mechanism* (originated in transformer [110] for neural machine translation by Google 2017) has also been deployed in the sequential recommendation. In contrast with vanilla attention, it does not include RNN structures, but performs much better than RNN-based models in recommender systems [146].

Graph neural networks (GNNs). GNN [154] can collectively aggregate information from the graph structure. Due to its effectiveness and superior performance in many applications, it has also obtained increasing interest in recommender systems. For example, Wu et al. [131] first used GNN for session-based recommendation by capturing more complex relationships between items in a sequence, and each session is represented as the composition of the long-term preference and short-term interests within a session using an attention network.

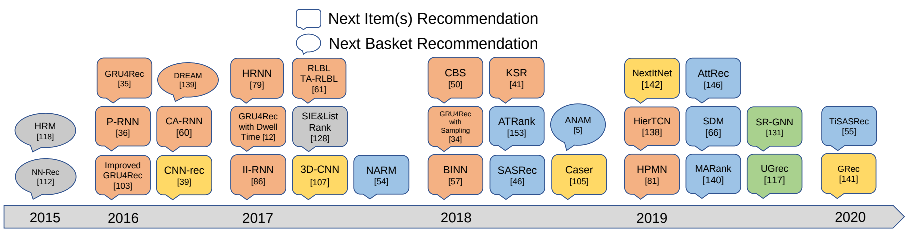


Fig. 8. Some recent and representative DL-based sequential recommendation models. Different colors indicate different DL techniques (grey: MLP; orange: RNN; yellow: CNN; blue: attention mechanism; green: GNN).

2.3.3 *Concluding Remarks.* Compared with conventional methods, DL-based methods are a much more active research area in the recent years. The MC- and MF-based models assume that a user's next behavior is related to only a few recent behavior(s), while DL methods utilize a much longer sequence for prediction [119], as they are able to effectively learn the *theme* of the whole sequence. Thus, they generally obtain better performance (in terms of accuracy measurement) than traditional models. Meanwhile, DL methods are more robust to sparse data and can adapt to varied length of the input sequence. The representative DL-based sequential recommendation algorithms are presented in Figure 8, which will be introduced in details in next sections.

The major problems of DL-based sequential recommendation methods include: 1) they are lack of explanability for the generated recommendation results. Besides, it is also difficult to calibrate why the recommendation models are effective, and thus to yield a robust DL-based model for varied scenarios; 2) the optimization is generally extremely challenging and more training data is required for complex networks.

3 SEQUENTIAL RECOMMENDATION ALGORITHMS

In this section, in order to figure out whether sequential recommendation tasks have been sufficiently explored, we classify sequential recommendation algorithms in terms of the three tasks (Section 2.2): *experience-based sequential recommendation*, *transaction-based sequential recommendation*, and *interaction-based sequential recommendation.*

3.1 Experience-based Sequential Recommendation

As we have introduced, in an experience-based behavior sequence, a user interacts with a same item with different behavior types. The goal of experience-based sequential recommendation is to predict the next behavior type that the user will implement on the item, and thus it is also referred to as *multi-behavior recommendation.* Accordingly, we first explore the studies on multi-behavior recommendation and then present DL-based models that leverage multi-behavior information in the sequential recommendation.

3.1.1 Conventional models for multi-behavior recommendation. Ajit el al. [91] first proposed a collective matrix factorization model (CMF) to simultaneously factorize multiple user-item interaction matrices (in terms of different behavior types) by sharing the item-side latent matrix (item embedding) across matrices. Other studies [48, 152] extended CMF to handle different user behaviors (e.g., social relationships). Besides, there are also some models addressing multi-behavior recommendation with Bayesian learning. For example, Loni et al. [62] proposed multi-channel BPR to adapt the sampling rule for different behavior types. Qiu et al. [76] further proposed an adaptive sampling method for BPR by considering the co-occurrence of multiple behavior types. Guo et al. [28] aimed to resolve the data sparsity problem by sampling unobserved items as positive items based on item-item similarity, which is calculated by multiple behavior types. Ding et al. [23] developed a margin-based learning framework to model the pairwise ranking relations among purchase, view, and non-view behaviors.

3.1.2 DL-based multi-behavior recommendation. DL techniques have also been applied in multi-behavior recommendation. For example, *NMTR* [26] is proposed to tackle some representative problems of conventional models for multi-behavior recommendation, e.g., lack of behavior semantics, unreasonable embedding learning and incapability in modeling complicated interactions. To capture the sequential relationships between behavior types, NMTR [26] cascades predictions of different behavior types by considering the sequential dependency relationship among different behaviors in practice[6], which thus translates the heterogeneous behavior problem into the experience-based sequential recommendation problem as we have defined. It should be noted

[6]For example, the *search*, *click*, and *purchase* operations for the same item are usually sequentially ordered in e-commerce.

that this cascaded prediction, which could be regarded as pre-training embedding layers of other behavior types before learning a recommendation model for the target behavior, only considers the connections between target behavior and previous behaviors but ignores the ones between target behavior and subsequent behaviors. Thus, it does not fully explore the relationship on various behavior types. In this view, multi-task learning (MTL) can address this problem by providing a paradigm to predict multiple tasks simultaneously which also exploits similarities and differences across tasks. The performance of the MTL model proposed in [26] is generally better than those using the sequential training. Besides, Xia et al. [133] proposed a multi-task model with LSTM to explicitly model users' purchase decision process by predicting the stage and decision of a user at a specific time with the assistance of a pre-defined set of heuristic rules, and thus obtaining more accurate recommendation results.

3.2 Transaction-based Sequential Recommendation

In transaction-based sequential recommendation, there is only a single behavior type (transaction-related, e.g., purchase), and recommendation models generally consider the sequential dependency relationships between different objects (items) as well as user preferences. As there are a substantial amount of DL-based models for this task, we further summarize the existing models in terms of the employed specific DL techniques.

3.2.1 RNN-based Models. RNN structures have been well exploited in transaction-based sequential recommendation task, and we summarize RNN-based approaches from the following perspectives.

(1) GRU4Rec-related models. Hidasi et al. [34] proposed a GRU-based RNN model for sequential recommendation (i.e., *GRU4Rec*), which is the first model that applies RNN to sequential recommendation, and does not consider a user's identity (i.e., anonymous user). On its basis, a set of improved models [12, 33, 103] have been proposed, which also use RNN architectures for modeling behavior sequence. The architecture of GRU4Rec is shown in Figure 9. As introduced in [34], the input of GRU4Rec is a session (behavior sequence), which could be a single item, or a set of items appeared in the session. It uses one-hot encoding to represent the current item, or a weighted sum of encoding to represent the set of items. The core of the model is the GRU layer(s), where the output of each layer is the input for the next layer, but each layer can also be connected to a deeper non-adjacent GRU layer in the network. Feedforward layers are added between the last GRU layer and the output layer. The output is the probability of each candidate item that will appear in the next behavior.

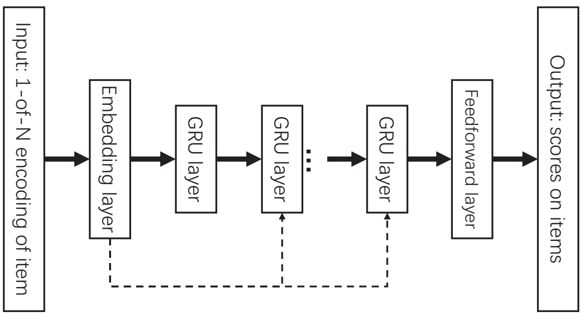


Fig. 9. Architecture of GRU4Rec.

GRU4Rec employs *session-parallel mini-batches* and *popularity-based negative sampling* for training. The reason for using session-parallel mini-batches is to form sessions with equal length while the length of actual sessions can be greatly varied. On the other hand, if simply breaking a session into different parts to force them into equal length, we could not well model the behavior sequence and fail to capture how a session evolves over time [34].

The extended studies strive to improve the model performance from the perspectives of model training and designing more advanced model structures for better learning item information. For example, for **facilitating training**, [103] applied *data augmentation* to enhance training of GRU4Rec. [12] considered the *dwell time* to modify the generation of mini-batch, which has been verified to greatly improve performance. On the other hand, popularity-based sampling suffers from the problem that model learning slows down after all the candidate items have been ranked above popular ones, which could be a relatively serious problem for long-tail items recommendation. Thus, [33] proposed the *additional sampling* (a combination of uniform sampling and popularity sampling) for negative sampling in GRU4Rec, which can enormously improve performance.

For **better modeling item information**, [35] considered additional item information other than IDs (e.g., text descriptions and images) for improving prediction performance. Specifically, they introduced a number of parallel RNN (p-RNN) architectures to model sessions based on click behaviors and other features of the clicked items (e.g., pictures and text descriptions). Moreover, they particularly proposed alternative but more suitable training strategies for p-RNNs: *simultaneous*, *alternating*, *residual*, and *interleaving* training. In simultaneous training (baseline), every parameter of each subnet is trained simultaneously. In alternating training, subnets are trained in an alternating fashion per epoch. In residual training, subnets are trained one by one by the residual error of the ensemble of the previously trained subnets, while interleaving training is alternating training per mini-batch. Furthermore, [42] combined the session-based KNNs with GRU4Rec using the methods of *switching*, *cascading*, and *weighted hybrid*.

(2) With user representation. There are also some studies that aim to better model users' preference. For example, [149] proposed a RNN-based framework for click-through rate (CTR) prediction in sponsor search, which considers the impact of the click dwell time with the assumption that the longer a user stays on an ad page, the more attractive the ad is for the user. In total, three categories of features are considered: ad features (ad ID, position and query text), user features (user ID, user's query) and sequential features (time interval, dwell time and click sequence). [94] took one-hot encoding of items in users' behavior sequences as input of a GRU-based RNN to learn users' historical embeddings. *RRN* [129] is the first recurrent recommender network that attempts to capture the dynamics of both user and item representation. [11] further improved the RRN's interpretability by devising a time-varying neighborhood style explanation scheme, which jointly optimizes prediction accuracy and interpretability of the sequential recommendation.

Considering that simply embedding a user's historical information into a single vector may lose the per-item or feature-level correlation information between a user's historical sequences and long-term preference, Chen et al. [15] thus proposed a memory-augmented neural network for the sequential recommendation. The model explicitly stores and updates every user's historical information by leveraging an external memory matrix. Huang et al. [40] further improved [15] by adopting a separate GRU component for capturing sequential dependency and incorporated knowledge base (KB) information for better learning attribute (feature)-level user preference. Towards better modeling lifelong sequential patterns for each user, Ren et al. [80] proposed a Hierarchical Periodic Memory Network (*HPMN*) to capture multi-scale sequential patterns, where periodic memory updating mechanism is designed to avoid unexpected knowledge drifting and hierarchical memory slots are used to deal with different update periods.

HRNN [78][7] uses GRU to model users and sessions respectively. The session-level GRU considers a user's activities within a session and thus generates recommendations, while the user-level GRU models the evolution of a user's preference across sessions. Given that the length of sessions of different users are varied, it deploys user parallel mini-batch training, which is extended from session parallel mini-batch of GRU4Rec. Donkers et al. [24] further proposed a user-based GRU framework (including linear user-based GRU, rectified linear user-based GRU, and attentional user-based GRU) to integrate user information for giving better user representations. *HierTCN* [138] also involves a GRU-based high-level model to aggregate users' evolving long-term preferences across different sessions, while *SDM* [65] particularly designs a gated fusion module to effectively integrate users' short-term and long-term preferences.

(3) Context-aware sequential recommendation. Most of the previous models have ignored the huge amount of context information in real-word scenarios. In this view, [59] summarized two types of contexts: *input contexts* and *transition contexts*. Input contexts refer to the ones by which users conduct their behaviors, e.g., location, time and weather, whilst transition contexts mean the transitions between two adjacent input elements in historical sequences (e.g., time intervals between the adjacent behaviors). It further designed the context-aware recurrent neural networks (*CA-RNN*) to simultaneously model the sequential and contextual information. Besides, [96] proposed *ARNN* to consider the user-side contexts, e.g., age, gender and location. Specifically, *ARNN* extracts high-order user-contextual preferences using a product-based neural network, which is capable of being incorporated with any existing RNN-based sequential recommendation models.

(4) Other models. Other than the aforementioned three categories, there are other RNN-based models (e.g., *DREAM* [139]) for transaction-based sequential recommendation in the literature. For example, [22] used RNN for the collaborative filtering task and considered two different objective functions in the RNN model: categorical cross-entropy (*CCE*) and *Hinge*, where CCE has been widely used in language modeling, and Hinge is extended from the objective function of SVMs. [85] deployed a multi-layer GRU network to capture sequential dependencies and user interest from both the inter-session and intra-session levels. In view of that existing studies assume that there is only an implicit purpose for users in a session, Wang et al. [122] proposed a mixture-channel purpose routing networks (MCPRNs) to capture the possible multi-purposes of users in a session (a channel implies a latent purpose). MCPRNS consists of a purpose router (PRN) and a multi-channel recurrent framework with purpose-specific recurrent units.

3.2.2 CNN-based Models. RNN models are limited to model relatively short sequences due to their network structures and relatively expensive computing costs, which can be partially alleviated by CNN models [89]. For example, *3D-CNN* [107] designs an embedding matrix to concatenate the embedding of item ID, name, and category. *Caser* [105] views the embedding matrix of L previous items as an 'image', and thus uses a horizontal convolutional layer and a vertical convolutional layer to capture point-level and union-level sequential patterns respectively. Using convolution, the perception of relevant skip behaviors becomes possible. It also captures long-term user preferences through user embedding. The network structure of *CNN-Rec* [38] is highly similar to Caser in terms of user embedding and horizontal convolution, but it does not deploy vertical convolution. *NextItNet* [142] is a generative CNN model with the residual block structure for the sequential recommendation. It is capable of capturing both long and short-term item dependencies. *GRec* [141] further extends *NextItNet* by utilizing a gap-filling based encoder-decoder framework with masked-convolution operations to jointly consider the past and future contexts (data) without the data leakage issue. In *HierTCN* [138], a low-level model implemented with Temporal Convolutional

[7] github.com/mquad/hgru4rec.

Networks (TCN) unitizes the long-term user preference learned from GRU module and the short-term user preference within a session to generate the final recommendation.

3.2.3 Attention-based Models. The attention mechanisms have been largely applied to the sequential recommendation, and are capable of identifying more 'relevant' items to a user given the user's historical experience. We conclude these models according to the deployed attention mechanism types: *vanilla attention* and *self-attention* (see Section 2.3.2).

(1) Vanilla attention mechanisms. *NARM*[8] [53] is an encoder-decoder framework for transaction-based sequential recommendation. In the local encoder, RNN is combined with vanilla attention to capture the major purposes (or interest) of a user in the current sequence. With the attention mechanism, NARM is able to eliminate noises from unintended behaviors, such as accidental (unintended) clicks. [120] applied the vanilla attention mechanism to weight each item in a sequence to reduce the negative impact of unintended interactions. Liu et al. [61] proposed a short-term attention/memory priority model, which uses vanilla attention to calculate attention scores of items in a sequence as well as the attention correlations between previous items and the most recent item in the sequence. Ren et al. [81] considered the repeated consumption issue, and thus proposed *RepeatNet* which evaluates the recommendations from both the repeat mode and the explore mode, which refer to the old item from a user's history and the new item, respectively. [86] incorporated vanilla attention with a Bi-GRU network to model user's short-term interest for music recommendation. [5] proposed a unified attribute-aware neural attentive model (*ANAM*), which applies vanilla attention mechanism on feature level. To better capture users' short-term preferences, Yu et al. [140] designed a multi-order attention network which is instantiated with two k-layer residual networks to model individual-level and union-level item dependencies respectively.

(2) Self-attention mechanisms. Self-attention mechanisms have also obtained increasing interest in the sequential recommendation. For example, Zhang et al. [146] utilized the self-attention mechanism to infer the item-item relationship from the user's historical interactions. With self-attention, it is capable of estimating weights of each item in the user's interaction trajectories to learn more accurate representations of the user's short-term intention, while it uses a metric learning framework to learn the user's long-term interest. In *SDM* [65], a multi-head self-attention module is incorporated to capture a user multiple interests in a session (i.e., short-term preference), while long-term preference of the user is also encoded through attention and dense fully connected networks based on various types of side information, e.g., item ID, first level category, leaf category, brand and shop in the user's historical transactions. Similarly, *SASRec* [45] adopts a self-attention layer to balance short-term intent and long-term preference, and seeks to identify items relevant to the next behavior from the user's historical behavior sequences.

BERT4Rec [99] is the improved version of SASRec, which introduces the transformer architecture for the sequential recommendation and trains the bidirectional model to model sequential data using Cloze task. *TiSASRec* [54] further improves SASRec by taking time intervals between items in a sequence into consideration. Specifically, it models a relation matrix between items for each user according to the time interval information between each two items in the historical sequences. Besides, to overcome the drawbacks of RNN-based sequential recommender systems, such as not supporting parallelism and only modeling one-way transitions between consecutive items, *SANSR* [100] incorporates the transformer framework [110] to speed up the training process and learn the relations between items in the session regardless the distance and the direction.

As we know, most of the previous studies on the sequential recommendation focus on recommendation accuracy, but ignore the *diversity* of recommendation results, which is also a quite important measurement for effective recommendation. With respect to this issue, Chen et al. [14]

[8]github.com/lijingsdu/sessionRec_NARM.

proposed an intent-aware sequential recommendation algorithm, which uses the self-attention mechanism to model a user's multi-intents in a given session.

3.2.4 Other Models. There are also some other DL-structures (e.g., MLP [118], GNN [126] and autoencoder) that have been adopted in the sequential recommendation. For example, *NN-rec* [112] is the first work considering neural network for next-basket recommendation, which is inspired by NLPM [9]. Wu et al. [131] used GNN for session-based recommendation to capture complex transitions among items. In this model, each session is modeled as a directed graph, and proceeded by a gated graph neural network to obtain session representations (local session embedding and global session embedding). *GACOforRec* [144] utilizes graph convolutional neural networks to learn the item order within a session as well as the spatiality within the network to handle a users' short-term intents, while it designs ConvLSTM to capture the user's long-term preference. Besides, considering that different behaviors may have different impacts, it proposes a new pair of attention mechanisms which consider the different propagation distances in the graph convolutional network to obtain the different weights. *UGrec* [117] models user and item interactions as a graph network, defines sequential recommendation paths from users' purchased histories, and further aggregates different paths using an attention mechanism. Finally, a particularly designed translation learning objective function in graph embedding is designed for model learning and inference.

Sachdeva et al. [87] explored the variational autoencoder for modeling a user's preference through his/her historical sequence, which combines latent variables with temporal dependencies for preference modeling. Ma at al. [66] specifically designed a hierarchical gating network (*HGN*) with BPR to capture both the long-term and short-term user preferences.

3.3 Interaction-based Sequential Recommendation

Compared to the aforementioned two tasks, the interaction-based one is much more complicated as each behavior sequence consists of both different behavior types and different behavior objects. Thus, the recommendation models are expected to capture both the sequential dependencies among different behaviors, different items as well as behaviors and items, respectively. Next, we summarize the related models according to the deployed DL techniques.

3.3.1 RNN-based Models. RNN-based models still take the majority role in this task [63]. For example, [108] proposed a RNN-based model without explicitly learning user representation. Given the task of predicting the next item expected to appear in terms of a target behavior type, Le et al. [49] firstly divided a session into a target sequence and a supporting sequence according to the target behavior type. Its basis idea is that the target behavior type (e.g., purchase) contains the most efficient information for the prediction task, and the remaining behaviors (e.g., click) can thus be utilized as the supporting sequences that can facilitate the next-item prediction task for the target behavior type. Besides, in order to better model the dependencies among different behavior types, some studies would also assume that there is a cascading relationship (as in Section 3.1.2) among different types of behaviors (i.e., different behavior types are sequentially ordered). For example, Li et al. [56] proposed a model that consists of two main components: *neural item embedding* [7] and *discriminative behavior learning*. For behavior learning, it utilizes all types of behavior (e.g., click, purchase and collect) to capture a user's present consumption motivation. Meanwhile, it selects purchase related behaviors (e.g, purchase, collect and add-to-cart) from user's historical experience to model the user's underlying long-term preference. Considering that RNN cannot well handle users' short-term intent in a sequence whereas log-bilinear model (LBL) cannot capture users' long-term preference, Liu et al. [60] combined RNN with LBL to construct two models (*RLBL* and *TA-RLBL*) for modeling multi-behavioral sequences. TA-RLBL is an extension of RLBL [60] which considers the continuous time difference information between input behavior objects and thus further improve the performance of RLBL. [93] took context information (e.g., behavior type) into consideration by modifying the structures of RNN.

3.3.2 Other Models. There are also some other DL techniques applied in the interaction-based sequential recommendation, including attention mechanisms, MLPs and Graph-based models. For example, [153] proposed *ATRank*[9] which adopts both *self-attention* and *vanilla attention* mechanisms. Considering the heterogeneity of behaviors, ATRank models the influence among behaviors via self-attention, while it uses vanilla attention to model the impact of different behaviors on the recommendation task. *CSAN* [41] is the improved version of ATRank by also considering side information and polysemy of behavior types. Wu et al. [128] proposed a deep listNet ranking framework (MLP-based) to jointly consider user's clicks and views. Ma et al. [67] proposed a graph-based broad-aware network (*G-BBAN*) for news recommendation, which considers multiple user behaviors, behavioral sequence representations, and user representation.

Table 1. Categories of representative algorithms regarding sequential recommendation tasks and DL models.

Task	DL Model	Papers
Experienced-based	MLP	[26]
	RNN	[133]
Transaction-based	RNN	[12, 15, 24, 34, 59, 78, 96, 103, 139] [14, 22, 33, 35, 40, 42, 85, 100, 149] [11, 65, 80, 94, 129, 130, 138]
	CNN	[38, 105, 107, 138, 141, 142]
	MLP	[112, 118]
	Attention mechanism	[5, 45, 53, 61, 81, 86, 120, 137] [14, 54, 65, 99, 130, 140, 146]
	GNN	[117, 126, 131, 144]
	Other networks	[66, 87]
Interaction-based	RNN	[49, 56, 60, 63, 93, 108]
	MLP	[128]
	Attention mechanism	[41, 63, 153]
	GNN	[67]

3.4 Concluding Remarks

In this section, we have introduced representative algorithms for the three sequential recommendation tasks. We list the representative algorithms in terms of tasks and DL techniques in Table 1. In summary, RNNs and attention mechanisms have been greatly explored in both transaction and interaction-based sequential recommendation tasks, where the effectiveness of other DL models (e.g., GNN and generative models) needs much further investigation. Besides, there are also some issues for the existing models especially for the complicated interaction-based sequential recommendation: (1) the behavior type and the item in a behavior 2-tuple (c_i, o_i) are mostly equally treated. For example, ATRank [153] and CSAN [41] adopt the same attention score for the item and the corresponding behavior type; (2) different behavior types are not distinguished successfully. For example, [108] used the same network to model different types of behaviors, assuming that different behavior types have similar patterns; (3) the correlation between behaviors in a sequence

[9] github.com/jinze1994/ATRank.

is easily ignored. For example, [128] used pooling operation to model multi-type behavior in a sequence. In view of these issues, more advanced approaches are needed for much more effective sequential recommendation, especially for the task of interaction-based sequential recommendation. In the next two sections, we will further summarize and evaluate the factors that might impact the performance of a DL-based model in regard to recommendation accuracy, which are expected to better guide future research.

4 INFLUENTIAL FACTORS ON DL-BASED MODELS

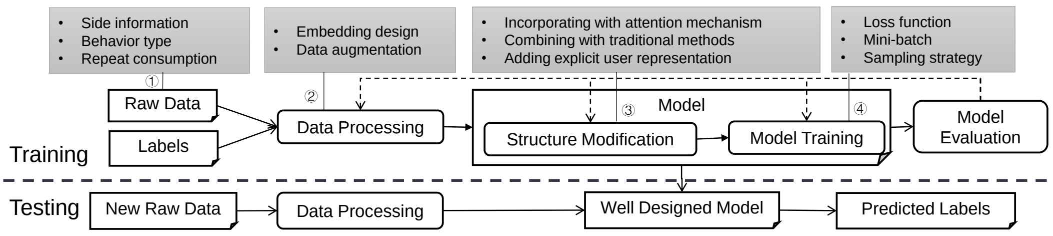


Fig. 10. Influential factors of DL-based models.

Figure 10 shows the *training* and *testing* process of a sequential recommender system. In the training, the input includes raw data and label information, which are then fed into the data processing module, mainly including *feature extraction* and *data augmentation*. Feature extraction refers to converting raw data into structured data, while data augmentation is normally used to deal with data sparsity and cold-start problems, especially in DL-based models. Third, a model is trained and evaluated based on the processed data, and the model structure or training method (e.g., learning rate, loss function) can be updated in an iterated way based on the evaluation results till satisfactory performance is reached. In the testing, the data processing module only includes feature extraction, and then the obtained trained model is used to make recommendations.

On the basis of a thorough literature study, we identify some representative factors (listed in *grey* boxes in Figure 10 and Table 2) that might impact the performance of DL-based models in terms of recommendation accuracy. The details of these factors are discussed subsequently.

4.1 Input Module

Side information and *behavior types* are critical factors to DL-based models in the input module.

4.1.1 Side Information. Side information has been well recognized to be effective in facilitating recommendation performance [101]. It refers to information about items (other than IDs), e.g., *category*, *images*, *text descriptions*, and reviews, or information related to transactions (behaviors) like *dwell time*. Text and image information about items have been widely explored in DL-based collaborative filtering systems [6, 18, 73, 79, 145], as well as in some DL-based sequential recommender systems [35, 41]. For example, p-RNN [35] uses a parallel RNNs framework to process the item IDs, images and texts. Specifically, the first parallel architecture trains a GRU network (i.e., subnet) for item representation on the basis of each kind of information, respectively. The model concatenates the hidden layers of the subnets and generates the output. The second architecture has a shared hidden state to output weight matrix. The weighted sum of the hidden states is used to produce the output instead of being computed by separate subnets. In the third structure called parallel interaction, the hidden state of the item feature subnet is multiplied by the hidden state of the ID subnet in an element-wise manner before generating the final outcome. CSAN [41] utilizes

Table 2. The influential factors on DL-based sequential recommender systems.

Module	**Factor**	**Method**	**Papers**
Input	side information	utilize image/text	[35, 41]
		utilize dwell time	[12, 20, 149]
	behavior type	simple behavior embedding	[153]
		divide session into groups for different purposes	[49, 56]
	repeat consumption	consider repeat behavior	[39, 81, 111]
Data processing	embedding design	item embedding	[27]
		w-item2vec	[56]
		session embedding	[128]
	data augmentation		[103, 107]
Model structure	incorporating attention mechanism	only attention mechanism	[5, 41, 146, 153]
		incorporating vanilla attention mechanism with other DL methods	[45, 53, 61, 120]
	combining with conventional methods	KNN	[42]
		metric learning	[146]
	adding explicit user representation	user embedded models	[105, 118]
		user recurrent models	[11, 15, 24, 56, 78, 80, 129]
Model training	negative sampling	uniform	[33, 34]
		popularity-based	[33, 34]
		additional	[33]
		sample size	[33]
	mini-batch creation	session parallel	[34]
		item boosting	[12]
		user parallel	[78]
	loss function	TOP1	[34]
		TOP1-max & BPR-max	[33]
		CCE & Hinge	[22]

word2vec and CNN to learn the representation of texts and images respectively. Previous models have demonstrated that side information like item images and texts can alleviate the data sparsity [35, 55, 116] and cold-start [41, 115, 127, 153] problems.

On the other hand, side information like *dwell time* partially imply a user's degrees of interest on different items. For example, when a user browses a web page for an item, the longer he/she stays, we can infer that the more he/she is interested in. Bogina et al. [12] applied item boosting according to the dwell time for generating mini-batch in training. In particular, assuming that a predefined threshold of dwell time is t_d seconds, if the dwell time on an object i in a session is within the range of $[2t_d, 3t_d)$, then the parallel mini-batch of this session will contain 2 repeated behaviors regarding to i, i.e., the presence of i in the session increases. This strategy (referred as *item boosting*) can be considered as to re-measure the importance of behavior objects in terms of the corresponding dwell time. Zhang et al. [149] treated dwell time as a sequence feature, and concatenated it with other features (e.g., query text). Similarly, Dallmann et al. [20] proposed an extension to existing RNN approaches by adding user dwell time. Experiments in [12] show that incorporating dwell time with GRU4Rec [34] makes a great improvement (up to 153.1% on MRR@20).

4.1.2 Behavior Type. In the sequential recommendation, behaviors in user behavior sequences are usually heterogeneous and polysemous [41, 153], and different behavior types imply users' different intents. For instance, a purchase action is a better indicator of a user's preference on an item than a click behavior. Therefore, it is critical to treat different behavior types differently [26, 49, 108, 153]. For example, CBS [49] divides a sequence into target sequence and supporting one in terms of behavior types, where the target sequence is related to the behavior type (e.g., purchase) that has the most efficient information for prediction. Similarly, BINN [56] utilizes all behavior types (e.g., click, purchase and collect) to capture a user's present interest whereas models the user's long-term preference using only purchase related information (e.g, purchase, add-to-cart and collect). [153] learns the representation of each behavior type and then concatenate them with the corresponding item embedding vectors. Experiments generally support that purchase behavior can more accurately capture a user's long-term preferences, whilst other behavior types can facilitate the learning of short-term interests [26, 49, 108, 153].

4.1.3 Repeat Consumption. Repeat consumption refers to that an item is repeatedly appeared in a user's historical sequences, which is mostly ignored in the sequential recommendation. Anderson et al. [2] investigated the dynamics of repeated consumption on seven real datasets and found that recency is the strongest predictor of repeated consumption. Bhagat et al. [10] presented four models (i.e., repeated customer probability model, aggregate time distribution model, Possion-Gamma model, and modified Possion-Gamma model) for repeat purchase recommendations. [113] further combined collaborative filtering and Hawkes Process to build a holistic model for recommendation, and the item-specific temporal dynamics of repeat consumption are captured. For sequential recommendation, Wan et al. [111] used *loyalty* factor to model repeated consumption which can further boost the performance of next-basket recommendation. Ren et al. [81][10] also considered this issue, and their results confirm that the consideration of repeat consumption patterns in DL network design can improve the recommendation performance. The recent study of [39] just pointed out that RNN-based models might not well capture repeated behaviors for next-basket recommendation, and thus proposed a KNN-based model for capturing repeated consumption in next-basket recommendation.

It should be noted that, although side information and behavior types could greatly improve model performance, their collections might be either infeasible or cost-consuming.

4.2 Data Processing

An appropriate design of feature extraction methods (i.e., *embedding design*) and *data augmentation* for generating more training data have been validated to be effective in existing DL-based models.

4.2.1 Embedding Design. In the sequential recommendation, embedding methods are used to represent information about an item, a user, or a session. For example, Greenstein et al. [27] adopted the word embedding methods GloVe [75] and Word2Vec [70] (CBOW) for *item embedding* in e-commerce applications. Li et al. [56] further proposed w-item2vec (inspired by item2vec [7])

[10]github.com/PengjieRen/RepeatNet.

on the basis of the Skip-gram model and thus formed unified representations of items. Wu et al. [128] designed a session embedding for pre-training by considering different user search behaviors such as clicks and views, the target item embedding and the user embedding together to have a comprehensive session understanding (i.e., session representation).

4.2.2 Data Augmentation. In the sequential recommendation, in some scenarios there might be no user profiles or historical information for a new user, or a user who does not log in, i.e., cold-start problems. Therefore, data augmentation becomes an important technique. For example, Tan et al. [103] proposed an augmentation method, where prefixes of the original input sessions are treated as new training sequences as shown in Figure 11. That is, given the original session (l_1, l_2, l_3, l_4), we can generate 3 training sequences: (l_1, l_2), (l_1, l_2, l_3), (l_1, l_2, l_3, l_4), while a recommendation algorithm predicts the last item for each training sequence. With this method, a session is repeatedly utilized during training, which is demonstrated to improve 14.7% over GRU4Rec on MRR@20 [103]. Besides, the *dropout* method [98] is further adopted to prevent the over-fitting problem (see Figure 11, a circle with the dotted line is the dropout behavior in each sequence). Besides, considering that items after a target item may also contain valuable information, these items are thus viewed as privileged information as [109] to facilitate the learning process.

Similarly, with regard to two behavior types (i.e., add-to-cart and click), 3D-CNN [107] also uses *data augmentation* which treats all prefixes up to the last add-to-cart item as training sequences for each session containing at least one add-to-cart item. Besides, it uses *right padding* or *simple dropping* methods to keep all the sequences of the same length.

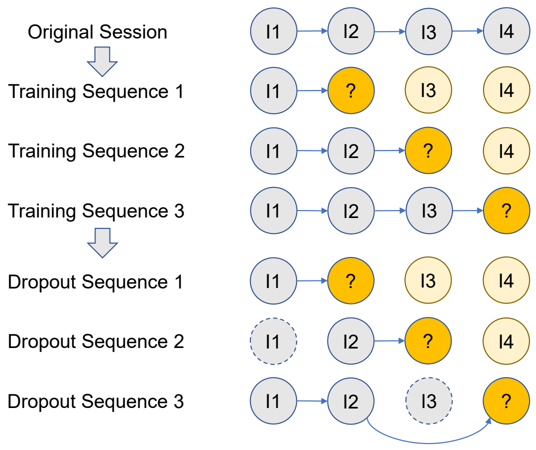


Fig. 11. Data augmentation. The orange circles represent the predicted items; the dotted circles represent the item that is deleted in the dropout method, and light orange circles make up privileged information.

4.3 Model Structure

We summarize the major methods to improve model structures in the previous DL-based models as: *incorporating attention mechanisms*, *combining with conventional models*, and *adding explicit user representation.*

4.3.1 Incorporating Attention Mechanisms. In Section 2.3.2, we discuss that there are mainly *vanilla attention* and *self-attention*. Overall, we can incorporate attention mechanism with other DL models, or just build attention models to address the sequential recommendation problems. For the first scenario, NARM [53], ATEM [120] and STAMP [61] incorporate the *vanilla attention mechanism* with RNN or MLP, aiming to capture user's main purpose in a given session. Experiments verify that their performance surpassed GRU4Rec by 25%, 92% and 30% respectively. SASRec [45] combines self-attention with feedforward network to model correlations between different behaviors, and can improve recommendation accuracy by 47.7% and 4.5% on HR@10 compared with GRU4Rec and Caser [105], respectively. For the second scenario, for example, AttRec [146] simply deploys a self-attention mechanism to capture users' short-term interest, where its performance exceeds Caser by 8.5% on HR@50. ATRank [153] and CSAN [41] combine self-attention with vanilla attention for the sequential recommendation. Attention mechanisms can be further employed to capture attribute-level importance level of items for modeling users' interest. For example, ANAM [5] applies attention mechanism to track a user's appetite for items and their attributes.

To conclude, previous studies demonstrate that incorporating attention mechanisms can improve recommendation of the DL-based models, while mostly only using self-attention mechanisms can have better performance than some DL-based models without attention mechanisms.

4.3.2 Combining with Conventional Methods. DL-based models can also be combined with traditional methods to boost their performance on the sequential recommendation tasks. For example, [42] combined a session-based KNN with GRU4Rec [34] in three different ways (i.e., switching, cascading, and weighted hybrid), showing that the best combination can exceed original GRU4Rec by 9.8% in some applications. AttRec [146] combines self-attention (for short-term interest learning) and metric learning (for long-term preference modeling), and the performance exceeds Caser[105] by 8.5% on HR@50.

4.3.3 Adding Explicit User Representation. Given the application scenarios where users' IDs can be recognized, we can design methods for explicitly learning user representation, i.e., users' long-term preferences can be well modeled by *user embedded models* or *user recurrent models.*

User embedded models. This fold of models explicitly learns user representations [105, 118] via embedding methods, but not in a recurrent process as item representation. They can facilitate the performance of the sequential recommendation models [105]. However, such models might suffer from the cold-start user problem since the long-term interest of a user with little historical information cannot be well learned. Another issue is that, user representation via user embedded models is learned in a relatively static way, which cannot capture users evolved and dynamic preferences. In this view, user recurrent models are expected to be more effective, which learn user representations in a recurrent way as item representation learning.

User recurrent models. They treat both user and item representations as recurrent components in the DL-based models, which can better capture users' evolving preferences, including memory-augmented neural network [15, 80], RNN-based models [24, 56, 78] and recurrent neural networks [11, 129]. For example, [24, 56, 78] used RNN framework to learn users' long-term interest from their historical behavior sequences. Experiments verify that considering a user's long-term interest is critically valuable for personalized recommendation, e.g., HRNN [78] exceeds GRU4Rec by 3.5% with explicit user representation in some scenarios.

In summary, model structures play an important role in the sequential recommendation, where better designs can help more effectively capture the sequential dependencies among items and behaviors, and thus better understand both users' short-term and long-term preferences.

4.4 Model Training

Well-designed training strategies can also facilitate the learning of DL-based sequential recommendation models. With a comprehensive investigation, we have summarized three major strategies: *negative sampling*, *mini-batch creation* and *loss function.*

4.4.1 Negative Sampling. *Popularity-based sampling* and *uniform sampling* have been widely used in recommendation. Popularity-based sampling assumes that the more popular an item is, the more possibly that a user knows about it. That is to say, if a user does not interact with it previously, it is more likely that the user dislikes it. [33] further proposed a novel sampling strategy (called *additional sampling*) by combining these two sampling strategies, which takes the advantages but overcomes the shortcomings of both strategies in negative sampling. In the additional sampling strategy, negative samples are selected with a probability proportional to supp_i^{α}, where supp_i is the support of item i and α is a parameter ($0 \leq \alpha \leq 1$). The cases of $\alpha = 0$ and $\alpha = 1$ are equivalent to uniform and popularity-based sampling respectively. Experiment results show that additional sampling can surpass both the popularity-based sampling and uniform sampling methods under certain scenarios (e.g., loss functions). Besides, *the size of negative samples* can also affect the performance of sequential recommendation models.

4.4.2 Mini-batch Creation. *Session parallel mini-batch training* [34] was proposed to accommodate sessions of varied lengths and strive to capture the dynamics of sessions over time. In particular, sessions are firstly arranged in time order. Then, the first event (behavior) of the first X sessions (X is the number of sessions) is used to form the input of the first mini-batch (whose desired output is the second event of the active sessions). The second mini-batch is formed from the second event of the X sessions, and so on and so forth. If any of the X sessions reaches its ending, the next available session out of the X sessions is placed in the corresponding place to continually form the mini-batch. Session parallel mini-batch has two variants: *item boosting* and *user-parallel mini-batch*. In item boosting, some items can be repeatedly used in mini-batch in terms of identified factors like the dwell time [12], while regarding the latter variant, for example, HRNN [78] designs user-parallel mini-batch (i.e., parallel sessions belong to different users) to model the evolution of users' preferences across sessions.

4.4.3 Loss Function Design. Loss functions can also greatly impact the model performance. In the sequential recommendation, quite a few loss functions have been employed, including *TOP1-max* (ranking-max version of *TOP1*), *BPR-max* (ranking-max version of *BPR*), *CCE* (Categorical Cross-Entropy) and *Hinge*.

TOP1 is a regularized approximation of relative rankings of positive and negative samples. As shown in Equation 2, it consists of two parts: the first part inclines to penalize the incorrect ranking between positive sample i and any negative sample j (N_S is the size of negative samples), and the second part is used as the regularization.

$$L_{\text{TOP1}} = \frac{1}{N_S} \sum_{j=1}^{N_S} \sigma(r_j - r_i) + \sigma(r_j^2) \tag{2}$$

where $\sigma(.)$ is a sigmoid function, r_i and r_j are the ranking scores for sample i and j respectively. Following the same notations, **BPR** (Bayesian Personalized Ranking) [82] is defined as:

$$L_{\text{BPR}} = -\frac{1}{N_S} \sum_{j=1}^{N_S} \log \sigma(r_i - r_j) \tag{3}$$

TOP1 and BPR loss functions might suffer from the gradients vanishing problems for DL-based models (e.g., in GRU4Rec [33]). In this view, ranking-max loss function family is proposed [33] to address this issue, where the ranking score is only compared to the negative sample which is most relevant to the target sample, i.e., the one has the highest ranking score. Accordingly, we have **TOP1-max** and **BPR-max**, which are formulated as Equations 4 and 5 respectively. They can be considered as the weighted version of TOP1 and BPR, respectively. Previous research validates that the two loss functions largely improve the performance of RNN-based sequential recommendation models [33].

$$L_{\text{TOP1-max}} = \sum_{j=1}^{N_S} s_j \left(\sigma(r_j - r_i) + \sigma(r_j^2) \right) \tag{4}$$

where s_j is the normalized score of r_j using softmax function.

$$L_{\text{BPR-max}} = -\log \sum_{j=1}^{N_S} s_j \sigma(r_i - r_j) \tag{5}$$

In addition to the ranking-based loss functions, *CCE* (categorial cross-entropy) and *Hinge* loss functions have also been applied in the sequential recommendation [22]. **CCE** is defined as:

$$\text{CCE}(\mathbf{o}, i) = -\log(\text{softmax}(\mathbf{o})_i) \tag{6}$$

where $\mathbf{o}$ is a model output and i is a target item. CCE suffers from the computation complexity issue due to the softmax function. On the contrary, **Hinge** compares the predicted results with a pre-defined threshold (e.g., 0):

$$\text{Hinge}(\mathbf{o}, i) = \sum_{j \in C} \max(0, 1 - o_j) + \gamma \sum_{j \in F} \max(0, o_j) \tag{7}$$

where C is the set of recommendations containing item i, while F is the set of recommendations not containing i (i.e., bad recommendations). γ is a parameter to balance the impacts of the two parts of errors (correctly recommended vs. incorrectly recommended). With Hinge loss, the recommendation task is transformed to a binary classification problem where a recommender system determines whether an item should be recommended or not.

5 EMPIRICAL STUDIES ON INFLUENTIAL FACTORS

Here, we conduct experiments[11] on real datasets to showcase the impact of influential factors on DL-based models in terms of recommendation accuracy, where mostly the ways of incorporating influential factors are widely adopted by representative sequential recommender systems.

5.1 Experimental Settings

5.1.1 Datasets. We use three real-world datasets: *RSC15*, *RSC19* and *LastFM*. *RSC15* is published by RecSys Challenge 2015[12], which contains click and buy behaviors from an online shop. Only the click data is used in our evaluations. *RSC19* is published by RecSys Challenge 2019[13], which contains hotel search sessions from a global hotel platform. *RSC19 (user)* is a subset of *RSC19*. *LastFM* is collected via the LastFM API, and each sample is a 4-tuple (user, artist, song, timestamp).

Following the common way in data pre-processing [34, 78], for *RSC15* and *RSC19*, we firstly filter out sessions which have less than 2 behaviors, and items that appear less than 5 times. Then we consider the sessions that end in the last day as the test set, while the others are for model training/validation. For *RSC19 (user)*, we further select users with more than 10 sessions, and consider the last session of each user as the test set. For *LastFM*, due to the lack of session identities in LastFM, we manually divide the behavior sequence of each user into sessions every 30 minutes. Then we filter out sessions which have less than 3 behaviors, items that appear less than 5 times, and users that appear less than 3 times. We also consider the last session of each user as the test set. Here, we want to emphasize that different ways of data filtering, which lead to different data scenarios,

[11]The source codes and datasets of the experiments are shared on Github: https://github.com/sttich/dl-recommendation.
[12]www.kaggle.com/chadgostopp/recsys-challenge-2015.
[13]www.recsyschallenge.com/2019/.

will result in varied performance. For example, we check the performance of GRU4Rec on different data scenarios by filtering out sessions less than {2, 3, 4, 5, 10, 15, 20} on *RSC15* respectively. For fair comparisons, under all data scenarios, we use the same test set as data scenario of 20 following the aforementioned data pre-process procedure. We have tuned the hyperparameters under each scenario, and the results are presented in Figure 12. As shown in Figure 12, the performance of GRU4Rec drops as the length of sequence gets longer. This might be caused by the decreasing number of sessions for training, i.e., the available training sessions on the RSC dataset are 7966888, 4419603, 2810308, 1876772, 448561, 167318, and 78486 under the seven data scenarios, respectively.

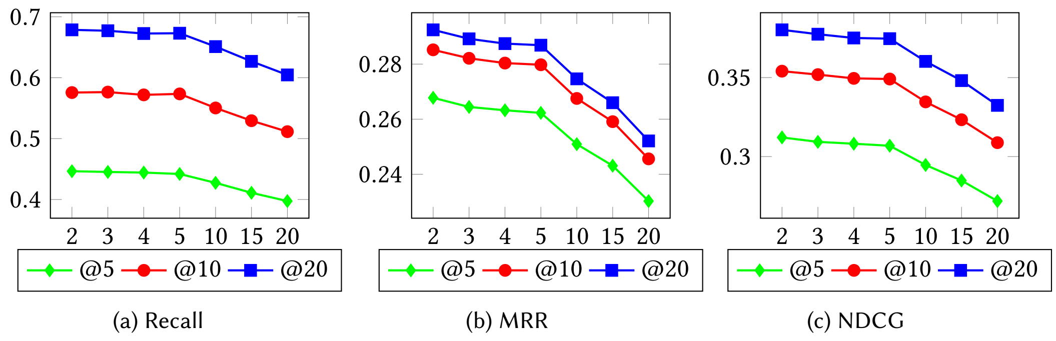


Fig. 12. The effect of the length of session on RSC15.

Besides, it should be noted that few studies have explicitly discussed their data splitting methods for model training/validation/testing. By reading the source codes publicized by the corresponding authors, we summarize the following two major forms (which are differed mainly owing to that whether user information is considered or not in model design): (1) to divide the sessions that have occurred in the latest *n* days as the test set; (2) to treat each user's latest session as the test set. The former one is much more commonly adopted in the sequential recommendation.

The statistic information of these datasets are summarized in Table 3.

Table 3. The statistic information of the four datasets.

Feature	RSC15	RSC19	RSC19 (user)	LastFM
Sessions	7,981,581	356,318	1,885	23,230
Items	37,483	151,039	3,992	122,816
Behaviors	31,708,461	3,452,695	49,747	683,907
Users	–	279,915	144	277
ABS	3.97	9.69	26.39	29.44
ASU	–	1.27	13.09	83.86

ABS: Average Behaviors per Session
ASU: Average Sessions per User

5.1.2 Model Settings. We choose GRU4Rec [34] (Figure 9) as our *basic* model, and then consider the influential factors in Figure 10 to check their effects on the basic model. The main reason of using GRU4Rec is that lots of algorithms in the literature make improvement on it, or recognize it as a representative and competitive baseline for the sequential recommendation tasks. This makes GRU4Rec as a perfect fit to showcase the effects of influential factors on DL-based model. *Specifically,*

Table 4. Other parameters settings for different scenarios.

Model	RSC15			
	Batch Size	Lr	RNN Size	dropout rate
Default	32	0.2	100	0
GRU4Rec (Category)	50	0.001	100	0.5
C-GRU	50	0.001	120	0.5
P-GRU	50	0.001	100 (item), 20 (category)	0.5
NARM	512	0.001	100	0.25
Model	**RSC19**			
	Batch Size	Lr	RNN Size	dropout rate
Default	32	0.2	100	0
GRU4Rec (Behavior)	50	0.001	100	0.5
B-GRU	50	0.001	100	0.5
NARM	512	0.001	100	0.25
User Implicit	50	0.001	50	0.5
User Embedded	50	0.001	50	0.5
User Recurrent	50	0.01	100 (item), 100 (user)	0
Model	**LastFM**			
	Batch Size	Lr	RNN Size	dropout rate
User Implicit	50	0.001	50	0.5
User Embedded	50	0.001	50	0.5
User Recurrent	200	0.02	50 (item), 50 (user)	0

in our experiments, we focus on the widely explored transaction-based sequential recommendation task, which aims to predict the next item a user will like/purchase on the basis of transaction-based sequences. In the future, we can consider other representative models using different DL structures, e.g., *NextItNet* [142] (CNN-based model) and *NARM* [53] (attention-based model).

The *default* parameters for the **basic** model is no data augmentation, no user representation (i.e., implicit user representation), BPR-max loss function, uniform negative sampling with a sample size of 128 for *RSC15* and *RSC19*. In the next experiments, if not being particularly figured out, other models also use these default settings.

For the input module, we choose two kinds of **side information**: *item category* and *dwell time*. For the item category, following the previous studies, we implement two improved versions of the basic model: **C-GRU** (concatenating item embedding with category embedding [18]) and **P-GRU** (parellelly training two basic models for item and category respectively, and then concatenating the output of the two subnets [35]) with mini-batch parallel negative sampling (batch size = 50). The corresponding control model **GRU4Rec (category)** is the basic **GRU4Rec** model with the same setup as **C-GRU** and **P-GRU** except for the RNN size. For the dwell time, we implement the model in [12], and according to the distribution of the dwell time, we choose 75 and 100 seconds as thresholds for *RSC15*, and 45 and 60 seconds for *RSC19*.

To verify the impact of **behavior types**, we design a new network (**B-GRU**) by adding a behavior type embedding module to the basic model. Specifically, **B-GRU** takes both the item one-hot vectors and behavior type one-hot vectors as input and converts them into embedding vectors, where item embedding vectors are fed into a GRU model, whose output is concatenated with behavior type embedding vectors for MLP layers. The current design aims to capture the intuition that a user's next behavior in the sequence is not only related to item sequence that the user has previously interacted with, but also might be impacted by the user¡¯s previous behavior type. Besides, **B-GRU** uses mini-batch parallel negative sampling method with a sample size of 50. The respective control model **GRU4Rec (Behavior)** is the basic **GRU4Rec** model with the same setup as **B-GRU**. The structures of **C-GRU**, **P-GRU** and **B-GRU** are shown in Figure 13.

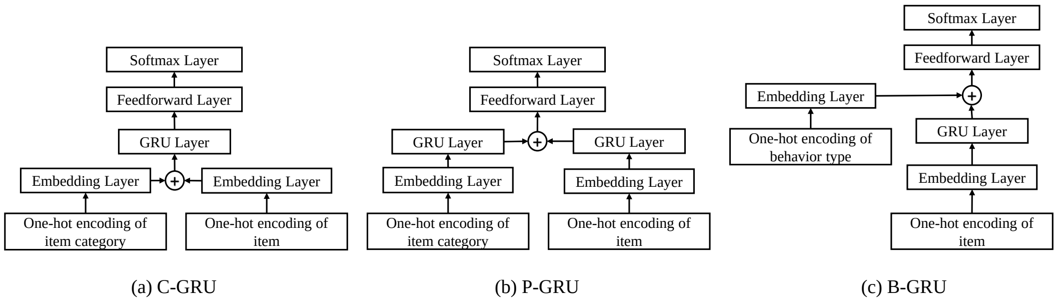


Fig. 13. The network structures of C-GRU, P-GRU and B-GRU.

For the data processing module, we implement the **data augmentation** method in [103] (see Figure 11) on the basic model. Specifically, we randomly select 50% sessions in the training set to conduct data augmentation, and randomly treat a part of each session as new sessions.

For the model structure module, we consider three structures: **NARM** [53] (*incorporating the basic model with the attention mechanism*), **weighted model** in [42] (*combining the DL model with KNN*) and **adding an explicit user representation** in two ways, i.e., the recurrent way in [78] (referred as *User Recurrent*), and the embedded way by adding a user embedding layer based on user IDs (referred as *User Embedded*), which is concatenated with the output of GRU in the basic GRU4Rec model, and uses the same training method as in [78]). Noted that we use user-parallel mini-batches [78] in training for both user recurrent and embedded models.

For the model training module, we consider three factors: **loss function** (i.e., cross-entropy, BPR-max, BPR, TOP1-max, and TOP1), **sampling method** (i.e., additional sampling in [33]), and $\alpha \in \{0, 0.25, 0.5, 0.75, 1\}$, and the size of negative samples $\{0, 1, 32, 128, 512, 2, 048\}$. Other parameters in terms of different datasets are summarized in Table 4, where **Lr** refers to learning rate of these DL-based models. It should be noted that we use the default hyper-parameters (e.g., batch size, learning rate, and RNN size) recommended in the source codes for the basic models (control models). Besides, for demonstrating the effectiveness of the corresponding factors, in the experimental models (with factor effects), we fix the major default hyper-parameters to be the same as the corresponding control models. With this kind of setting, we aim to effectively demonstrate the impact of the influential factors (as well as the designed components) in the sequential recommendation, while maximally eliminate the impact of other factors.

5.1.3 Evaluation Metrics. To compare the performance of different models, we use three widely used accuracy metrics: **Recall@k**, **MRR@k** (Mean Reciprocal Rank) and **NDCG@k** (Normalized Discounted Cumulative Gain) as previous sequential recommendation models, where k is set to 5, 10 and 20 respectively. For these three metrics, a larger value implies better performance. We refer interesting readers to [146] for detailed definitions of Recall and MRR evaluation metrics. Noted that GRU4Rec is to predict a user's next behavior (*next-item prediction*), i.e., only one item in the recommendation list will be actually selected by the user. In this case, MRR is equivalent to Mean Average Precision (MAP), and Recall is identical to Hit Ratio (HR) [99].

- **Recall**@k: it measures the coverage of the corrected recommended items in terms of ground-truth items.
- **MRR**@k: it refers to how well a model ranks the ground-truth items.
- **NDCG**@k: it rewards each ground-truth item based on its position in the recommendation list, indicating how strongly an item is recommended.

$$\text{NDCG@}k = \frac{1}{\log_2(rank + 1)}$$

where $rank$ denotes the ranking position of the ground-truth item.

It should be noted that for each method, we run each experiment 5 times and report the performance in the form of "mean ± std deviation" in terms of the three metrics as in Tables 5 and 6, and show the mean values in other Figures.

5.2 Experiment Results

Table 5. Results of incorporating item category or behavior type. Statistical significance of pairwise differences of each improved model vs. basic model (GRU4Rec) is determined by a paired t-test (* for p-value ≤ 0.1, ◇ for p-value ≤ 0.05, Δ for p-value ≤ 0.01).

Model	RSC15					
	Recall@5	MRR@5	NDCG@5	Recall@20	MRR@20	NDCG@20
GRU4Rec	0.313±0.0047	0.168±0.0036	0.203±0.0039	0.554±0.0043	0.192±0.0034	0.273±0.0035
C-GRU	0.328^{Δ}± 0.0023	0.178^{Δ}± 0.0016	0.215^{Δ}± 0.0015	0.564^{Δ}± 0.0032	0.202^{Δ}± 0.0015	0.283^{Δ}± 0.0012
P-GRU	$\mathbf{0.335}^{\Delta}$±0.0014	$\mathbf{0.180}^{\Delta}$± 0.0017	$\mathbf{0.218}^{\Delta}$±0.0014	$\mathbf{0.570}^{\Delta}$±0.0018	$\mathbf{0.204}^{\Delta}$± 0.0017	$\mathbf{0.286}^{\Delta}$±0.0015

Model	RSC19					
	Recall@5	MRR@5	NDCG@5	Recall@20	MRR@20	NDCG@20
GRU4Rec	0.568± 0.0065	0.489±0.001	0.509± 0.0017	0.696± 0.0023	0.502±0.0019	0.546± 0.0015
B-GRU	$\mathbf{0.586}^{\Delta}$± 0.0025	$\mathbf{0.494}^{\Delta}$± 0.0032	$\mathbf{0.517}^{\Delta}$±0.0030	$\mathbf{0.708}^{\Delta}$± 0.0008	$\mathbf{0.507}^{\Delta}$± 0.0029	$\mathbf{0.552}^{\Delta}$±0.0023

Here, we systematically present the experimental results of different influential factors in terms of the four modules.

5.2.1 Input Module. First, we present the experimental results regarding the factors related to the input module: side information and behavior types.

Side information effects. Tables 5 and 6 show the results of the two types of the side information on DL-based model respectively[14]. As shown in Table 5, incorporating **item category** information into GRU4Rec can improve the model performance in terms of all the three metrics. Specifically, C-GRU and P-GRU perform better than the basic model. As we can see in Table 6, **dwell time** can greatly improve the performance, e.g., Recall@20 increases by about 28% and 19% on *RSC15* and *RSC19* datasets respectively. To conclude, utilizing the side information can significantly improve the model performance, and the way in which the side information is incorporated also matters. Thus, it is necessary to have a calibrated design by considering the impact of side information on the final prediction.

Behavior type effects. The results regarding impact of behavior types (*B-GRU*) are present in Table 5. We can see that B-GRU outperforms the basic model in terms of all metrics. If a dataset provides

[14]We have consistent results when $k = 10$. Due to the space limitation, we do not report it here, but on Github: https://github.com/sttich/dl-recommendation.

Table 6. Results on considering different factors. Statistical significance of pairwise differences of each improved model vs. the basic *GRU4Rec* is determined by a paired t-test (* for p-value ≤ 0.1, $\diamond$ for p-value ≤ 0.05, and Δ for p-value ≤ 0.01).

Factor	Variable	RSC15					
		Recall@5	MRR@5	NDCG@5	Recall@20	MRR@20	NDCG@20
Dwell time	0[4]	0.446±0.0011	0.268±0.0007	0.312±0.0008	0.676±0.0009	0.293±0.0006	0.380±0.0006
	75	**0.772**$^{\Delta}$±0.0004	**0.692**$^{\Delta}$± 0.0005	**0.712**$^{\Delta}$±0.0005	**0.865**$^{\Delta}$±0.0005	**0.702**$^{\Delta}$± 0.0005	**0.739**$^{\Delta}$±0.0004
	100	0.730$^{\Delta}$±0.0010	0.635$^{\Delta}$±0.0007	0.659$^{\Delta}$±00.0007	0.841$^{\Delta}$±0.0004	0.647$^{\Delta}$±0.0006	0.691$^{\Delta}$±0.0005
Data Aug[1]	Off[4]	0.446±0.0011	**0.268**±0.0007	**0.312**±0.0008	0.676±0.0009	**0.293**±0.0006	**0.380**±0.0006
	On	**0.446**±0.0018	0.267±0.0005	0.312±0.0007	**0.678**$^{\diamond}$ ±0.0011	0.292±0.0005	0.379±0.0005
Att[2]	Off[5]	0.480±0.0005	0.285±0.0005	0.334± 0.0005	0.703±0.0001	0.309±0.0005	0.400± 0.0004
	On	**0.486**$^{\Delta}$±0.0003	**0.290**$^{\Delta}$±0.0002	**0.339**$^{\Delta}$± 0.0002	**0.708**$^{\Delta}$±0.0002	**0.314**$^{\Delta}$±0.0003	**0.404**$^{\Delta}$± 0.0002
KNN weight	0[4]	0.446±0.0011	0.268±0.0007	0.312±0.0008	0.676±0.0009	0.293±0.0006	0.380±0.0006
	0.1	0.452$^{\Delta}$±0.0008	0.270$^{\Delta}$±0.0003	0.315$^{\Delta}$±0.0002	0.693$^{\Delta}$±0.0006	0.296$^{\Delta}$±0.0005	0.386$^{\Delta}$±0.0004
	0.3	**0.460**$^{\Delta}$±0.0007	**0.278**$^{\Delta}$±0.0004	**0.323**$^{\Delta}$±0.0004	**0.698**$^{\Delta}$±0.0009	**0.303**$^{\Delta}$±0.0005	**0.393**$^{\Delta}$±0.0003
Factor	Variable	RSC19					
		Recall@5	MRR@5	NDCG@5	Recall@20	MRR@20	NDCG@20
Dwell time	0[4]	0.640±0.0018	0.547±0.0028	0.571±0.0025	0.751±0.0017	0.559±0.0028	0.602±0.0025
	45	**0.845**$^{\Delta}$±0.0074	**0.783**$^{\Delta}$±0.0063	**0.799**$^{\Delta}$±0.0065	**0.893**$^{\Delta}$±0.0071	**0.788**$^{\Delta}$±0.0062	**0.813**$^{\Delta}$±0.0063
	60	0.830±0.0007	0.763$^{\Delta}$±0.0013	0.780$^{\Delta}$±0.0011	0.885±0.0010	0.768$^{\Delta}$±0.0015	0.795$^{\Delta}$±0.0013
Data Aug[1]	Off[4]	0.640±0.0018	0.547±0.0028	0.571±0.0025	0.751±0.0017	0.559±0.0028	0.602±0.0025
	On	**0.641** ±0.0015	**0.551**$^{\diamond}$ ±0.0019	**0.574**$^{\diamond}$ ±0.0013	**0.754**$^{\diamond}$ ±0.0004	**0.562**$^{\diamond}$ ±0.0020	**0.606**$^{\diamond}$ ±0.0014
Att[2]	Off[5]	0.736±0.0015	0.569±0.0010	0.611±0.0011	0.905±0.0009	0.587±0.0001	0.661±0.0001
	On	**0.742**$^{\Delta}$±0.0023	**0.572**$^{\Delta}$± 0.0024	**0.615**$^{\Delta}$±0.0023	**0.912**$^{\Delta}$±0.0012	**0.591**$^{\Delta}$± 0.0022	**0.665**$^{\Delta}$±0.0020
KNN weight	0[4]	0.640±0.0018	0.547±0.0028	0.571±0.0025	0.751±0.0017	0.559±0.0028	0.602±0.0025
	0.1	0.643±0.0039	0.549±0.0064	0.572±0.0057	0.753±0.0034	0.560±0.0063	0.604±0.0055
	0.3	**0.657**$^{\Delta}$±0.0023	**0.562**$^{\Delta}$±0.0067	**0.586**$^{\Delta}$±0.0057	**0.765**$^{\Delta}$±0.0033	**0.573**±0.0067	**0.617**±0.0057
Factor	Variable	LastFM					
		Recall@5	MRR@5	NDCG@5	Recall@20	MRR@20	NDCG@20
User Rep[3]	Implicit[6]	**0.173**$^{\Delta}$±0.0021	**0.147**$^{\Delta}$±0.0018	**0.154**$^{\Delta}$±0.0018	**0.193**$^{\Delta}$±0.0024	**0.149**$^{\Delta}$±0.0018	**0.160**$^{\Delta}$±0.0018
	Embedded	0.006±0.0011	0.004±0.0016	0.004±0.0014	0.016±0.0026	0.005±0.0015	0.007±0.0012
	Recurrent	0.002±0.0002	0.001±0.0002	0.002±0.0007	0.003±0.0009	0.002±0.0005	0.002±0.0002
Factor	Variable	RSC19 (user)					
		Recall@5	MRR@5	NDCG@5	Recall@20	MRR@20	NDCG@20
User Rep[3]	Implicit[6]	**0.713**$^{\Delta}$±0.0165	**0.654**$^{\Delta}$±0.0128	**0.668**$^{\Delta}$±0.0130	**0.781**$^{\Delta}$±0.0134	**0.661**$^{\Delta}$±0.0128	**0.689**$^{\Delta}$±0.0125
	Embedded	0.032±0.0098	0.023±0.0092	0.025±0.0093	0.060±0.0113	0.025±0.0090	0.032±0.0094
	Recurrent	0.030±0.0199	0.016±0.0113	0.019±0.0131	0.079±0.0198	0.020±0.0111	0.033±0.0126

[1] "Data Aug" refers to Data Augmentation.
[2] "Att" refers to Attention Mechanism.
[3] "User Rep" refers to User Representation.
[4] "0" and "Off" scenarios refer to that the corresponding influential factor is not considered, i.e., being equivalent to the basic *GRU4Rec*.
[5] "Off" here denotes NARM [53] without attention mechanism.
[6] "Implicit" denotes no user representation.

the **behavior type** information, it is better to integrate it into the final model with appropriately designed modules, e.g., the simple module as ours.

5.2.2 Data Processing. Here, we check the impact in regard to the data augmentation.

Data augmentation effects. As shown in Table 6, the model with data augmentation performs slightly better (0.2%) than the basic model on Recall@20 for *RSC15*, but worse in terms of MRR@20 and NDCG@20. On *RSC19*, data augmentation improves the model performance in terms of all metrics but with a lower significance level (i.e., 5%). Similar mixed results can be observed when $k = 5$ (see Table 6). To conclude, simple data augmentation cannot significantly enhance the model performance for **GRU4Rec** model. We might consider to design more complex ways according to the characteristics of **GRU4Rec** model.

5.2.3 Model Structure. In this subsection, we present the experimental results with respect to varying the DL structures.

Incorporating attention mechanism effects. As shown in Table 6, incorporating **attention mechanism** enhances the performance of the model almost for all the scenarios.

Combining with conventional method effects. Combing the basic model with **KNN** improves model performance in terms of all metrics on both *RSC15* and *RSC19*, and KNN weight of 0.3 provides better performance than that of 0.1, manifesting that the way of combining traditional models with DL-models can have a significant effect on the sequential recommendation.

User representation effects. *Implicit* represents the basic GRU4Rec model with session-parallel mini-batch method. *Recurrent* and *Embedded* refer to adding explicit user representation in two different ways as discussed in Section 5.1.2. For user representation, we find that adding an explicit user representation module, whether embedded or recurrent one, leads to a sharp decrease on all metrics. For a complementary investigation, we tuned the major hyperparameters (e.g., batch size, learning rate, and RNN size, etc) for user recurrent and user embedded models, and found that the large gap between these two models and the basic model cannot be significantly decreased. The main reasons might be three-folds: 1) session-parallel mini-batch (a session as a sample) is used for the implicit model while user-parallel mini-batch (a user's all historical sessions as a sample) is deployed for user embedded and recurrent models. In this case, training samples for user embedded and recurrent models are much fewer than the implicit model (since we did not find a specific model which adds user embedding in *GRU4Rec* directly, we also refer to the training method in [78] for training the user embedded model); 2) as shown in Table 3, the number of sessions is much greater than that of users on these two datasets; 3) according to the number of items and behaviors shown in Table 3, the average support of items is much smaller on these two datasets. It is worth mentioning that as reported in [78], GRU4REC with recurrent user representation performs better than the original model on their two datasets. Furthermore, we can see that the user embedded model outperforms user recurrent model in most scenarios, but performs worse than recurrent model in terms of Recall@20 and NDCG@20 on *RSC19 (user)*. In this case, in the personalized sequential recommendation, we can infer that whether selects the user embedded model or recurrent model largely depends on the characteristics of datasets and application scenarios. However, whether considering explicit user representation model is dependent on not only the application scenarios, but also a well-designed user representation component.

5.2.4 Model Training. Here, we present the experiments results from the perspectives of three factors: sampling methods, sample size, and loss functions.

Sampling method effects. Figures 14 and 15 depict the model performance with different α for **additional sampling strategy** on *RSC15* and *RSC19* respectively, where the results on different datasets are varied. For *RSC15*, the performance of cross-entropy, BPR-max and TOP1-max (on all metrics) are consistent. They firstly slowly increases as α increases from 0 to 0.25, And then decreases as α is larger than 0.25. Besides, the optimal *alpha* on *RSC15* corresponding to different loss functions and metrics are the same. On the contrary, on *RSC19*, the optimal α is varied for different loss functions and different evaluation metrics. For example, the optimal α for BPR-max loss function is 0.5 in terms of Recall@20, but for cross-entropy that is 0. In terms of MRR@20 and NDCG@20, the optimal α for BPR-max loss function is 0.75. Therefore, it is necessary to carry out

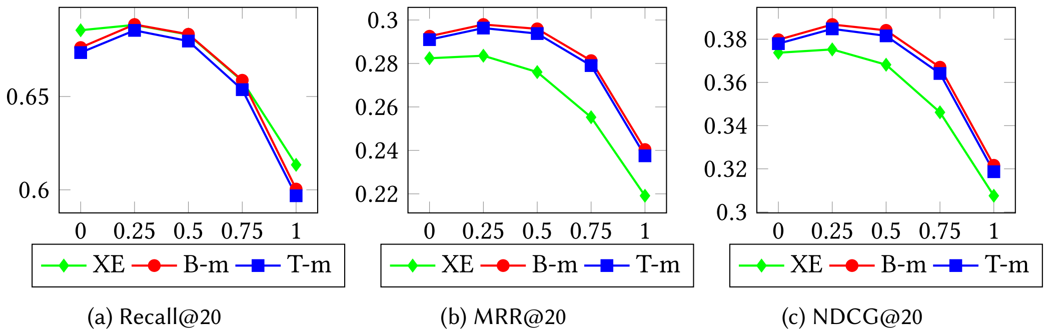


Fig. 14. The impact of α on additional sampling strategy on RSC15 (XE: cross-entropy; B-m: BPR-max; T-m: TOP1-max).

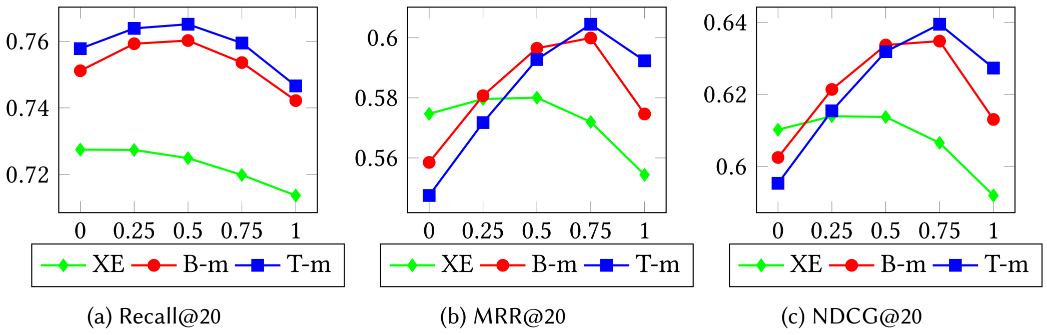


Fig. 15. The impact of α on additional sampling strategy on RSC19 (XE: cross-entropy; B-m: BPR-max; T-m: TOP1-max).

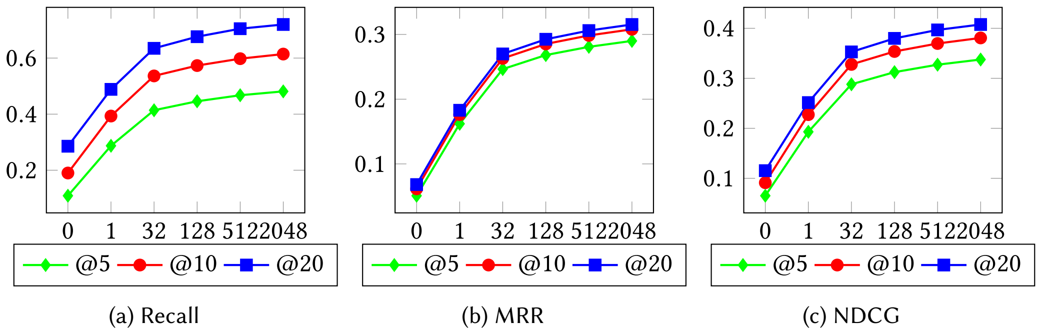
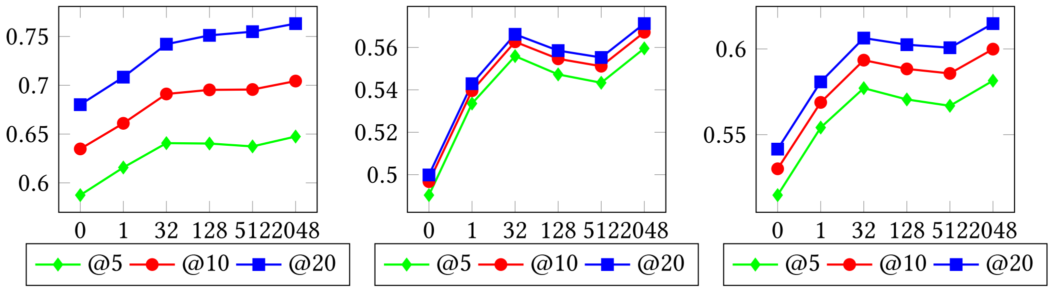


Fig. 16. The effect of sample size on RSC15.

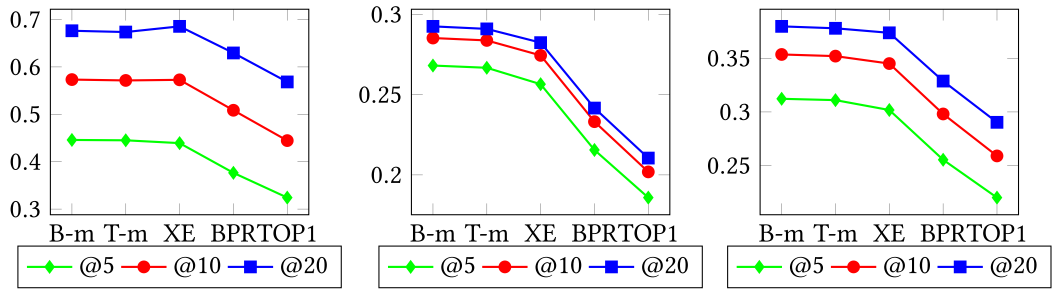
sufficient search in validation set to figure out the optimal combination of sampling strategy and loss function with regard to the most valuable evaluation measurements in real world applications.

The size of negative sampling effects. As described in Figure 16 the larger the **size of negative sample** is, the better performance the basic model can obtain regarding all evaluation measurements. In particular, the model performance improves dramatically when the size increases from 0 to 32, while the increasing speed drops with the further increase of the size. Empirical results on *RSC19*



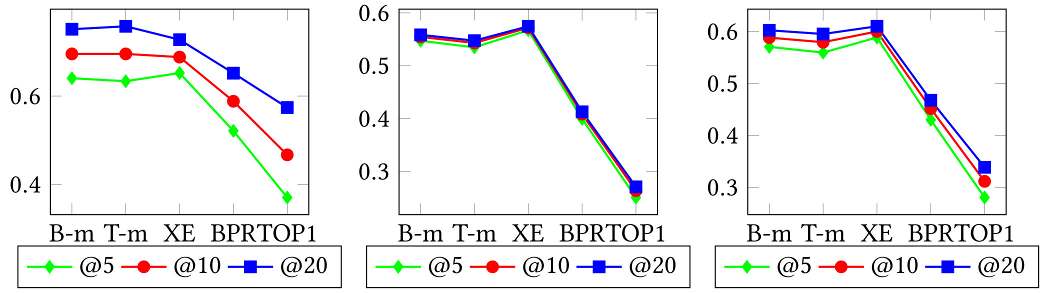
(a) Recall (b) MRR (c) NDCG

Fig. 17. The effect of sample size on RSC19.



(a) Recall (b) MRR (c) NDCG

Fig. 18. Model performance for different loss functions on RSC15 (B-m: BPR-max; T-m: Top1-max; XE: cross-entropy).



(a) Recall (b) MRR (c) NDCG

Fig. 19. Model performance for different loss functions on RSC19 (B-m: BPR-max; T-m: Top1-max; XE: cross-entropy).

(Figure 17) is almost similar, but there is a slight decrease when sample size varies from 32 to 128 in terms of MRR and NDCG. It should be noted that additional negative sampling leads to higher computational costs. Therefore, in real world applications, we need to keep a balance between model performance and training time considering the size of negative samples.

Loss function effects. As depicted in Figures 18 and 19, models with **loss functions** BPR-max, TOP1-max, and cross-entropy perform better than those with BPR and TOP1 in terms of all metrics, but the best loss function among the three is also dependent on the datasets. Overall, it is suggested to deploy these three loss functions in real-world applications.

5.2.5 Concluding Remarks. The experimental results verify that the summarized influential factors in Section 4 all play an important role in DL-based sequential recommendation. Our suggestions for best in practice are summarized as follows: 1) try all possible side information (such as texts and images) when allowed, and carefully design the corresponding modules; 2) well consider the connections between other behavior types with the target behavior, and be careful about the possible noisy information involved in final recommendation when model these connections; 3) always incorporate TOP1-max, BPR-max and cross-entropy loss functions for training, keep a balance between model performance and computational cost with regard to the size of negative samples, and carefully choose/design data augmentation strategies to further boost the corresponding recommendation performance, especially when the available training set is relatively small; and 4) for any DL-based models, consider to further improve their performance with attention mechanism, by possibly combing with the traditional sequential learning models, and a well-designed explicit user representation module accommodating to the corresponding data scenarios.

6 FUTURE DIRECTIONS AND CONCLUSIONS

6.1 Future Directions

As we have discussed, DL techniques have greatly promoted the sequential recommendation studies, but also are accompanied by some challenging issues. Thus, we summarize the following open issues which can be considered as future directions for DL-based sequential recommender systems.

6.1.1 Rigorous and Comprehensive Evaluations across Different Models. Our empirical study can be viewed as a horizontal investigation across *GRU4Rec* and its variants. In the literature, lots of *GRU4Rec* variants consider *GRU4Rec* [34] for baseline, but there are few comparisons among these variants. In this case, it is difficult to judge which one is better in a specific application scenario. Besides, other competitive baselines could also be considered, e.g., *NextItNet* [142] (CNN-based model), *NARM* [53] (attention-based model) or further refer to the comparison framework proposed in [64]. There is a growing consensus that only complex deep learning structures could not always guarantee better and more robust recommender systems [19, 102]. Besides, one critical issue has attracted increasing attention in the field of recommender systems: there are few effective benchmarks for evaluation, especially in the sequential recommendation. Thus, benchmarking study for rigorous and comprehensive evaluations should be urgently put on the research agenda.

6.1.2 Explainable Sequential Recommender Systems. As has pointed out, most DL-based models are lack of explanability and considered as black-box for platform practitioners and users. As such, designing explainable sequential recommender systems is of great importance. On the one hand, without understanding the reasons behind predictions, users will be reluctant to trust an individual prediction sufficiently and thus take actions based on it. On the other hand, model practitioners strive to fully understand the model (i.e., how the different factors like data, features and model hyper-parameters impact the model outputs?) and thus enhance their control towards the whole recommender system. Noted that our survey sheds light on the second issue, and more ideas can be borrowed from those explanability studies towards deep learning networks [3, 46, 72, 88]

For the first issue, there are mainly two categories of methods for explainable recommendation which tries to make users understand why such items or lists are recommended: model-based and post-hoc ones [148]. Model-based methods aim to design interpretable algorithms to simultaneously provide accurate and explainable recommendation items for users (mostly using a multi-task learning framework). Besides user-item interactions, they usually incorporate side information (e.g., textual and visual item descriptions, textual reviews, social information) to facilitate the explainable task. For example, Huang et al. [40] leveraged knowledge graph for better explanability in the sequential recommendation. In contrast, without directly linking explanations with side information for recommendations, the post-hoc methods [74] try to devise separate methods to explain the recommendation results produced by those black-box based recommendation algorithms.

6.1.3 Better Designs on Different Components to Facilitate the Recommendation. On the basis of the empirical results, we consider that more efforts can be devoted to the following components, which can be incorporated into every sequential recommendation model to improve the recommendation performance correspondingly.

More designs on embedding methods. Most previous studies adopt the embedding methods from NLP. However, in the sequential recommendation, it is rather challenging to pre-train an embedding model (e.g., word2vec) as the item information and the dependencies relationship among items is constantly changing while the words and their connections in NLP are relatively fixed and static. On the other hand, behaviors in the sequential recommendation are also very complex than words as they involve both behavior objects and types. Furthermore, the incorporation of embedding vectors in existing sequential recommendation models are also in a relatively simple way. In this case, more advanced and particular designs of embedding methods are needed for the sequential recommendation [111]. For example, [58] designed a sequential embedding approach which also takes the dependencies relationships among items and their attributes into consideration for the sequential recommendation. A possible solution is to take advantage of metric learning [106] for obtaining better item, user or sequence representations by understanding sequential data. Metric learning focuses on distance metrics that capture the important relationships among data [50], and has been verified to be effective in traditional static recommendation tasks [37, 134].

Better modeling user long-term preference. On the basis of our study and empirical investigation, the module in DL-based models for user representation (especially the long-term preference) is still far from satisfactory, compared to the designed modules for item representation. In this case, further research can consider to design more favorable modules for user representation, as well as think about how to better combine a user's long-term preference with short-term preference.

Advanced sampling strategies. In the sequential recommendation, most existing studies use the sampling strategies of uniform, popularity-based, or their straightforward combination (i.e., additional sampling), which are comparatively simple contrasting with the ones used in NLP. In this view, future research could consider to borrow or extend more advanced sampling strategies from other areas (e.g., NLP or graph representation [124, 136]).

6.1.4 Personalized Recommendation Based on Polymorphic Behavior Trajectory. We summarize behavior sequences into three types, and to the best of our knowledge, there is relatively few studies that well distinguish the behavior types and model their connections in the sequential recommendation for interaction-based sequential recommendation tasks. Our empirical evaluation also indicates that well considering another behavior type for a target type is very challenging. In this case, more DL-models can be designed by considering the connections between polymorphic behavior types and thus for better recommendation performance in the sequential recommendation. For example, Qiu et al. [76] proposed a Bayesian personalized ranking model for heterogeneous behavior types (BPRH) that incorporates the target behavior, auxiliary behavior, and negative behavior into a unified model, and the idea might also be applicable for the sequential recommendation. Besides, more advanced deep learning models, such as GNNs for heterogeneous networks [52] can be considered to fulfill the goal. For example, Song et al. [95] proposed a dynamic-graph-attention neural network to capture dynamic user preferences in sequences and social influence (social network) for better recommendation.

6.1.5 Learning Behavior Sequences in Real Time. Every behavior of the user might reflect a possible interest transfer, in this case, recommender systems are expected to ideally capture this kind of information and timely justify the recommendation strategies. Reinforcement learning is a promising choice for addressing this issue. For example, Zhao et al. [151] combined MF, RNN, and GAN in film recommendations to dynamically provide movie recommendations. Shih et al. [90] treated the generation of music playlists as a language modeling problem and used an attention-based language model with the policy gradient in reinforcement learning.

Besides, another important issue for DL-based models is their scalability, where models should be capable of dealing with increasing amount of data. For example, in order to speed up retraining procedure, [150] designed a new training method (a.k.a. sequential meta-learning method) which aims to abandon the historical data through learning to transfer the past training experience. Furthermore, being validated to be effective and efficient in image recognition [36, 135], *knowledge distillation* techniques have also started to be introduced in recommender systems [114]. For example, Tang and Wang [114] proposed ranking distillation (RD) method by incorporating knowledge distillation technique for learning to rank problem. Experimental results demonstrated that the student model using less than half of the model parameters could achieve similar or better ranking performance than the teacher model. Chen et al. [16] formulated the problem of recommendation with external knowledge (e.g., online reviews) into a *generalized distillation framework* which simultaneously balances both the effectiveness and efficiency of recommendation model .

6.1.6 Sequential Recommendation for Specific Domains and Cross-Domains. There are little research to specifically identify the suitable recommendation algorithms for different application areas, whereas most research assumes that their models are applicable to the sequential recommendation tasks in all areas. Future research can be conducted to design specific models for particular areas by capturing the characteristics of these areas, which is more valuable for real-world applications.

On the other side, more and more large companies incline to provide services/products in different domains. For example, ByteDance (bytedance.com) simultaneously offers news service and video service to users. Compared to single-domain recommendation, cross-domain recommendation addresses the problem of leveraging data in different domains to generate desirable recommendation [13], where the main underlying idea is to transfer knowledge from source domains to a target domain, thus further boost the recommendation performance in the target domain. Following this idea, as deep learning is a good fit to transfer knowledge across different domains [147], future research can consider to well investigate the characteristics of sequential data in different domains, and design more advanced cross-domain sequential recommendation models. For example, Zhuang et al. [155] exploited the sequential behavioral data of one user from different domains to mine her/his novelty-seeking traits for improving recommendation performance. Ma et al. [68] further formulated the cross-domain sequential recommendation problem as a parallel sequential recommendation problem. By considering that user behaviors on two domains are synchronously shared at each timestamp, they proposed π-Net (i.e., consisting of a shared account filter unit and a cross-domain transfer unit) to simultaneously generate recommendation for two domains.

6.1.7 Towards Robust Sequential Recommendation Models. It is a common sense that a single model cannot guarantee to perform consistently well in different application scenarios, as the data distributions vary. In other words, a model might be relatively fragile and vulnerable to adversarial perturbations on data samples (e.g., noisy data or purposeful user profile attacks [71]). To tackle the challenge, some models introduce the denoising techniques [1, 123], e.g., denoising AE-based models [132] by firstly corrupting the data using man-made noises. Besides, adversarial training [32, 104, 143] has also been adopted to improve the robustness of DL-based recommendation algorithms. For example, Yuan et al. [143] proposed a general adversarial training framework for neural network-based recommendation model and further designed a minmax game for nonlinear model optimization. They tested the effectiveness of the framework on the modified collaborative denoising AE (CDAE) model [132]. Furthermore, considering that previous adversarial training techniques might ignore to consider the characteristic of sequential data, Jia et al. [43] designed a new adversarial training approach for sequential data by specifically answering when and how to perturb a sequence.

6.1.8 Tackling Cold-start and Data Sparsity Challenges for Sequential Recommendation. Cold-start and data sparsity are long-standing challenges for recommender systems, including sequential recommendation [25, 101]. Most of the existing sequential recommendation algorithms we have previously discussed ignore to address these two issues, as their effective implementations rely on relatively strict requirements on sequential data (e.g., dropping sequences shorter than a minimal threshold). Therefore, future research can consider these two issues when building practical sequential recommender systems for real-world applications.

To tackle these two issues, first, machine learning techniques which are capable of learning from a limited number of data samples (e.g., few shot learning [125]), can be adopted for sequential recommendation. Few-shot learning strives to bridge the gap between artificial intelligence and human-like learning, and can learn a task with limited information by incorporating prior knowledge and deploying different ML techniques (e.g., the meta-learning, embedding learning and generative modeling methods) [125]. For instance, Du et al. [25] proposed s^2Meta (Scenario-specific Sequential Meta learner) which combines the scenario-specific learning with a model-agnostic sequential meta learning for more effective recommendation. Secondly, sequential recommendation algorithms can be built on different types of side information (e.g., social networks, user profiles, item descriptions, and knowledge graph) to partially relieve the two issues [69, 101], while deep neural networks are quite suitable to process multi-modal information. For example, *SDM* considers multiple types of side information, such as item ID, first level category, leaf category, brand and shop, to better model users' long-term preferences.

6.2 Conclusions

The study systematically investigated the DL-based sequential recommendation. Specifically, we designed a novel taxonomy for investigating the sequential recommendation tasks in terms of the three types of behavior sequences: experienced-based, transaction-based, and interaction-based. Based on it, we surveyed and explored a considerable amount of representative DL-based algorithms in the sequential recommendation, with the aim of a better understanding on whether sequential recommendation tasks have been sufficiently or insufficiently studied. Thirdly, for better guiding the development of DL-based sequential recommender systems, we thoroughly identified the possible influential factors that impact the performance of DL-based models in regards to recommendation accuracy from the four perspectives with respect to learning a better model: model input, data processing, model structure and model training. We further comprehensively showcased their impacts via well-designed evaluations, which can be viewed a testbed. Finally, we discussed the challenges and provided new potential directions for the research on DL-based sequential recommendation.

REFERENCES

[1] Xavier Amatriain, Josep M Pujol, Nava Tintarev, and Nuria Oliver. 2009. Rate it again: increasing recommendation accuracy by user re-rating. In *RecSys*. 173–180.

[2] Ashton Anderson, Ravi Kumar, Andrew Tomkins, and Sergei Vassilvitskii. 2014. The dynamics of repeat consumption. In *WWW*. 419–430.

[3] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS One* 10, 7 (2015).

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by Jointly Learning to Align and Translate. In *ICLR*.

[5] Ting Bai, Jian-Yun Nie, Wayne Xin Zhao, Yutao Zhu, Pan Du, and Ji-Rong Wen. 2018. An attribute-aware neural attentive model for next basket recommendation. In *SIGIR*. 1201–1204.

[6] Trapit Bansal, David Belanger, and Andrew McCallum. 2016. Ask the GRU: Multi-task learning for deep text recommendations. In *RecSys*. 107–114.

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

[8] Zeynep Batmaz, Ali Ihsan Yurekli, Alper Bilge, and Cihan Kaleli. 2018. A review on deep learning for recommender systems: challenges and remedies. *Artificial Intelligence Review* (2018), 1–37.

[9] Yoshua Bengio, Rejean Ducharme, Pascal Vincent, and Christian Janvin. 2003. A neural probabilistic language model. *Journal of Machine Learning Research* 3, 6 (2003), 1137–1155.

[10] Rahul Bhagat, Srevatsan Muralidharan, Alex Lobzhanidze, and Shankar Vishwanath. 2018. Buy It Again: Modeling Repeat Purchase Recommendations. In *KDD*. 62–70.

[11] Homanga Bharadhwaj and Shruti Joshi. 2018. Explanations for temporal recommendations. *Künstliche Intelligenz* 32, 4 (2018), 267–272.

[12] Veronika Bogina and Tsvi Kuflik. 2017. Incorporating dwell time in session-based recommendations with recurrent Neural networks. In *RecTemp@ RecSys*. 57–59.

[13] Iván Cantador, Ignacio Fernández-Tobías, Shlomo Berkovsky, and Paolo Cremonesi. 2015. Cross-domain recommender systems. In *Recommender systems handbook*. Springer, 919–959.

[14] Wanyu Chen, Pengjie Ren, Fei Cai, and Maarten de Rijke. 2019. Improving End-to-End Sequential Recommendations with Intent-aware Diversification. *arXiv preprint arXiv:1908.10171* (2019).

[15] Xu Chen, Hongteng Xu, Yongfeng Zhang, Jiaxi Tang, Yixin Cao, Zheng Qin, and Hongyuan Zha. 2018. Sequential recommendation with user memory networks. In *WSDM*. 108–116.

[16] Xu Chen, Yongfeng Zhang, Hongteng Xu, Zheng Qin, and Hongyuan Zha. 2018. Adversarial distillation for efficient recommendation with external knowledge. *TOIS* 37, 1 (2018), 1–28.

[17] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* (2014).

[18] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *RecSys*. 191–198.

[19] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. 2019. Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches. In *RecSys (RecSys '19)*. ACM, New York, NY, USA, 101–109. https://doi.org/10.1145/3298689.3347058

[20] Alexander Dallmann, Alexander Grimm, Christian Pölitz, Daniel Zoller, and Andreas Hotho. 2017. Improving session recommendation with recurrent neural networks by exploiting dwell time. *arXiv preprint arXiv:1706.10231* (2017).

[21] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, and Blake Livingston. 2010. The YouTube video recommendation system. In *RecSys*. 293–296.

[22] Robin Devooght and Hugues Bersini. 2017. Long and short-term recommendations with recurrent neural networks. In *UMAP*. 13–21.

[23] Jingtao Ding, Guanghui Yu, Xiangnan He, Yuhan Quan, Yong Li, Tat-Seng Chua, Depeng Jin, and Jiajie Yu. 2018. Improving implicit recommender systems with view data.. In *IJCAI*. 3343–3349.

[24] Tim Donkers, Benedikt Loepp, and Jürgen Ziegler. 2017. Sequential user-based recurrent neural network recommendations. In *RecSys*. 152–160.

[25] Zhengxiao Du, Xiaowei Wang, Hongxia Yang, Jingren Zhou, and Jie Tang. 2019. Sequential Scenario-Specific Meta Learner for Online Recommendation. In *KDD*. 2895–2904.

[26] Chen Gao, Xiangnan He, Danhua Gan, Xiangning Chen, Fuli Feng, Yong Li, Tat-Seng Chua, Lina Yao, Yang Song, and Depeng Jin. 2019. Learning to Recommend with Multiple Cascading Behaviors. *IEEE Transactions on Knowledge and Data Engineering* (2019).

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

[28] Guibing Guo, Huihuai Qiu, Zhenhua Tan, Yuan Liu, Jing Ma, and Xingwei Wang. 2017. Resolving data sparsity by multi-type auxiliary implicit feedback for recommender systems. *Knowledge-Based Systems* 138 (2017), 202–207.

[29] Ruining He, Chen Fang, Zhaowen Wang, and Julian McAuley. 2016. Vista: A visually, socially, and temporally-aware model for artistic recommendation. In *RecSys*. 309–316.

[30] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017. Translation-based recommendation. In *RecSys*. 161–169.

[31] Ruining He and Julian McAuley. 2016. Fusing similarity models with markov chains for sparse sequential recommendation. In *ICDM*. 191–200.

[32] Xiangnan He, Zhankui He, Xiaoyu Du, and Tat-Seng Chua. 2018. Adversarial personalized ranking for recommendation. In *SIGIR*. 355–364.

[33] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *CIKM*. 843–852.

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

[35] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. 2016. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *RecSys*. 241–248.

[36] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* (2015).

[37] Cheng-Kang Hsieh, Longqi Yang, Yin Cui, Tsung-Yi Lin, Serge Belongie, and Deborah Estrin. 2017. Collaborative metric learning. In *WWW*. 193–201.

[38] Kai-Chun Hsu, Szu-Yu Chou, Yi-Hsuan Yang, and Tai-Shih Chi. 2016. Neural network based next-song recommendation. *arXiv preprint arXiv:1606.07722* (2016).

[39] Haoji Hu, Xiangnan He, Jinyang Gao, and Zhi-Li Zhang. 2020. Modeling Personalized Item Frequency Information for Next-basket Recommendation. In *SIGIR*.

[40] Jin Huang, Wayne Xin Zhao, Hongjian Dou, Ji-Rong Wen, and Edward Y Chang. 2018. Improving sequential recommendation with knowledge-enhanced memory networks. In *SIGIR*. 505–514.

[41] Xiaowen Huang, Shengsheng Qian, Quan Fang, Jitao Sang, and Changsheng Xu. 2018. CSAN: Contextual self-attention network for user sequential recommendation. In *MM*. 447–455.

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

[43] Xiaowei Jia, Sheng Li, Handong Zhao, Sungchul Kim, and Vipin Kumar. 2019. Towards robust and discriminative sequential data learning: When and how to perform adversarial training?. In *KDD*. 1665–1673.

[44] Santosh Kabbur, Xia Ning, and George Karypis. 2013. Fism: factored item similarity models for top-n recommender systems. In *KDD*. 659–667.

[45] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *ICDM*. IEEE, 197–206.

[46] Pang Wei Koh and Percy Liang. 2017. Understanding black-box predictions via influence functions. In *ICML*. JMLR. org, 1885–1894.

[47] Yehuda Koren. 2009. Collaborative filtering with temporal dynamics. In *KDD*. ACM, 447–456.

[48] Artus Krohn-Grimberghe, Lucas Drumond, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2012. Multi-relational matrix factorization using bayesian personalized ranking for social network data. In *WSDM*. 173–182.

[49] Duc-Trong Le, Hady W Lauw, and Yuan Fang. 2018. Modeling contemporaneous basket sequences with twin networks for next-item recommendation. In *IJCAI*. 3414–3420.

[50] Joonseok Lee, Sami Abu-El-Haija, Balakrishnan Varadarajan, and Apostol Natsev. 2018. Collaborative deep metric learning for video understanding. In *KDD*. 481–490.

[51] Lukas Lerche, Dietmar Jannach, and Malte Ludewig. 2016. On the value of reminders within e-commerce recommendations. In *UMAP*. 27–35.

[52] Hui Li, Yanlin Wang, Ziyu Lyu, and Jieming Shi. 2020. Multi-task Learning for Recommendation over Heterogeneous Information Network. *TKDE* (2020).

[53] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *CIKM*. 1419–1428.

[54] Jiacheng Li, Yujie Wang, and Julian McAuley. 2020. Time Interval Aware Self-Attention for Sequential Recommendation. In *WSDM*. 322–330.

[55] Sheng Li, Jaya Kawale, and Yun Fu. 2015. Deep collaborative filtering via marginalized denoising auto-encoder. In *CIKM*. 811–820.

[56] Zhi Li, Hongke Zhao, Qi Liu, Zhenya Huang, Tao Mei, and Enhong Chen. 2018. Learning from history and present: next-item recommendation via discriminatively exploiting user behaviors. In *KDD*. 1734–1743.

[57] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing* 7, 1 (2003), 76–80.

[58] Kuan Liu, Xing Shi, and Prem Natarajan. 2018. A Sequential Embedding Approach for Item Recommendation with Heterogeneous Attributes. arXiv:cs.IR/1805.11008

[59] Qiang Liu, Shu Wu, Diyi Wang, Zhaokang Li, and Liang Wang. 2016. Context-aware sequential recommendation. In *ICDM*. 1053–1058.

[60] Qiang Liu, Shu Wu, and Liang Wang. 2017. Multi-behavioral sequential prediction with recurrent log-bilinear model. *TKDE* 29, 6 (2017), 1254–1267.

[61] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: short-term attention/memory priority model for session-based recommendation. In *KDD*. 1831–1839.

[62] Babak Loni, Roberto Pagano, Martha Larson, and Alan Hanjalic. 2016. Bayesian personalized ranking with multi-channel user feedback. In *RecSys*. 361–364.

[63] Pablo Loyola, Chen Liu, and Yu Hirate. 2017. Modeling user session and intent with an attention-based encoder-decoder architecture. In *RecSys*. 147–151.

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

[65] Fuyu Lv, Taiwei Jin, Changlong Yu, Fei Sun, Quan Lin, Keping Yang, and Wilfred Ng. 2019. SDM: Sequential deep matching model for online large-scale recommender system. In *CIKM*. 2635–2643.

[66] Chen Ma, Peng Kang, and Xue Liu. 2019. Hierarchical gating networks for sequential recommendation. In *KDD*. 825–833.

[67] Mingyuan Ma, Sen Na, Cong Xu, and Xin Fan. 2018. The graph-based broad behavior-aware recommendation system for interactive news. *arXiv preprint arXiv:1812.00002* (2018).

[68] Muyang Ma, Pengjie Ren, Yujie Lin, Zhumin Chen, Jun Ma, and Maarten de Rijke. 2019. π-Net: A Parallel Information-sharing Network for Shared-account Cross-domain Sequential Recommendations. In *SIGIR*. 685–694.

[69] Wenjing Meng, Deqing Yang, and Yanghua Xiao. 2020. Incorporating User Micro-behaviors and Item Knowledge into Multi-task Learning for Session-based Recommendation. In *SIGIR*.

[70] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[71] Bamshad Mobasher, Robin Burke, Runa Bhaumik, and Chad Williams. 2007. Toward trustworthy recommender systems: An analysis of attack models and algorithm robustness. *TOIT* 7, 4 (2007), 23–es.

[72] Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. 2018. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing* 73 (2018), 1–15.

[73] Hanh TH Nguyen, Martin Wistuba, Josif Grabocka, Lucas Rego Drumond, and Lars Schmidt-Thieme. 2017. Personalized deep learning for tag recommendation. In *PAKDD*. 186–197.

[74] Georgina Peake and Jun Wang. 2018. Explanation mining: Post hoc interpretability of latent factor models for recommendation systems. In *KDD*. 2060–2069.

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

[76] Huihuai Qiu, Yun Liu, Guibing Guo, Zhu Sun, Jie Zhang, and Hai Thanh Nguyen. 2018. BPRH: Bayesian personalized ranking for heterogeneous implicit feedback. *Information Sciences* 453 (2018), 80–98.

[77] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. 2018. Sequence-aware recommender systems. *ACM Computing Surveys (CSUR)* 51, 4 (2018), 1–36.

[78] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *RecSys*. 130–137.

[79] Yogesh Singh Rawat and Mohan S Kankanhalli. 2016. ConTagNet: Exploiting user context for image tag recommendation. In *MM*. 1102–1106.

[80] Kan Ren, Jiarui Qin, Yuchen Fang, Weinan Zhang, Lei Zheng, Weijie Bian, Guorui Zhou, Jian Xu, Yong Yu, Xiaoqiang Zhu, et al. 2019. Lifelong Sequential Modeling with Personalized Memorization for User Response Prediction. In *SIGIR*. 565–574.

[81] Pengjie Ren, Zhumin Chen, Jing Li, Zhaochun Ren, Jun Ma, and Maarten de Rijke. 2019. RepeatNet: A repeat aware neural recommendation machine for session-based recommendation. In *AAAI*, Vol. 33. 4806–4813.

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

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

[84] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. Fast context-aware recommendations with factorization machines. In *SIGIR*. 635–644.

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

[86] Noveen Sachdeva, Kartik Gupta, and Vikram Pudi. 2018. Attentive neural architecture incorporating song features for music recommendation. In *RecSys*. 417–421.

[87] Noveen Sachdeva, Giuseppe Manco, Ettore Ritacco, and Vikram Pudi. 2019. Sequential Variational Autoencoders for Collaborative Filtering. In *WSDM*.

[88] Thomas Schnake, Oliver Eberle, Jonas Lederer, Shinichi Nakajima, Kristof T Schütt, Klaus-Robert Müller, and Grégoire Montavon. 2020. XAI for Graphs: Explaining Graph Neural Network Predictions by Identifying Relevant Walks. *arXiv preprint arXiv:2006.03589* (2020).

[89] Tian Shi, Yaser Keneshloo, Naren Ramakrishnan, and Chandan K Reddy. 2018. Neural abstractive text summarization with sequence-to-sequence models. *arXiv preprint arXiv:1812.02303* (2018).

[90] Shun-Yao Shih and Heng-Yu Chi. 2018. Automatic, personalized, and flexible playlist generation using reinforcement learning. *arXiv preprint arXiv:1809.04214* (2018).

[91] Ajit P Singh and Geoffrey J Gordon. 2008. Relational learning via collective matrix factorization. In *KDD*. 650–658.

[92] Ayush Singhal, Pradeep Sinha, and Rakesh Pant. 2017. Use of deep learning in modern recommendation system: A summary of recent works. *IJCA* (2017).

[93] Elena Smirnova and Flavian Vasile. 2017. Contextual sequence modeling for recommendation with recurrent neural networks. In *the 2nd Workshop on Deep Learning for Recommender Systems*. 2–9.

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

[95] Weiping Song, Zhiping Xiao, Yifan Wang, Laurent Charlin, Ming Zhang, and Jian Tang. 2019. Session-based social recommendation via dynamic graph attention networks. In *WSDM*. 555–563.

[96] Younghun Song and Jae-Gil Lee. 2018. Augmenting recurrent neural networks with high-order user-contextual preference for session-based recommendation. *arXiv preprint arXiv:1805.02983* (2018).

[97] Gabriele Sottocornola, Panagiotis Symeonidis, and Markus Zanker. 2018. Session-based news recommendations. In *WWW*. 1395–1399.

[98] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[99] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *CIKM*. 1441–1450.

[100] Shiming Sun, Yuanhe Tang, Zemei Dai, and Fu Zhou. 2019. Self-Attention Network for Session-Based Recommendation With Streaming Data Input. *IEEE Access* (2019).

[101] Zhu Sun, Qing Guo, Jie Yang, Hui Fang, Guibing Guo, Jie Zhang, and Robin Burke. 2019. Research commentary on recommendations with side information: A survey and research directions. *Electronic Commerce Research and Applications* 37 (2019), 100879.

[102] Zhu Sun, Di Yu, Hui Fang, Jie Yang, Zhang Jie Qu, Xinghua, and Cong Geng. 2020. re We Evaluating Rigorously? Benchmarking Recommendation for Reproducible Evaluation and Fair Comparison. In *Recsys*.

[103] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved recurrent neural networks for session-based recommendations. In *the 1st Workshop on Deep Learning for Recommender Systems*. 17–22.

[104] Jinhui Tang, Xiaoyu Du, Xiangnan He, Fajie Yuan, Qi Tian, and Tat-Seng Chua. 2019. Adversarial training towards robust multimedia recommender system. *TKDE* (2019).

[105] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*. 565–573.

[106] Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Latent relational metric learning via memory-based attention for collaborative ranking. In *WWW*. 729–739.

[107] Trinh Xuan Tuan and Tu Minh Phuong. 2017. 3D convolutional networks for session-based recommendation with content features. In *RecSys*. 138–146.

[108] Bartłomiej Twardowski. 2016. Modelling contextual information in session-aware recommender systems with neural networks. In *RecSys*. 273–276.

[109] Vladimir Vapnik and Akshay Vashist. 2009. A new learning paradigm: Learning using privileged information. *Neural Networks* 22, 5-6 (2009), 544–557.

[110] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*. 5998–6008.

[111] Mengting Wan, Di Wang, Jie Liu, Paul Bennett, and Julian McAuley. 2018. Representing and Recommending Shopping Baskets with Complementarity, Compatibility and Loyalty. In *CIKM*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3269206.3271786

[112] Shengxian Wan, Yanyan Lan, Pengfei Wang, Jiafeng Guo, Jun Xu, and Xueqi Cheng. 2015. Next basket recommendation with neural networks. In *RecSys Posters*.

[113] Chenyang Wang, Min Zhang, Weizhi Ma, Yiqun Liu, and Shaoping Ma. 2019. Modeling Item-Specific Temporal Dynamics of Repeat Consumption for Recommender Systems. In *WWW*. 1977–1987.

[114] Haoyu Wang, Defu Lian, and Yong Ge. 2019. Binarized collaborative filtering with distilling graph convolutional networks. *arXiv preprint arXiv:1906.01829* (2019).

[115] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *KDD*. 1235–1244.

[116] Hao Wang, SHI Xingjian, and Dit-Yan Yeung. 2016. Collaborative recurrent autoencoder: Recommend while learning to fill in the blanks. In *NIPS*. 415–423.

[117] Pengfei Wang, Hanxiong Chen, Yadong Zhu, Huawei Shen, and Yongfeng Zhang. 2019. Unified Collaborative Filtering over Graph Embeddings. In *SIGIR*. 155–164.

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

[119] Shoujin Wang, Longbing Cao, and Yan Wang. 2019. A survey on session-based recommender systems. *arXiv preprint arXiv:1902.04864* (2019).

[120] Shoujin Wang, Liang Hu, Longbing Cao, Xiaoshui Huang, Defu Lian, and Wei Liu. 2018. Attention-based transactional context embedding for next-item recommendation. In *AAAI*. 2532–2539.

[121] Shoujin Wang, Liang Hu, Yan Wang, Longbing Cao, Quan Z Sheng, and Mehmet Orgun. 2019. Sequential recommender systems: challenges, progress and prospects. In *IJCAI*. AAAI Press, 6332–6338.

[122] Shoujin Wang, Liang Hu, Yan Wang, Quan Z. Sheng, Mehmet A. Orgun, and Longbing Cao. 2019. Modeling Multi-Purpose Sessions for Next-Item Recommendations via Mixture-Channel Purpose Routing Networks. In *IJCAI*. 3771–3777.

[123] Wenjie Wang, Fuli Feng, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. 2020. Denoising Implicit Feedback for Recommendation. *arXiv preprint arXiv:2006.04153* (2020).

[124] Xiang Wang, Yaokun Xu, Xiangnan He, Yixin Cao, Meng Wang, and Tat-Seng Chua. 2020. Reinforced Negative Sampling over Knowledge Graph for Recommendation. In *WWW*. 99–109.

[125] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. 2020. Generalizing from a few examples: A survey on few-shot learning. *CSUR* (2020), 1–34.

[126] Ziyang Wang, Wei Wei, Gao Cong, Xiao-Li Li, Xian-Ling Mao, and Minghui Qiu. 2020. Global Context Enhanced Graph Neural Networks for Session-based Recommendation. In *CIKM*. 169–178.

[127] Jian Wei, Jianhua He, Kai Chen, Yi Zhou, and Zuoyin Tang. 2016. Collaborative filtering and deep learning based hybrid recommendation for cold start problem. In *DASC/PiCom/DataCom/CyberSciTech*. 874–877.

[128] Chen Wu and Ming Yan. 2017. Session-aware information embedding for e-commerce product recommendation. In *CIKM*. 2379–2382.

[129] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent recommender networks. In *WSDM*. 495–503.

[130] Qitian Wu, Yirui Gao, Xiaofeng Gao, Paul Weng, and Guihai Chen. 2019. Dual Sequential Prediction Models Linking Sequential Recommendation and Information Dissemination. In *KDD*. 447–457.

[131] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2019. Session-based recommendation with graph neural networks. In *AAAI*, Vol. 33. 346–353.

[132] Yao Wu, Christopher DuBois, Alice X Zheng, and Martin Ester. 2016. Collaborative denoising auto-encoders for top-n recommender systems. In *WSDM*. 153–162.

[133] Qiaolin Xia, Peng Jiang, Fei Sun, Yi Zhang, Xiaobo Wang, and Zhifang Sui. 2018. Modeling consumer buying decision for recommendation based on multi-task deep learning. In *CIKM*. 1703–1706.

[134] Longqi Yang, Cheng-Kang Hsieh, Hongjian Yang, John P Pollak, Nicola Dell, Serge Belongie, Curtis Cole, and Deborah Estrin. 2017. Yum-me: a personalized nutrient-based meal recommender system. *TOIS* 36, 1 (2017), 1–31.

[135] Yiding Yang, Jiayan Qiu, Mingli Song, Dacheng Tao, and Xinchao Wang. 2020. Distilling Knowledge From Graph Convolutional Networks. In *CVPR*. 7074–7083.

[136] Zhen Yang, Ming Ding, Chang Zhou, Hongxia Yang, Jingren Zhou, and Jie Tang. 2020. Understanding Negative Sampling in Graph Representation Learning. In *KDD*.

[137] Haochao Ying, Fuzhen Zhuang, Fuzheng Zhang, Yanchi Liu, Guandong Xu, Xing Xie, Hui Xiong, and Jian Wu. 2018. Sequential recommender system based on hierarchical attention network. In *IJCAI*.

[138] Jiaxuan You, Yichen Wang, Aditya Pal, Pong Eksombatchai, Chuck Rosenburg, and Jure Leskovec. 2019. Hierarchical temporal convolutional networks for dynamic recommender systems. In *WWW*. 2236–2246.

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

[140] Lu Yu, Chuxu Zhang, Shangsong Liang, and Xiangliang Zhang. 2019. Multi-order attentive ranking model for sequential recommendation. In *AAAI*, Vol. 33. 5709–5716.

[141] Fajie Yuan, Xiangnan He, Haochuan Jiang, Guibing Guo, Jian Xiong, Zhezhao Xu, and Yilin Xiong. 2020. Future Data Helps Training: Modeling Future Contexts for Session-based Recommendation. In *WWW*. 303–313.

[142] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. 2019. A simple convolutional generative network for next item recommendation. In *WSDM*. 582–590.

[143] Feng Yuan, Lina Yao, and Boualem Benatallah. 2019. Adversarial Collaborative Neural Network for Robust Recommendation. In *SIGIR*.

[144] Mingge Zhang and Zhenyu Yang. 2019. GACOforRec: Session-Based Graph Convolutional Neural Networks Recommendation Model. *IEEE Access* (2019).

[145] Qi Zhang, Jiawen Wang, Haoran Huang, Xuanjing Huang, and Yeyun Gong. 2017. Hashtag recommendation for multimodal microblog using co-attention network.. In *IJCAI*. 3420–3426.

[146] Shuai Zhang, Yi Tay, Lina Yao, Aixin Sun, and Jake An. 2019. Next item recommendation with self-attentive metric learning. In *AAAI*, Vol. 9.

[147] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 5.

[148] Yongfeng Zhang and Xu Chen. 2018. Explainable recommendation: A survey and new perspectives. *arXiv preprint arXiv:1804.11192* (2018).

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

[150] Yang Zhang, Fuli Feng, Chenxu Wang, Xiangnan He, Meng Wang, Yan Li, and Yongdong Zhang. 2020. How to Retrain Recommender System? A Sequential Meta-Learning Method. In *SIGIR*.

[151] Wei Zhao, Benyou Wang, Min Yang, Jianbo Ye, Zhou Zhao, Xiaojun Chen, and Ying Shen. 2019. Leveraging Long and Short-Term Information in Content-Aware Movie Recommendation via Adversarial Training. *IEEE Transactions on Cybernetics* (2019).

[152] Zhe Zhao, Zhiyuan Cheng, Lichan Hong, and Ed H Chi. 2015. Improving user topic interest profiles by behavior factorization. In *WWW*. 1406–1416.

[153] Chang Zhou, Jinze Bai, Junshuai Song, Xiaofei Liu, Zhengchao Zhao, Xiusi Chen, and Jun Gao. 2018. ATRank: An attention-Based user behavior modeling framework for recommendation. In *AAAI*.

[154] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2018. Graph neural networks: a review of methods and applications. *arXiv preprint arXiv:1812.08434* (2018).

[155] Fuzhen Zhuang, Yingmin Zhou, Fuzheng Zhang, Xiang Ao, Xing Xie, and Qing He. 2017. Sequential transfer learning: cross-domain novelty seeking trait mining for recommendation. In *WWW Companion*. 881–882.