Deep Learning-based Sequential Recommender Systems: Concepts, Algorithms, and Evaluations

Hui Fang, Danning Zhang, Yiheng Shu, and Guibing Guo

Abstract—In the field of sequential recommendation, deep learning 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, DL-based methods also have some critical drawbacks, such as insufficient modeling of user representation and ignoring to distinguish the different types of interactions (i.e., user behavior) among users and items. 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 sequence, summarize the key factors affecting the performance of DL-based models, and conduct corresponding evaluations to demonstrate the effects of these factors. We conclude this survey by systematically outlining future directions and challenges in this field.

Index Terms—sequential recommendation, session-based recommendation, sequential data, deep learning, influential factors

1 Introduction

With the prevalence of information technology (IT), recommender system has long been acknowledged as an effective tool for addressing information overload problem, which makes users easily filter and locate information 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 on items on the basis of either explicit or implicit interactions between users and items. However, these approaches ignore to consider the time information of each interaction, not to mention to record that some interactions of a user could occur in a short time window, resulting in inaccurate modeling of users' preferences. In this case, sequential recommendation (a.k.a. session-based) has become increasingly popular in academic research and practical applications.

The sequential recommendation is also often referred to as session-based, session-aware, or sequence-aware recommendation. Considering that the concept of session is mainly used in e-commerce, whilst in other scenarios the boundaries between different terms are ambiguous, we thus use the 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 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 [1] and session-based KNN [2], which are criticized by their incomplete modeling problem, as they fail to thoroughly model users' long-term patterns by combining different sessions.

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 sequential recommender systems, and many DL-based models have achieved state-of-the-art performance [3]. The number of relevant arXiv articles in last 5 years 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 Challenge 2015[2]), POI (Point-of-Interest), music (e.g., Last.fm[3]), and movies/video (e.g., MovieLens[4]). Figure 2 depicts the word cloud of keywords in sequential recommendation related articles, which to some extent reflects the hot topics in sequential recommendation field.

On the other hand, we find that the DL-based models still have some 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 conclude all interactions into one type, instead of distinguishing different types. Therefore, we tend to thoroughly discuss these issues in this survey.

- *Hui Fang is with the School of Information Management and Engineering, Shanghai University of Finance and Economics, China.*
  E-mail: fang.hui@mail.shufe.edu.cn
- *Danning Zhang is with the School of Information Management and Engineering, Shanghai University of Finance and Economics, China.*
  E-mail: zhangdanning5@gmail.com
- *Yiheng Shu is with the Software College, Northeastern University, China.*
  E-mail: shuyiheng29@gmail.com
- *Guibing Guo is with the Software College, Northeastern University, China.*
  E-mail: guogb@swc.neu.edu.cn

1. We searched on arXiv.org with keywords related to the sequential recommendation and DL techniques in March 2019.
2. www.kaggle.com/chadgostopp/recsys-challenge-2015
3. labrosa.ee.columbia.edu/millionsong/lastfm
4. movielens.org

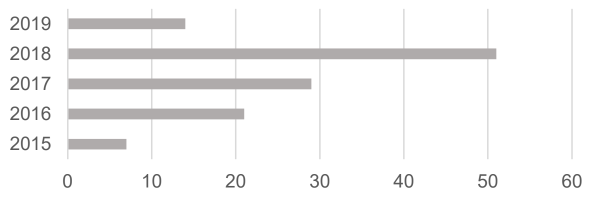


Fig. 1: The number of arXiv articles on DL-based sequential recommendation in recent five years.

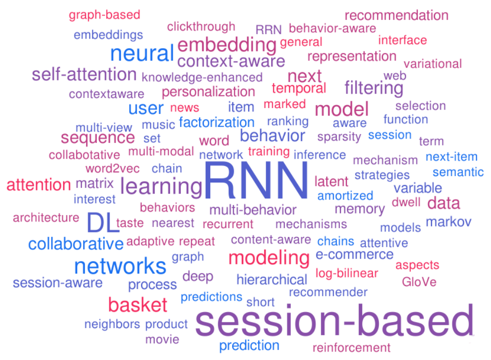


Fig. 2: Word cloud of the keywords in sequential recommendation related articles.

1.1 Related Survey

There have been some surveys on either DL-based recommendation or sequential recommendation. For DL-based recommendation, Singhal et al. [4] summarized DL-based recommender systems and categorized them into three types: collaborative filtering, content-based, and hybrid ones. Batmaz et al. [5] 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. [3] further discussed the state-of-the-art DL-based recommender systems, including several RNN-based sequential recommendation algorithms. For sequential recommendation, Quadrana et al. [6] proposed a categorization of the recommendation tasks and goals, and summarized existing solutions. Wang et al. [7] 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 DL-based sequential recommendation as well as discuss their issues, and explore the factors affecting their performance using a thorough demonstration of experimental evaluations on several real datasets.

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 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 sequential recommendation. First, we clarify the related concepts, and then formally describe the sequential recommendation task. Finally, we elaborate and compare the traditional ML and DL techniques for 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 choose 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.

Therefore, a **behavior** can be considered as a combination of behavior type and behavior object, i.e., a user interacting with a behavior object by a behavior type. A **behavior trajectory** can be defined as a behavior sequence consisting of multiple user behaviors. Thus, a sequential recommender system takes a user's behavior trajectories as input, and then recommends appropriate items or services to the user. 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., an behavior type c_i and behavior object o_i. A user who generates the sequence can either be anonymous or identified by the 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), behaviors within the basket may not be ordered by time, and then multiple baskets together form a behavior sequence.

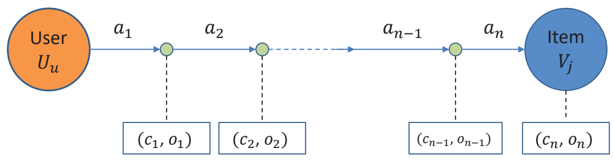


Fig. 3: A schematic diagram of the sequence recommendation. c_i: behavior type, o_i: behavior object. A behavior is represented by a 2-tuple. A behavior sequence (i.e., behavior trajectory) is a time ordered list of 2-tuples.

The input behavior sequence $\{a_1, a_2, a_3, ..., a_t\}$ is polymorphic, which can thus be divided into three types: *experience-based*, *transaction-based* and *interaction-based* behavior sequence, whose details are introduced as follows:

Experience-based behavior sequence. In an experience-based behavior sequence (see Fig. 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* keywords, then *clicks* the item of interest on the result pages and then *views* the details on the item page. Then the user may *share* the item with friends and *add it to cart* if she likes it. Different behavior types and orders 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 (after) *purchase* might imply the user's strong desire (satisfaction) to obtain (have) the item. For this type of behavior sequence, a model is expected to capture a user's underlying intentions indicated by different behaviors. The goal here is to predict the next behavior type (e.g., *buy*) that the user will impose if given an item.

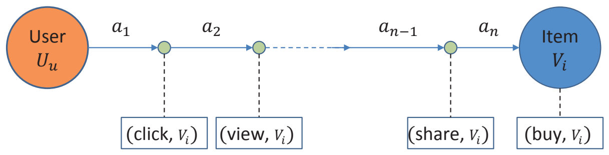


Fig. 4: Experience-based behavior sequence.

Transaction-based behavior sequence. The transaction-based behavior sequence (see Fig. 5) records a series of different items that a user interacts with, but the behaviors are of the same type (i.e. *buy*). In practice, *buy* behavior is the most concerned one for online sellers. Therefore, in transaction-based behavioral modeling, given the historical transactions of a user, the goal is to recommend the next item that the user will buy.

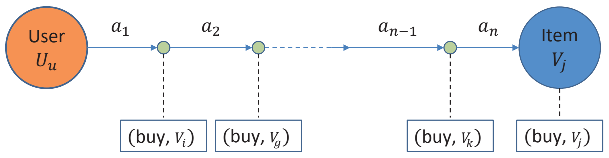


Fig. 5: Transaction-based behavior sequence.

Interaction-based behavior sequence. The interaction-based behavior sequence is a mixture of experience-based and transaction-based behavior sequences (see Fig. 6), i.e., consisting of different behavior objects and different behavior types simultaneously. It can be considered as a generalization of previous two types, and is much closer to the scenarios in real applications. In interaction-based behavioral sequence modeling, a recommender 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 goal is to predict the next item.

2.2 Sequential Recommendation Tasks

Figure 7 depicts two representative recommendation tasks in the literature: *next-item recommendation* and *next-basket recommendation*. In **next-item recommendation**, a user behavior contains only one object (i.e. item). Accordingly, an

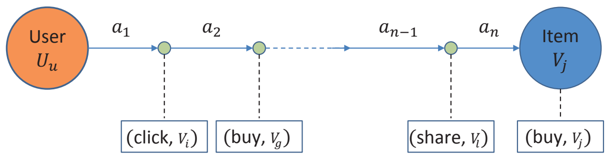


Fig. 6: Interaction-based behavior sequence.

item here also refers to information related to products, music, movies, or locations. In contrast, in **next-basket recommendation**, a user behavior contains more than one objects. However, either in next-item recommendation or

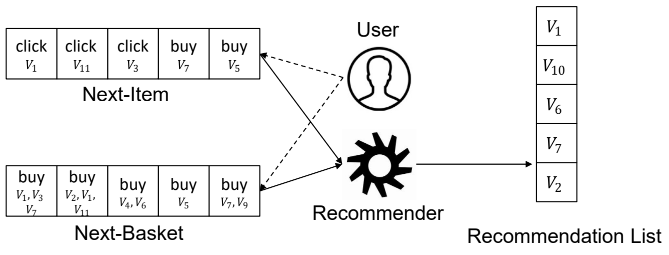


Fig. 7: Next-item and next-basket recommendation.

next-basket recommendation, the model is to predict the next item(s) for a user, and the most popular output is still a list of top-N ranked items, where the ranking could be determined by probabilities, absolute scores or relative rankings. In this case, softmax function is usually used to generate the output. Tan et al. [8] further proposed an embedding version of softmax output for fast prediction to accommodate the large volume of items in recommendation.

Therefore, the task of sequential recommendation can be considered as to generate a personalized item ranking list based on different types of user behavior sequences, 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 sequence $\{a_1, a_2, a_3, ..., a_t\}$ represents the input, u represents the corresponding user of the sequence, and p_i represents the probability that item i will be liked by user u at time $t+1$. I represents the number of candidate items for the recommender. The task can thus be considered to learn a complex function f for accurately predicting the probabilities that user u will choose each item at time $t+1$ based on the input sequence and the user profile.

Given the three types of behavior sequence, we can also categorize the sequential recommendation tasks as: *experience-based sequential recommendation*, *transaction-based sequential recommendation*, and *interaction-based recommendation*. We will comprehensively discuss these tasks in section 3.

2.3 Related Models

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

2.3.1 Traditional Methods

Conventional popular methods for sequential recommendation include frequent pattern mining, K-nearest neighbors, Markov chains, matrix factorization, and reinforcement learning [6]. They generally adopt matrix factorization for addressing users' long-term preferences across different sessions, whilst use first-order Markov Chains for capturing users' short-term interest within a session [2].

Next, we introduce these traditional methods as well as corresponding representative algorithms for sequential recommendation.

Frequent pattern mining. As we know, association rule [9] strives to use frequent pattern mining to mine frequent patterns with sufficient support and confidence. In sequential recommendation, a pattern is recognized only when the co-occurring items appear in the same order in different sequences, which is thus used to make recommendations. Although this kind of approaches is easy to implement, and relatively explicable for users, they suffer from the limited scalability problem as matching patterns, since 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-occurring 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 sequential recommendation. Item-based KNN [1], [10] only considers the last interaction in a given session and returns items that are most similar to it, where the similarities are usually calculated via the cosine similarity measure or other advanced ways [11].

In contrast, session-based KNN [10], [12], [13] compares the whole session with past sessions to recommend items, calculating similarities by Jaccard index or cosine similarity on binary vectors over the item space. It can be formulated as follows:

$$\text{Score}(s, i) = \sum_{w \in N_s} sim(s, w) \cdot 1_w\{i\} \tag{2}$$

where N_s denotes other neighboring sessions of session s, and $sim(\cdot)$ is a similarity function. The indicator function $1_w\{i\}$ returns 1 if session w contains item i, otherwise 0. KNN methods can generate highly explainable recommendation, and the similarities can also be pre-calculated, whereby recommendations can be quickly provided. However, the sequential dependency among items is completely ignored in KNN-based sequential recommendation.

Markov Chain (MC). In sequential recommendation, Markov models assume that future user behaviors only depend on the last or last few behaviors. For example, [14] only considers last behavior with first-order MC, while [2], [15] adopt high-order MCs, which considers the dependency with more previous behaviors. Considering only the last behavior or a few behaviors makes the MC-based models unable to leverage the dependency among behaviors in a long sequence and thus fail 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 [16] optimizes a pairwise ranking objective function via stochastic gradient descent (SGD). Twardowski [17] proposed a MF-based sequential recommender system (a simplified version of Factorization Machines [18]), where only the interaction between a session and a candidate item is considered for prediction. FPMC [19] is a representative baseline for next-basket recommendation, which integrates MF with first-order MCs. FISM [20] conducts matrix factorization on an item-item matrix, and thus no explicit user representation is learned. On the basis of FISM, FOSSIL [2] tackles the sequential prediction task by combining similarity-based methods and high-order Markov Chains. It performs better on sparse datasets in comparison with the traditional MC and FPMC methods. The main drawbacks of MF-based methods include: 1) a user or item's latent vector needs to be updated when the user have new experience, and computing cost is growing exponentially as the increase of the matrix size; 2) they generally ignore the time dependency among behaviors both within a session and across different sessions.

Reinforcement learning (RL). RL methods update recommendations according to the interactions between users and online platforms. When a platform recommends an item to a user, a positive reward is assigned if the user expresses her interest on the item (via behaviors such as click or view). This problem is usually formulated as a Markov decision process (MDP) whose goal is to maximize the cumulative rewards during several interactions [21], [22]. 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.

2.3.2 Deep Learning Techniques

Here, we summarize the DL models (e.g., RNN and CNN) that have been adopted by 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 sequential recommendation, RNN-based models are in the majority of DL-based models [23].

The main limitation of RNNs for sequential recommendation is that for dependency in longer sequence, their ability is limited, and training cost also increases.

Convolutional neural networks (CNNs). CNN is commonly used to process time series data and image data, and it typically 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 sequential recommendation, it can well capture local features within a session, where the input can take the time information into consideration [24], [25].

Multi-layer perceptrons (MLPs). MLP is a feed-forward neural network with multiple hidden layers, which can well learn the nonlinear relationship between the input and output via nonlinear activation functions, such as tanh and

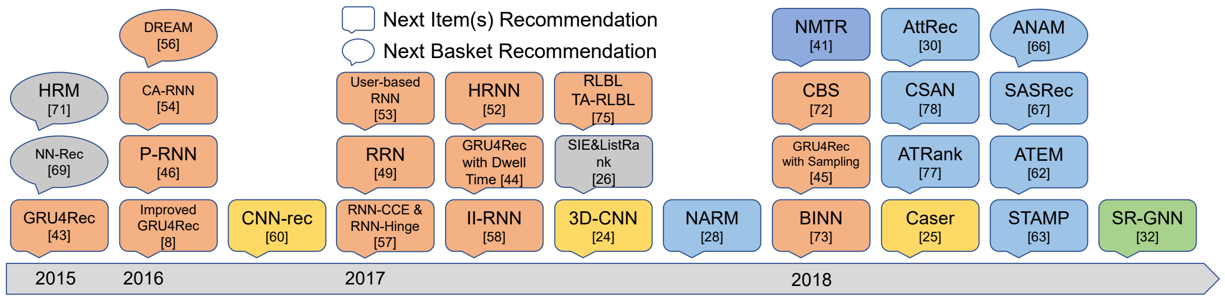


Fig. 8: Some recent and representative DL-based sequential recommendation models. Different colors indicate different DL techniques (grey: MLP; orange: RNN; yellow: CNN; blue: attention mechanism; green: GNN).

ReLU. Thus, they are expected to well capture the complex and nonlinear relationships among users behaviors [26].

Attention mechanisms. Attention mechanism is intuited from visual attentions of human-beings (incline to be attracted by more important parts of a target object). Bahdanau et al. [27] originally proposed an attention mechanism in neural machine translation task, which focused on modeling the importance of different parts of the input sentence on the output word.

Vanilla attention is to integrate this mechanism as a decoder of RNN, and has been widely used in sequential recommendation [28]. Besides, *self-attention mechanism* (originated in transformer [29] for neural machine translation by Google 2017), does not include RNN and vanilla attention, but performs much better than RNN-based models in recommender systems [30].

Graph neural networks (GNNs). GNN [31] is proposed to collectively aggregate information from graph structure. Wu et al. [32] 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.

2.3.3 *Concluding remarks.*

Compared with conventional methods, DL methods are a very active research area. 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 [7], as they are able to effectively learn the *theme* of the whole sequence. Thus, they generally obtain higher accuracy than traditional models. Meanwhile, DL methods are more flexible as they are 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; 2) the optimization is generally very challenging and more training data is required for complex networks. For the first problem, Huang et al. [33] proposed a knowledge-enhanced sequential recommender system, which combined RNN-based networks with key-value memory networks.

3 Sequential Recommendation Algorithms

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

3.1 Experience-based Sequential Recommendation

As we have introduced, in a experience-based behavior sequence, a user interacts with a same item with different behavior types. Its goal is to predict the next behavior type that the user will implement with the item, also referred to as *multi-behavior recommendation*. Next, we first explore the related work on multi-behavior recommendation and then introduce DL-based models that leverage multi-behavior information in sequential recommendation.

Conventional models for multi-behavior recommendation. Ajit el al. [34] 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 works [35], [36] extended CMF to handle different user behaviors. Besides, there are also some models addressing multi-behavior recommendation with Bayesian learning. For example, Loni et al. [37] proposed multi-channel BRP to adapt the sampling rule for different behavior types. Qiu et al. [38] further proposed an adaptive sampling method for BPR by considering the co-occurrence of multiple behavior types. Guo et al. [39] 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. [40] developed a margin-based learning framework to model the pairwise ranking relations among purchase, view, and non-view behaviors.

DL-based multi-behavior recommendation. DL techniques have also been applied in multi-behavior recommendation. For example, *NMTR* [41] 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. It formally defines the multi-behavior recommendation problem as: given a prediction of a candidate behavior, the input is user-item interaction data regarding the target behavior as well as other behavior types, and the output is the probability that a user will interact with an item under the target behavior. To capture the sequential relationships between behavior types, [41] cascaded predictions of different behavior types by considering the sequential dependency relationship among different behaviors in practice[5], which thus translates the heterogeneous behavior problem into the experience-based sequential recommendation problem as we have defined. Specifically, prediction tasks for different behaviors can be cascaded as follows:

$$
\begin{aligned}
\hat{y}_{ui}^{R} &= \sigma(\hat{y}_{ui}^{R-1} + f_{\Theta}^{R}(\mathbf{p}_u, \mathbf{q}_i) + b_i^R), \\
&\cdots\cdots \\
\hat{y}_{ui}^{2} &= \sigma(\hat{y}_{ui}^{1} + f_{\Theta}^{2}(\mathbf{p}_u, \mathbf{q}_i) + b_i^2), \\
\hat{y}_{ui}^{1} &= \sigma(f_{\Theta}^{1}(\mathbf{p}_u, \mathbf{q}_i) + b_i^1)
\end{aligned}
\tag{3}
$$

where $\mathbf{p}_u$ and $\mathbf{q}_i$ are user u and item i's embedding, respectively. b_i^r denotes the bias of item i regarding to behavior type r, and f_{Θ}^{r} denotes the interaction function for the behavior type r. R is the target behavior, and $\hat{y}_{ui}^{R}$ denotes the predicted probability that user u will interact with item i in terms of the behavior type R. It should be noted 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 ignore the ones between target behavior and subsequent behaviors. In this case, 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 [41] is generally better than that of sequential training.

Besides, Xia et al. [42] proposed a multi-task model with LSTM to explicitly model consumers' 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 come up with more accurate recommendations.

3.2 Transaction-based Sequential Recommendation

In transaction-based sequential recommendation, there is only a single behavior type (transaction related), 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 explored in transaction-based sequential recommendation task, and we summarize RNN-based approaches from the following perspectives.

(1) GRU4Rec-related models. Hidasi et al. [43] 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 [8], [44], [45] have been proposed and also use RNN architectures for sequential behavior modeling. The architecture of GRU4Rec is shown in Figure 9. As introduced in the original paper [43], its input is a session, 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). For a multi-layer GRU model, the output of each layer is the input to the next layer, but each layer can also be connected to a deeper non-adjacent GRU in the network. Feedforward layers are added between the last GRU layer and the output layer. The output is the probability scores of the candidate items appeared in the next behavior. The GRU4Rec model employs *session-parallel mini-*

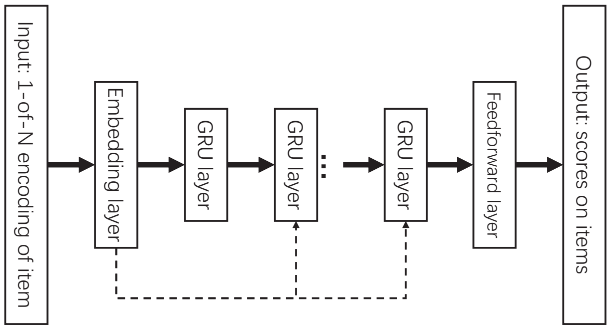


Fig. 9: Architecture of GRU4Rec neural network.

batches for training. Specifically, sessions are firstly arranged in time order. Then, we use the first event (behavior) of the first X sessions (X is the number of sessions) 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 reach its ending, the next available session out of the X sessions is placed in the corresponding place to continually form the mini-batch. The reason for using session-parallel mini-batches is that the length of sessions can be very different, and our goal is to capture how a session evolves over time, thus we could not simply break a session into different parts to force them into equal length [43]. Besides, in training, GRU4Rec uses *popularity-based negative sampling*, which assumes that the more popular an item is, the more possible for a user knows about it. In this case, if a user does not interact with it previously, it is more likely that the user dislikes it.

The improved works 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**, [8] applied *data augmentation* to enhance training of GRU4Rec. [44] considered the *dwell time* to modify the generation of mini-batch, which has verified to greatly improve performance. In particular, assuming that a predefined threshold of dwell time is t seconds, if the dwell time on an object i in a session is within the range of $[2t, 3t)$, then the parallel mini-batch of this session will contain 2 repeated behaviors of i, i.e., the presence of i in the session is increased. This strategy (referred as *item boosting*) can be considered to re-measure the

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

importance of behavior objects in terms of the corresponding dwell time. As popularity-based sampling suffers from the problem that model learning slows down after all the candidates items have been ranked above popular ones, which could be serious problem for long-tail items recommendation, [45] proposed additional sampling (a combination of uniform sampling and popularity sampling) for negative sampling in GRU4RC, which can improve recommendation accuracy by up to 53%. In 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 the parameter ($0 \leq \alpha \leq 1$), where the scenarios of $\alpha = 0$ and $\alpha = 1$ are equivalent to uniform and popularity-based sampling respectively.

For **better modeling item information**, [46] considered more 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). The first parallel architecture trains each GRU network (i.e., subnet) for item representation on the basis of each kind of information. The model concatenates the hidden layers of the subnets and computes 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 computing the score of the subnet. Moreover, they proposed more suitable alternative training strategies for p-RNNs than standard training, i.e., *simultaneous*, *alternating*, *residual* and *interleaving* training. In simultaneous training (baseline), each parameter of each subnet is trained simultaneously. In alternating training, subnets are trained in an alternating pattern per epoch. In residual training, subnets are trained one after the other, on the residual error of the ensemble of the previously trained subnets. Interleaving training is alternating training per mini-batch.

[13] combined traditional session-based KNN method with GRU4Rec using the methods of *switching*, *cascading*, and *weighted hybrid*, and demonstrated that the weighted hybrid version exceeds both original GRU4Rec and KNN methods and verified that RNN can better model item dependency than KNN.

(2) With user representation. There are also some works that try to better model users' preference. For example, [47] proposed an RNN-based framework for click-through rate (CTR) prediction in sponsor search, which considers the effect of 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. Specifically, for time $t \in \mathbf{N}$, the model takes a user's current feature vector $i(t)$ as input for RNN node, whose output $h(t)$ could thus be considered as a *dynamic representation of the user* on the basis of her previous behaviors. 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).

[48] took one-hot encoding of items in users' behavior sequences as input, converted them into dense vectors via lookup operation, which are sequentially fed into GRU-based RNN to learn users' historical embedding. Finally, the model projects user u's historical embedding h, user embedding, and target item i's embedding to predict user u's probability in choosing item i. *RRN* [49] is the first recurrent recommender network that attempts to capture the dynamics of both user and item representation, where each individual recurrent network is adopted to address the temporal evolution of each user and item respectively. To capture stationary attributes, it uses an additional set of auxiliary parameters for users and items respectively. [50] further improved the RRN's interpretability by devising a time-varying neighborhood style explanation scheme, which jointly optimizes prediction accuracy and interpretability of 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. [51] proposed a memory-augmented neural network for the sequential recommendation, which explicitly stores and updates the user's historical information by leveraging an external memory matrix. A user's representation is generated from two parts: the user's memory matrix and a free vector used to encode her intrinsic preference, which is not influenced by previous behaviors.

HRNN [52][6] uses GRU to model users and sessions respectively. The session-level GRU considers a user's activities within a session and thus generates recommendations. The user-level GRU models the evolution of the user's preference across sessions. When a session of a user starts, session-level GRU is initialized by the user-level GRU. At the end of each session, the user-level GRU is updated by the session-level GRU. Since the length of sessions for different users are varied, it adopts user parallel mini-batch training, which is extend from session parallel mini-batch of GRU4Rec. Donkers et al. [53] 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 user representation.

(3) Context-aware sequential recommendation. Most of the previous models ignore the huge amount of context information in real-word scenarios. In this case, [54] summarized two types of contexts: *input contexts* and *transition contexts*. Input contexts refer to the ones which users conduct behaviors, e.g., location, time and weather, whilst transition contexts mean transitions between two adjacent input elements in historical sequences (e.g., time intervals between adjacent behaviors). They thus further designed context-aware recurrent neural networks (*CA-RNN*) to simultaneously model sequential and contextual information, where input contexts and transition contexts are modeled by adaptive context-specific input matrices and transition matrices, respectively. [55] proposed *ARNN* to consider more rich user-side contexts, e.g., age, gender and location. Specifically, it extracts high-order user-contextual preference using a product-based neural network which is claimed to be

6. github.com/mquad/hgru4rec

capable of being incorporated with any existing RNN-based sequential recommendation models.

(4) Other models. *DREAM* [56] utilizes max pooling operation to model the correlations among objects in a basket to obtain the basket vector. Then, it sequentially feeds basket vectors into RNN to model sequential dependencies among baskets. The hidden state $h_{t_i}^u$ of i-th node can be considered as user u's user representation at time t_i. The probabilities of next items are calculated through multiplication of item embedding matrix M_N and user's representation $h_{t_i}^u$:

$$p_{u,t_i} = M_N^\top h_{t_i}^u \tag{4}$$

where N is the number of items in the dataset, p_{u,t_i} is a N dimensional vector, whose element represents the probability of the corresponding item that will be chosen by u in the next basket. [57] used RNN for collaborative filtering and considered two different objective functions in the RNN model: categorical cross-entropy (*CCE*) and **Hinge**, where CCE is the most widely used in language modeling, and Hinge is extended from the objective function of SVMs. [58] uses multi-layer GRU network to capture sequential dependencies and user interest from both inter-session and intra-session level.

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 [59]. For example, *3D-CNN* [24] designed an embedding matrix to concatenate the embedding of item id, name, and category. *Caser* [25] 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* [60] is highly similar to Caser in terms of user embedding and horizontal convolution, but it does not deploy vertical convolution. *NextItNet* [61] is a generative CNN model with residual block structure for sequential recommendation. It is capable of capturing both long and short-term item dependencies.

3.2.3 Attention-based Models

The attention mechanisms has been applied to sequential recommendation, and are capable of identifying more 'relevant' items to a user given the user's historical experience. We conclude the existing models according to the deployed type of attention mechanisms: *vanilla attention* and *self-attention* (Section 2.3.2).

(1) Vanilla attention mechanisms. *NARM*[7] [28] is an encoder-decoder framework for transaction-based sequential recommendation. Specifically, the encoder is hybrid and consists of two sub-encoders: global encoder and local encoder. The local encoder combines RNN with vanilla attention to captures the major purposes (or interest) of a user in the current sequence, while the global encoder utilizes RNN to model sequential dependencies among behaviors also in the current sequence, which together are used to obtain a unified sequence representation. The decoder uses a bi-linear scheme and calculates recommendations based on the unified sequence representation. With the attention mechanism, NARM is able to eliminate noises from unintended behaviors, such as accidental (unintended) clicks. [62] applied the vanilla attention mechanism to weight each item in a sequence to reduce the negative impact of unintended interactions. Liu et al. [63] 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 correlation between previous items and the most recent item in the sequence. Ren et al. [64] considered repeat consumption issue, and thus proposed *RepeatNet*. The structure of RepeatNet is the encoder-decoder. It evaluates the recommendation probability from both the repeat mode and the explore mode. They refer to the old item from the user's history and the new item, respectively. [65] incorporates vanilla attention with Bi-GRU network to model user's short-term interest for music recommendation. Firstly, it converts song ID and tags of the song into dense vectors, and then feed them into two parallel Bi-GRU respectively. Each Bi-GRU network is followed by the attention layer, which uses vanilla attention to calculated a weighted sum of all hidden states of Bi-GRU. The final probability are calculated from the concatenation of the output of two attention layers. [66] proposed a unified attribute-aware neural attentive model, which applies attention mechanism to feature level. It uses hierarchical attentive architecture to track user's varying appetite for both items and their attributes.

(2) Self-attention mechanisms. SASRec [67] aims to balance short-term intent and long-term preference, and seek to identify items relevant to the next behavior from the user's historical behavior sequence. The main components of the model include a embedding layer, a self-attention layer, and a point-wise feedforward network. In order to learn more complex item transitions, we can stack more self-attention blocks (attention layer and feed-forward network). Finally, the output from the last self-attention block is used to predict the probability of an item as the next item in a given sequence. BERT4Rec [68] is the improved version of SASRec, which introduces transformer architecture for sequential recommendation and trains the bidirectional model to model sequential data by using Cloze task. Zhang et al. [30] utilized the self-attention mechanism to infer the item-item relationship from the user's historical interactions. It takes item embedding vector as input, combines both a user's short-term intention and long-term interest to predict the next item. 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.

3.2.4 MLP-based Models

NN-rec [69] is the first work considering neural network for next-basket recommendation and inspired by the NLP model (NLPM) [70]. It aims to capture the local context in a user's last k baskets and predict the next basket. Specifically, it consists of four layers: the input layer, embedding layer,

7. github.com/lijingsdu/sessionRec_NARM

hidden layer, and output layer (softmax layer): 1) the input is one-hot encoding vectors of user ID and item IDs appeared in the user's last k baskets; 2) the embedding layer transforms the input into dense representations and produces basket vectors via an average pooling operation; 3) in the (dense) hidden layer, the input is the concatenation of user representation and k basket vectors obtained in the embedding layer; 4) the output layer uses a softmax function to predict the probabilities of all the candidates items in the next behavior.

HRM [71] is MLP-based model for next-basket recommendation. Specifically, it implements non-linear operations to consider the complex correlations between a user's short term preference (in the most recent basket) and her long-term preference. Its core is the two aggregation layers, where each aggregation operation can be either average pooling or max pooling. The first aggregation layer models correlations among items in a user's last basket (transaction) and forms the transaction representation via aggregating item vectors in the basket (a user's short-term interest). The second layer forms a hybrid representation by incorporating the user's short-term preference (transaction representation) with user's general interest, which is learned globally via user representation across all users. The hybrid representation is then used to predict items in the next basket.

3.2.5 Other Models

Wu et al. [32] first used GNN for session-based recommendation, which can capture complex transition of items. In this model, each session are modeled as a directed graph, and are proceeded by a gated graph neural network. After updating all nodes in session graphs until convergence, final node vectors are obtained. The session representation consists of local session embedding and global session embedding, which are calculated through final node vectors. The probabilities of next items are calculated from session representations and item embeddings. Sachdeva et al. [72] explored the variational autoencoders for modeling user preference through history sequence, which combines latent variables with temporal dependencies for preference modeling.

3.3 Interaction-based Sequential Recommendation

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

3.3.1 RNN-based Models

[17] proposed a RNN-based model without explicitly identifying user representation. It firstly converts behavior type and behavior object into embedding vectors, and then sequentially feeds these embedding vectors $\{X_{em}^{(1)}, X_{em}^{(2)}, ..., X_{em}^{(t)}\}$ into RNN, where $X_{em}^{(t)}$ is the embedding result of time t. The hidden state of RNN at time t (h_t) can be viewed as a sequence representation. X_i denotes the embedding vector of the target item i. The model concatenates X_i with h_t, which is together input into a feed-forward network to get y_i^t, the probability score of item i at time t given the current sequence. Given the task of predicting the next item expected to appear in terms of a target behavior, Le et al. [73] firstly divided a session into a target sequence and a supporting sequence according to the target behavior. Its basis idea is that the target behavior (e.g., purchase) contains the most efficient information for the prediction task, and the remaining behaviors (e.g., click) can thus utilized as the supporting sequence that can facilitate assist the next-item prediction task in target sequence. Considering the cascading relationship among different types of behaviors, Li et al. [74] proposed a model that consists of two main components: *neural item embedding* and *discriminative behavior learning*. For neural item embedding, an improved item2vec [75] is used to capture item similarities and learn unified item representations. For behavior learning, it utilizes all types of behavior (e.g., click, purchase and collect) to capture user's present consumption motivation. Meanwhile, it selects purchase behaviors 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. [76] combined RNN with LBL to construct two models (*RLBL* and *TA-RLBL*) for modeling multi-behavioral sequences. In particular, RLBL incorporates position-specific matrices and the recurrent structure to well model both short-term and long-term contexts, where several elements are simultaneously modeled in each hidden layer of RNN. Besides, behavior specific matrices are incorporated to capture properties of multiple behavior types. TA-RLBL is an extension of RLBL [76] which considers the time difference information. With regard to that sequential models often ignore the continuous time difference between input elements, TA-RLBL uses time-specific matrices to jointly model sequential information and time difference information, which further improves the performance of RLBL.

[77] took context information (e.g., behavior type) into consideration by modifying the structures of RNN. It considers two ways for incorporating context information with RNN: conditioning item representation on the context and conditioning hidden dynamics of RNN.

3.3.2 Attention-based Models

Attention mechanism is also validated to be effective in interaction-based sequential recommendation. For example, [78] proposed *ATRank*[8] which considers both *self-attention* and *vanilla attention* mechanisms. Specifically, it divides behaviors in a sequence into different groups in terms of different behavior types, where each group corresponds to a behavior embedding space projected by corresponding item embedding, behavior type embedding, and timestamp embedding. Considering the heterogeneity of behaviors, ATRank models the influence among behaviors via self-attention, while each semantic space corresponds to an attention matrix C. The element in position (i, j) in C indicates the influence between behavior i and j. Finally, it performs

8. github.com/jinze1994/ATRank.

vanilla attention on attention matrices for recommendation task. *CSAN* [79] is the improved version of ATRank by also considering side information and polysemy of behaviors. To be specific, it concatenates one-hot encoding of behavior type, item embedding and representations of side information (text representation via word2vec and image representation via CNN) to represent a behavior. It projects all behaviors into a common semantic space and then feeds them into a feature-wise self-attention network to capture polysemy of behaviors. Each feature corresponds to an attention matrix. Unlike ATRank, CSAN encodes position information of behaviors with two position encoding matrix.

Loyola et al. [80] proposed an encoder-decoder architecture for modeling user sessions in e-commerce, which incorporates two RNNs. Particularly, the encoder uses a bidirectional RNN to receive the input sequence and generates a fixed length representation. The RNN-based decoder takes the output of the encoder and uses it as the initial state from which it starts to estimate the target sequence. Besides, a second decoder can be added to incorporate user intent in two ways: (1) to predict the intent in the current session; (2) to obtain the intent from the next session.

3.3.3 Other Models

There are also some other DL techniques applied in interaction-based sequential recommendation, including MLPs and Graph-based model. For example, Wu et al. [26] proposed a deep listNet ranking framework, consisting of two parts: SIE and list-wise ranking. In SIE, it aggregates user's clicks and views separately to form click and view embedding vectors by using max pooling or average pooling, and then concatenates them with user embedding and target item embedding. Then, a feed-forward network is adopted to learn hybrid representation of a given session on the basis of the concatenation. In list-wise ranking, it calculates relevance score between obtained session representation and candidate items by mapping item ID vectors into dense vectors, and then uses DNN to project them to the space of session representation. Besides, Ma et al. [81] proposed a graph-based broad-aware network (*G-BBAN*) for news recommendation, which considers multiple user behaviors, behavioral sequence representations, and user representation. They established an interaction behavior graph for structuring the multi-level and multi-category data. G-BBAN predicts the probabilities of users' next potential behaviors on a news.

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 [78] and CSAN [79] adopt the same attention score for the item and the corresponding behavior type; (2) different behavior types are not distinguished successfully. For example, [17] 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 is easily ignored. For example, [26] used pooling operation to model multi-type behavior in a sequence. In view of these, more advanced and effective approaches are needed for sequential recommendation, especially for the task of interaction-based sequential recommendation. In the next sections, we will further summarize and evaluate the factors that might impact the performance of a DL-based model, which is expected to better guide future research.

TABLE 1: Categorizations of representative algorithms regarding sequential recommendation tasks, and basic DL models.

Task[1]	**Model**[2]	**Paper**
EB	RNN	[41], [42]
TB	RNN	[8], [43], [44], [51]–[56] [13], [45]–[50], [57], [58]
	CNN	[24], [25], [60], [61]
	MLP	[69], [71]
	att.	[28], [30], [62]–[68]
	GNN	[32]
IB	RNN	[17], [73], [74], [76], [77]
	MLP	[26]
	att.	[78]–[80]
	GNN	[81]

1 **Task** EB: Experience-based; TB: Transaction-based; IB: Interaction-based

2 **DL Model** att.: attention; RNN: recurrent neural network; CNN: convolutional neural network; MLP: multilayer perceptron; GNN: graph neural network

4 Influential Factors on 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. Thirdly, 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 given the processed data.

On the basis of a thorough literature study, we identify some representative factors (listed in grey boxes in Figure 10) that might impact the performance of DL-based models. 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.

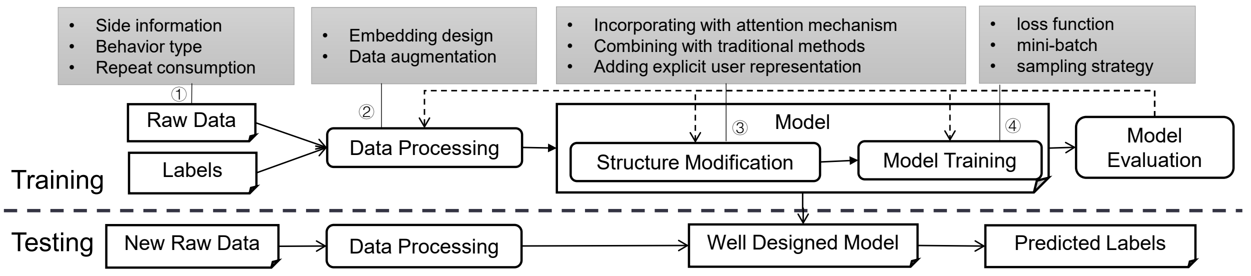


Fig. 10: Influential factors of DL-based models.

4.1.1 *Side Information*

The side information refers to information about items (other than IDs), e.g., *images*, *text descriptions*, and reviews, or information related to transactions (behavior) like *dwell time*. Text and image information about items have been widely explored in DL-based collaborative filtering systems [82]–[86], as well as in some DL-based sequential recommender systems [46], [79]. For example, p-RNN [46] uses a parallel RNNs framework to process the item IDs, images and texts. CSAN [79] utilizes 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 [46], [87], [88], cold-start [78], [79], [89], [90], and improve the model performance.

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 she stays, we can infer that the more she is interested in. Bogina et al. [44] applied item boosting according to the dwell time for generating mini-batch in training. Zhang et al. [47] treated dwell time as a sequence feature, and concatenated it with other features (e.g., query text). Similarly, Dallmann et al. [91] proposed an extension to existing RNN approaches by adding user dwell time. Experiments in [44] show that incorporating dwell time with GRU4Rec [43] 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 [78], [79], and different behaviors 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 [17], [41], [73], [78]. For example, CBS [73] divides a sequence into target sequence and supportive one, where the target sequence is related to the behavior (e.g., purchase) that has the most efficient information for prediction. Similarly, BINN [41] utilizes all action 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. 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 [17], [41], [73], [78].

4.1.3 *Repeat Consumption*

Repeat consumption refers to that an item is repeatedly appeared in a user's historical sequence, which is mostly ignored in sequential recommendation. Only *RepeatNet* proposed by Ren et al. [64] [9] has ever considered the issue, and their results confirm that the consideration of repeat consumption in network design can improve the recommendation performance.

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 as effective in existing DL-based models.

4.2.1 *Embedding Design*

In sequential recommendation, embedding methods are used to represent information about an item, a user, or a session. For example, Greenstein et al. [92] adopts the word embedding methods GloVe [93] and Word2Vec [94] (CBOW) for *item embedding* in e-commerce applications. Li et al. [74] further proposed w-item2vec (inspired by item2vec [75]) on the basis of the Skip-gram model and thus formed a unified representation of items. Wu et al. [26] 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.

4.2.2 *Data Augmentation*

In the sequential recommendation, sometimes 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. In this case, data augmentation becomes an important technique. For example, Tan et al. [8] proposed an augmentation method, where prefixes of the original input sessions are treated as new training sequences as shown in Figure 11, and the recommender predicts the last item for each training sequence. The method makes a session to be repeatedly utilized during training, which is demonstrated to improve 14.7% over GRU4Rec on MRR@20 [8]. Besides, the *dropout* method [95] is further adopted to prevent over-fitting problem (see Figure 11). Another consideration is that items after a target item

9. github.com/PengjieRen/RepeatNet

may also contain valuable information. Therefore, these items are utilized as privileged information as [96] to facilitate the learning process. Similarly, with regard to two behavior types (i.e., add-to-cart and click), 3D-CNN [24] 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.

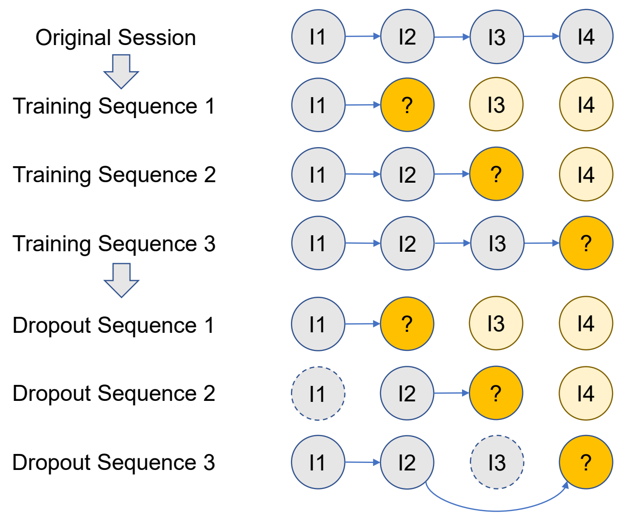


Fig. 11: Data argumentation. The orange circles represent the predicted items; the dotted circles represent the item that is deleted in the dropout method, and light orange circles make up privileged information.

4.3 Model Structure

We summarize the major methods to improve model structures 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 sequential recommendation problems. For the first scenario, NARM [28], ATEM [62] and STAMP [63] incorporate the *vanilla attention mechanism* with RNN or MLP, aiming to capture user's main purpose in a given session and experiments verify that their performance surpassed GRU4Rec by 25%, 92% and 30% respectively. SASRec [67] 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 [25], respectively.

For the second scenario, for example, AttRec [30] simply use self-attention to capture users' short-term interest, where its performance exceeds Caser by 8.5% on HR@50. ATRank [78] and CSAN [79] combine self-attention with vanilla attention for sequential recommendation. Attention mechanisms can be further employed to capture attribute-level importance level of items for users' interest. For example, ANAM [66] applies attention mechanism to track a user's appetite for items and their attributes.

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

4.3.2 Combining with Conventional Methods

DL models can also be combined with traditional methods to bootstrap their performance on sequential recommendation tasks. For example, [13] combines session-based KNN with GRU4Rec [43] in three ways, showing that the best combination can exceed original GRU4Rec by 9.8%. AttRec [30] combines self-attention (for short-term interest learning) and metric learning (for long-term preference modeling), and the performance exceeds Caser [25] 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 explicit user representation, i.e., users' long-term preferences can be well modeled by *user embedded models* or *user recurrent models*.

User embedded models. These models explicitly learn user representation [25], [71] via embedding methods, but not in a recurrent process as item representation. They can facilitate the performance of sequential recommendation models [25]. 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 embed models is learned in a relatively static way, which cannot capture users evolved and dynamic preferences. In this view, user recurrent models are more effective, which also learn user representation in a recurrent way.

User recurrent models. They treat both user and item representations as recurrent components in DL-based models, which can better capture users' evolving preferences, including memory-augmented neural network [51], RNN-based models [52], [53], [74] and recurrent neural networks [49], [50]. For example, [52], [53], [74] use RNN framework to learn users' long-term interest from their historical behavior sequences, and experiments verify that considering a user's long-term interest is critically valuable for personalized recommendation, e.g., HRNN [52] exceeds GRU4Rec by 3.5% with user representation.

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

4.4 Model Training

A well-designed training strategies can 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 (see Section 3.2) and *uniform sampling* have been widely used in recommendation. [45] 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. 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 was proposed in [43] to accommodate sessions of varying lengths and strive to capture the dynamics of sessions over time. It has two variants: the first one is *item boosting* [44]. Some items can be repeatedly used in mini-batch in terms of identified factors like the dwell time. The other one is *user-parallel mini-batch*. For example, HRNN [52] 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 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 5, 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 regularization.

$$L_{\text{TOP1}} = \frac{1}{N_S} \sum_{j=1}^{N_S} \sigma(r_j - r_i) + \sigma(r_j^2) \tag{5}$$

where $\sigma(.)$ is a sigmoid function, r_i and r_j is the ranking score for sample i and j respectively. Following the same notations, **BPR** (Bayesian Personalized Ranking) [16] is defined as:

$$L_{\text{BPR}} = -\frac{1}{N_S} \sum_{j=1}^{N_S} \log \sigma(r_i - r_j) \tag{6}$$

TOP1 and BPR loss functions might suffer from the gradients vanishing problems for DL-based models (e.g., in GRU4Rec [45]). In this view, ranking-max loss function family is proposed [45] 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 7 and 8 respectively. In this case, TOP1-max and BRP-max can be considered as weighted version of TOP1 and BPR. Previous research proves that the two loss functions largely improve the performance for RNN-based sequential recommendation models [45].

$$L_{\text{TOP1-max}} = \sum_{j=1}^{N_S} s_j \left(\sigma(r_j - r_i) + \sigma(r_j^2)\right) \tag{7}$$

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

$$L_{\text{BPR-max}} = -\log \sum_{j=1}^{N_S} s_j \sigma(r_i - r_j) \tag{8}$$

In addition to ranking-based loss functions, *CCE* and *Hinge* loss functions have also been applied in sequential recommendation [57]. **CCE** is defined as:

$$\text{CCE}(\mathbf{o}, i) = \log(\text{softmax}(\mathbf{o})_i) \tag{9}$$

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

$$\text{Hinge}(\mathbf{o}, i) = \sum_{j \in C} \max(0, 1 - o_j) - \gamma \sum_{j \in F} \max(0, o_j) \tag{10}$$

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 that a recommender decides whether an item should be recommended or not.

5 Empirical Studies on Influential Factors

In this section, we conduct experiments on real datasets to further comprehensively evaluate the impact of influential factors on DL-based models.

5.1 Experimental Settings

5.1.1 Datasets

We conduct the experiments on three real-world datasets: *RSC15*, *RSC19* and *LastFM*. *RSC15* is published by RecSys Challenge 2015[10], which contains click and buy behaviors from an online shop, where only the click data is used in this evaluation. *RSC19* is published by RecSys Challenge 2019[11], which contains hotel search sessions from a global hotel platform. *RSC19 (user)* is a subset of *RSC19*. We select the users with more than 10 sessions, and consider the last session of each user as testing set. *LastFM* is collected via the LastFM API, and each sample is presented by a 4-tuple (user, artist, song, timestamp). Due to the lack of session identities in LastFM, we manually divide the sequence of each user into sessions every 30 minutes. The statistic information of these datasets are summairzed in Table 2.

5.1.2 Model settings

We choose GRU4Rec [43] (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 sequential recommendation. The *default* parameters for **basic** model is no data augmentation, no user representation, BRP-max loss function, uniform

10. www.kaggle.com/chadgostopp/recsys-challenge-2015
11. www.recsyschallenge.com/2019/

TABLE 2: 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

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

TABLE 3: Other parameters settings for different scenarios.

Model	RSC15		
	Batch Size	Lr	RNN Size
Default	32	0.2	100
GRU4Rec (Category)	50	0.001	100
C-GRU	50	0.001	140
P-GRU	50	0.001	70
NARM	512	0.001	100
	RSC19		
	Batch Size	Lr	RNN Size
Default	32	0.2	100
GRU4Rec (Behavior)	50	0.001	100
B-GRU	50	0.001	100
User Implicit	50	0.001	50
User Embedded	50	0.001	50
User Recurrent	100	0.001	50
	LastFM		
	Batch Size	Lr	RNN Size
Default	50	0.001	50
User Recurrent	200	0.02	50

negative sampling with a sample size of $2,048$ for *RSC15* and 128 for *RSC19* in terms of the dataset size. In the next experiments, if not particularly figure 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, we implement two improved versions of the basic model: **C-GRU** (concatenating item embedding with category embedding) and **P-GRU** (parellelly training two basic models for item and category respectively, and then concatenating the output of the two subnets) with mini-batch parallel sampling (batch size = 50). For the dwell time, we implement the model in [44], 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 an behavior type module to the basic model. Specifically, B-GRU takes both the item one-hot 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. Besides, B-GRU uses uniform negative sampling method with a sample size of 32. The structure of **C-GRU**, **P-GRU** and **B-GRU** are shown in Figure 12.

For the data processing module, we implement the **data augmentation** method with the basic model. Specifically, we randomly select 50% sessions in the training set and randomly treat a part of each session as new sessions.

For the model structure module, we consider three structures: **NARM** [28] (*incorporating the basic model with the attention mechanism*), **weighted model** in [13] (*combining the DL model with KNN*) and **adding an explicit user representation** in two ways (i.e., the model in [52] and the second one adding a user embedding part based on user IDs, which is concatenated with the output of GRU in basic model). Besides, for the last one, we have a small dataset from RSC19 (**RSC19 (user)** in Table 2) which selects users who have substantial historical data for representation learning. Moreover, the size of negative samples is 32 and we use user-parallel mini-batches in training.

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 [45]), and $\alpha \in \{0, 0.25, 0.5, 0.75, 1\}$), and the size of negative samples $(0, 32, 128, 512, 2,048)$. Other parameters in terms of different datasets are summarized in Table 3, where **Lr** refers to learning rate of these DL models.

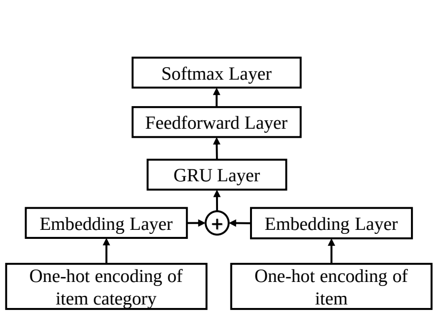
5.1.3 Evaluation metrics

To compare the performance of different models, we use **Recall@k** and **MRR@k** (mean reciprocal rank) as previous sequential recommendation models. Besides, we also use another two commonly used ranking metrics **MAP@k** (mean average precision) and **NDCG@k** (normalized discounted cumulative gain [97]) where k is set to 5, 10 and 20 respectively. For these four metrics, a larger value implies better performance. We refer interesting readers to [30] for detailed definitions of Recall and MRR evaluation metrics. Noted that GRU4Rec is to predict a user's next behavior, i.e., only one item in the recommendation list will be actually selected by the user. However, in real applications, the user can select multiple items in the top-k recommendation list [25]. Therefore, we compute MAP and NDCG in a way that users can click multiple items in the future, and the formulations are as follows (the ground-truth item refers to the target item and the ground-truth list we used for calculating MAP and NDCG are multiple items in the corresponding session after the input item):

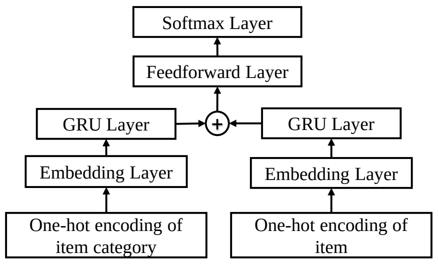
$$\begin{aligned} \text{AP@}k(m) &= \frac{1}{m_g}\sum_{i=1}^{k}\frac{P(i)I_m(i)}{i}, \\ \text{MAP@}k &= \frac{1}{|M|}\sum_{m\in M}\text{AP@}k(m) \end{aligned} \tag{11}$$

where m_g denotes the size of the ground-truth list of item m (i.e., M_m), which refers to the sequence after m in the current session. $I_m(\cdot)$ is an indicator function. If the ith item in the top-k recommendation is in the M_m, it returns 1, otherwise 0. $P(\cdot)$ is a counting function. If the ith item is in M_m, it will be added by 1.

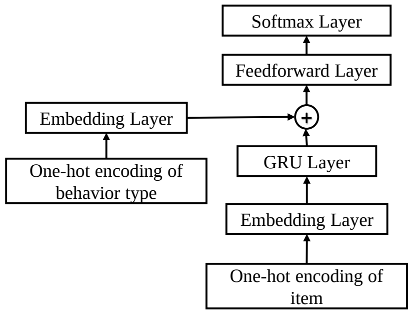
$$\text{NDCG@}k = \frac{\text{DCG@}k}{\text{IDCG@}k}, \text{DCG@}k = \sum_{g\in G}\frac{s_g I_k(g)}{\log_2(i_g+1)} \tag{12}$$



(a) C-GRU



(b) P-GRU

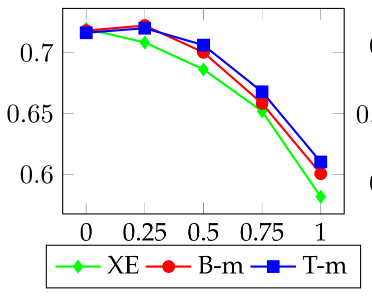


(c) B-GRU

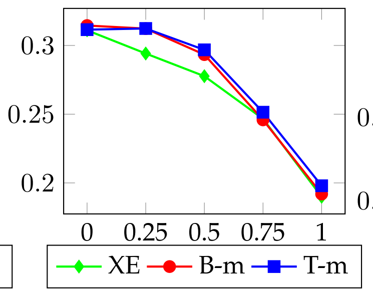
Fig. 12: Network structures of C-GRU, P-GRU and B-GRU.

TABLE 4: Results of incorporating item category or behavior type.

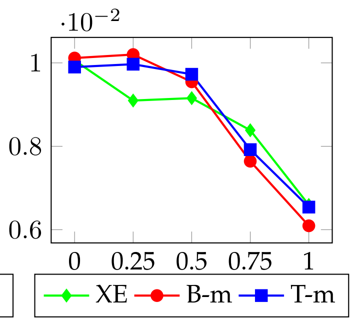
Model	RSC15				Model	RSC19			
	Recall@20	MRR@20	MAP@20	NDCG@20		Recall@20	MRR@20	MAP@20	NDCG@20
GRU4Rec	0.53621	0.19788	0.00742	0.04701	GRU4Rec	0.60346	0.38475	**0.00275**	**0.01775**
C-GRU	**0.54664**	0.19832	0.00884	0.05318	B-GRU	**0.61484**	**0.38901**	0.00216	0.01428
P-GRU	0.54356	**0.20483**	**0.00887**	**0.05322**					



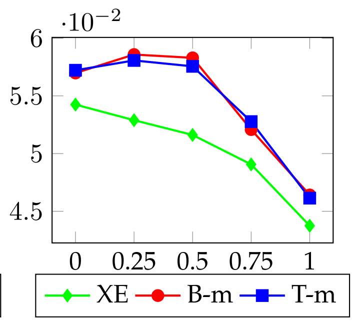
(a) Recall@20



(b) MRR@20



(c) MAP@20



(d) NDCG@20

Fig. 13: The impact of α on additional sampling strategy on RSC15 (XE: cross-entropy; B-m: BPX-max; T-m: TOP1-max).

where G denotes the ground-truth list, s_g represents the predicted score of item g in G. i_g is the index of g in G. $I_k(\cdot)$ is an indicator function which returns 1 if item g is in top-k recommendation, otherwise 0. IDCG is the DCG of ideal ground-truth list which refers to the descending ranking of ground-truth list in terms of predicted scores.

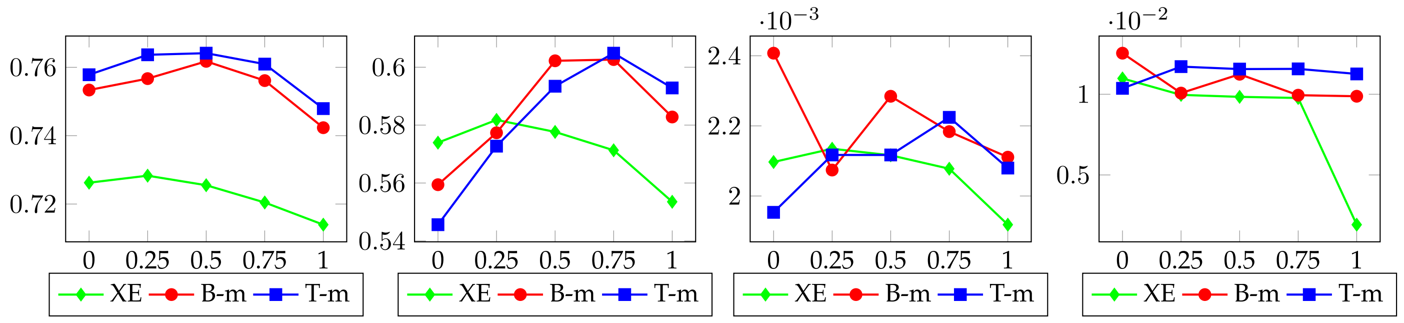
It should be noted that *Recall* measures the coverage of the corrected recommended items in terms of ground-truth items. *MRR* measures how well a model ranks the target items. High *MAP* indicates that items in ground-truth list appear at a higher ranking orders in the top-k recommendation list. *NDCG* measures the quality of the top-k recommendation, where high *NDCG* implies that the order in which an item appear in the top-k recommendation list is close to its order in ground-truth list. In real world applications, different metrics would be selected in terms of the requirements.

5.2 Experiment Results

In this section, we present the experimental results of different factors.

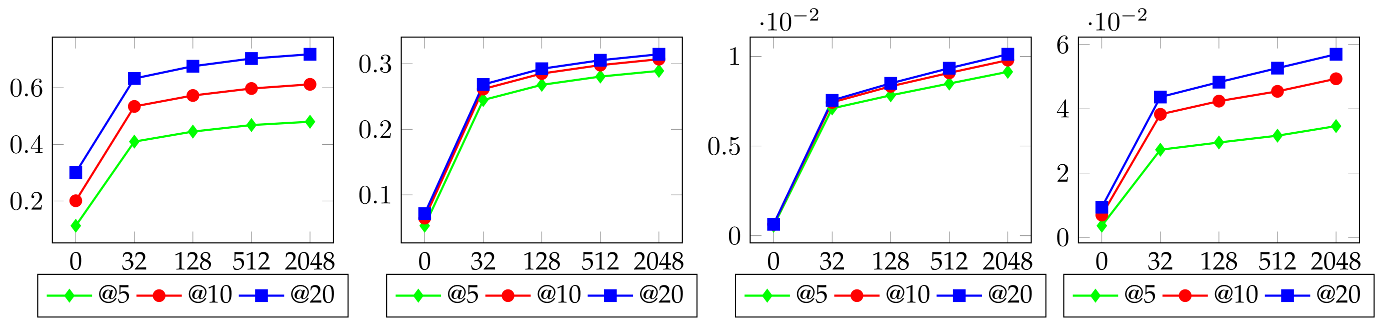
Side information effects. Tables 4 and 5 show the results of the two types of the side information on DL-based model respectively. As shown in Table 4, incorporating **item category** information into GRU4Rec can improve the model performance in terms of the four metrics. Specifically, C-GRU and P-GRU perform better than the basic model. As we can see in Table 5, **dwell time** can greatly improve the performance, e.g., Recall@20 increases by about 23% and 19% on *RSC15* and *RSC19* dataset 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 of impact of **behavior types** (*B-GRU*) are presents in Table 4. We can see that B-GRU outperforms the basic model on Recall@20, MRR@20, and performs worse on MAP@20 and NDCG@20, and the differences are insignificant. The main reason might be that *RSC19* only contains four behavior types and one of them accounts for 62%. In this case, without a careful



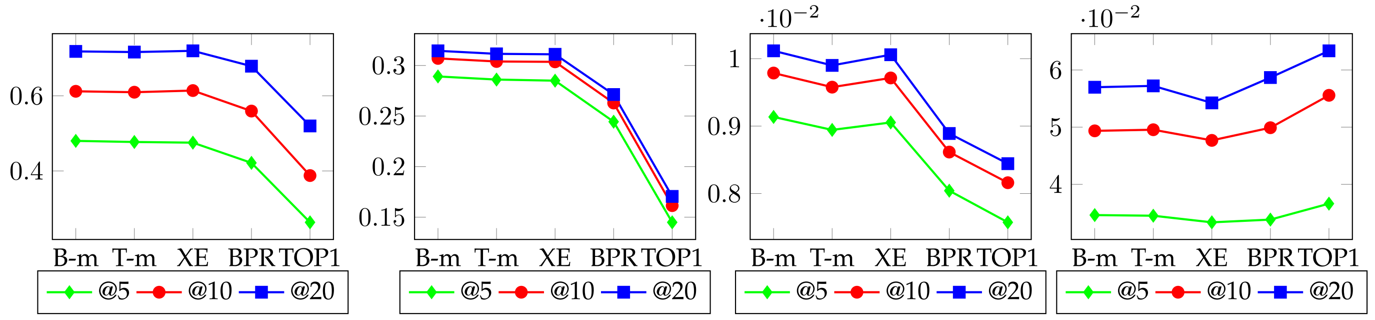
(a) Recall@20 (b) MRR@20 (c) MAP@20 (d) NDCG@20

Fig. 14: The impact of α on additional sampling strategy on RSC19 (XE: cross-entropy; B-m: BPX-max; T-m: TOP1-max).



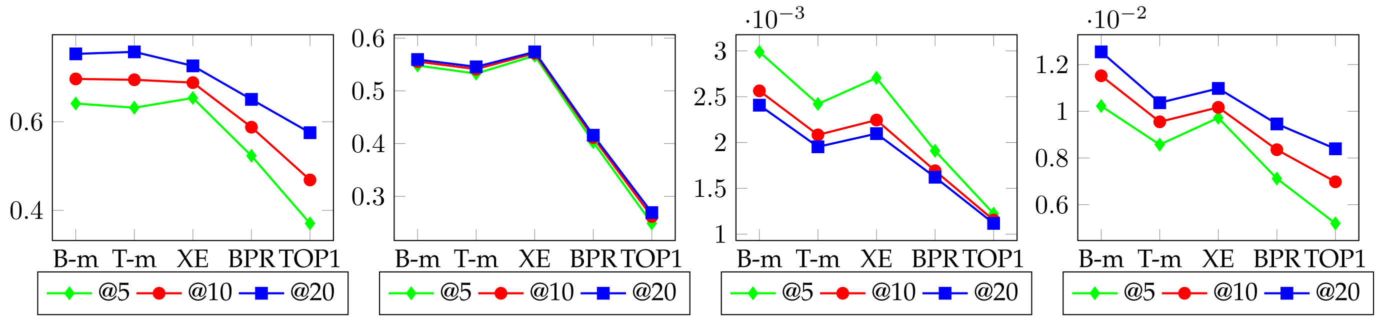
(a) Recall (b) MRR (c) MAP (d) NDCG

Fig. 15: The effect of sample size on RSC15.



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

Fig. 16: Model performance for different loss functions on RSC15 (B-m: BPR-max; T-m: Top1-max; XE: cross-entropy).



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

Fig. 17: Model performance for different loss functions on RSC19 (B-m: BPR-max; T-m: Top1-max; XE: cross-entropy).

TABLE 5: Results of different factors.

Factor	Variable	RSC15				RSC19			
		Recall@20	MRR@20	MAP@20	NDCG@20	Recall@20	MRR@20	MAP@20	NDCG@20
Dwell time	0	0.71820	0.31448	**0.01012**	0.05698	0.75335	0.55942	**0.00241**	0.01254
	(75, 45)	**0.88276**	**0.70885**	0.00491	0.07217	**0.89598**	**0.78898**	0.00109	**0.01442**
	(100, 60)	0.86111	0.65478	0.00579	**0.07380**	0.87224	0.75365	0.00116	0.01195
Data augmentation	Off	0.71820	0.31448	0.01012	**0.05698**	0.75335	0.55942	**0.00241**	**0.01254**
	On	**0.71836**	**0.31493**	**0.01013**	0.05692	**0.75638**	**0.56547**	0.00223	0.01075
Attention mechanism	Off	0.67886	0.27126	**0.00889**	**0.05868**	0.65055	0.41590	0.00162	**0.00946**
	On	**0.69827**	**0.30292**	0.00878	0.05542	**0.65623**	**0.41735**	**0.00164**	0.00885
KNN weight	0	0.71820	0.31448	0.01012	**0.05698**	0.75335	0.55942	**0.00241**	**0.01254**
	0.1	0.72022	**0.31547**	0.01308	0.05183	0.75675	0.56576	0.00128	0.00689
	0.3	**0.72307**	0.31315	**0.01340**	0.05206	**0.76662**	**0.57872**	0.00132	0.00696
Factor	Variable	LastFM				RSC19 (user)			
		Recall@20	MRR@20	MAP@20	NDCG@20	Recall@20	MRR@20	MAP@20	NDCG@20
User Representation	Implicit	**0.16996**	**0.12496**	0.00408	0.08126	**0.67981**	**0.56814**	0.01452	0.08368
	Embedded	0.01634	0.00436	0.00837	0.21537	0.00479	0.00378	0.00773	0.20750
	Recurrent	0.00346	0.00058	**0.01230**	**0.42749**	0.06276	0.03058	**0.04508**	**0.79612**

design, considering other supporting behaviors for the target behavior might be probably incorporated noisy information.

Data augmentation effects. As shown in Table 5, the model with *data argumentation* outperforms the basic model in terms of most metrics except NDCG@20: Recall@20 increases by 0.02% on *RSC15* and 0.4% on *RSC19*, while MRR@20 improves 0.14% and 1.1% on *RSC15* and *RSC19* respectively.

Model structure effects. As shown in Table 5, incorporating **attention mechanism** enhances the performance of the model almost for all the scenarios (except NDCG). Combing the basic model with **KNN** improves model performance in terms of Recall@20 and MRR@20 on both *RSC15* and *RSC19*, and KNN weight of 0.3 provides better performance than that of 0.1, manifesting the way of combining traditional models with DL-models can have a significant effect on sequential recommendation. For **user representation**, we find that adding an explicit user representation module, whether embedded or recurrent one, leads to a sharp decrease on Recall@20 and MRR@20. The main reasons might be three-folds: 1) session-parallel mini-batch (a session as a sample) is used for user 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 less than user implicit model; 2) as shown in Table 2, No. of sessions is much greater than that of users on these two datasets; 3) according to No. of items and behaviors shown in Table 2, the average support of items is much smaller on these two datasets. On the other hand, in terms of NDCG@20 and MAP@20, user representation models greatly outperform the basic model, while the user recurrent model performs better than the user embedded model. In this case, in personalized sequential recommendation, the user recurrent model is suggested for better learning users' long-term preferences. However, whether considering explicit user representation model is largely dependent on the application scenarios.

Sampling method effects. Figures 13 and 14 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 BPR-max and TOP1-max (on Recall@20, MAP@20 and NDCG@20) firstly slowly increases as α increases, but decreases quickly as α is larger than 0.25. Meanwhile, the performance in terms of cross-entropy loss function is steadily declines with the increase α. 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 TOP1-max that is 0.25. In this case, it is necessary to carry out 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 15, 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. It should be noted that additional negative sampling leads to higher computing costs. Therefore, in real world applications, we need to keep the balance between model performance and training time considering the size of negative samples.

Loss function effects. As depicted in Figures 16 and 17, models with **loss functions** BPR-max, TOP1-max, and cross-entropy perform better than those with BPR and TOP1 in terms of all metrics (except NDCG), 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.

Concluding remarks. The experimental results on the real datasets verify that the summarized influential factors in Section 4 all play an important role in DL-based sequential recommendation models. Our suggestions for best in practice are summarized as follows: 1) try all possible side information (such as texts and images), 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 data argumentation, TOP1-max, BPR-max and cross-entroy loss functions for training, keep a balance between model performance and computing cost with regard to size of negative sample; and 4) for any DL-based models, consider to further improve their performance with attention mechanism, by possibly combing with traditional sequential learning, and a well explicit user representation.

6 Open Issues and Future Directions

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

Objective and comprehensive evaluations across different models. Our empirical study can be considered as a horizontal investigation across GRU4Rec and its variants. In the literature, lots of GRU4Rec variants consider GRU4Rec [43] 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 [61] (CNN-based model) or further refer to the comparison framework proposed in [9].

More designs on embedding methods. Most previous studies adopt the embedding methods from NLP. However, in sequential recommendation, it is rather challenging to pre-train an embedding model (e.g., word2vec) as the information is constantly changing while the words and their connections in NLP are relatively fixed. Besides, 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 sequential recommendation.

Advanced sampling strategies. In sequential recommendation, most existing works 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).

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.

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 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 sequential recommendation. For example, Qiu et al. [38] proposed a Bayesian personalized ranking model for heterogeneous behavior types (BPRH) that incorporated target behavior, auxiliary behavior, and negative behavior into a unified model, and the idea might be applicable for sequential recommendation.

Learning behavior sequences in real time. Every behavior of the user might reflect a possible interest transfer, in this case, recommendation systems are expected to ideally capture these kind of information and timely justify the recommendation strategies. Reinforcement learning is a promising choice for addressing this issue. For example, Zhao et al. [21] combined MF, RNN, and GAN in film recommendations to dynamically provide movie recommendations. Shih et al. [22] 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.

Sequential recommendation for specific 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 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 useful for real-world applications.

Acknowledgments

We gratefully acknowledge the support of National Natural Science Foundation of China (Grant No. 71601104, 71601116, 71771141 and 61702084). This work is also supported by the Fundamental Research Funds for the Central Universities under Grant No. N181705007.

References

[1] J. Davidson, B. Liebald, J. Liu, P. Nandy, T. V. Vleet, U. Gargi, S. Gupta, Y. He, M. Lambert, and B. Livingston, "The youtube video recommendation system," in *RecSys*, 2010, pp. 293–296.

[2] R. He and J. McAuley, "Fusing similarity models with markov chains for sparse sequential recommendation," in *ICDM*, 2016, pp. 191–200.

[3] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," *ACM Computing Surveys (CSUR)*, vol. 52, no. 1, p. 5, 2019.

[4] A. Singhal, P. Sinha, and R. Pant, "Use of deep learning in modern recommendation system: A summary of recent works," *arXiv preprint arXiv:1712.07525*, 2017.

[5] Z. Batmaz, A. I. Yurekli, A. Bilge, and C. Kaleli, "A review on deep learning for recommender systems: challenges and remedies," *Artificial Intelligence Review*, pp. 1–37, 2018.

[6] M. Quadrana, P. Cremonesi, and D. Jannach, "Sequence-aware recommender systems," *arXiv preprint arXiv:1802.08452*, 2018.

[7] S. Wang, L. Cao, and Y. Wang, "A survey on session-based recommender systems," *arXiv preprint arXiv:1902.04864*, 2019.

[8] Y. K. Tan, X. Xu, and Y. Liu, "Improved recurrent neural networks for session-based recommendations," in *DLRS*, 2016, pp. 17–22.

[9] M. Ludewig and D. Jannach, "Evaluation of session-based recommendation algorithms," *arXiv preprint arXiv:1803.09587*, 2018.

[10] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, no. 1, pp. 76–80, 2003.

[11] G. Sottocornola, P. Symeonidis, and M. Zanker, "Session-based news recommendations," in *WWW'18*, 2018, pp. 1395–1399.

[12] L. Lerche, D. Jannach, and M. Ludewig, "On the value of reminders within e-commerce recommendations," in *UMAP*, 2016, pp. 27–35.

[13] D. Jannach and M. Ludewig, "When recurrent neural networks meet the neighborhood for session-based recommendation," in *RecSys*, 2017, pp. 306–310.

[14] R. He, W.-C. Kang, and J. McAuley, "Translation-based recommendation," in *RecSys*, 2017, pp. 161–169.

[15] R. He, C. Fang, Z. Wang, and J. McAuley, "Vista: A visually, socially, and temporally-aware model for artistic recommendation," in *RecSys*, 2016, pp. 309–316.

[16] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," in *UAI*, 2009, pp. 452–461.

[17] B. Twardowski, "Modelling contextual information in session-aware recommender systems with neural networks," in *RecSys*, 2016, pp. 273–276.

[18] S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme, "Fast context-aware recommendations with factorization machines," in *SIGIR*, 2011, pp. 635–644.

[19] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, "Factorizing personalized markov chains for next-basket recommendation," in *WWW*, 2010, pp. 811–820.

[20] S. Kabbur, X. Ning, and G. Karypis, "Fism: factored item similarity models for top-n recommender systems," in *SIGKDD*, 2013, pp. 659–667.

[21] W. Zhao, W. Wang, J. Ye, Y. Gao, M. Yang, Z. Zhao, and X. Chen, "Leveraging long and short-term information in content-aware movie recommendation," *arXiv preprint arXiv:1712.09059*, 2017.

[22] S.-Y. Shih and H.-Y. Chi, "Automatic, personalized, and flexible playlist generation using reinforcement learning," *arXiv preprint arXiv:1809.04214*, 2018.

[23] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[24] T. X. Tuan and T. M. Phuong, "3d convolutional networks for session-based recommendation with content features," in *RecSys*, 2017, pp. 138–146.

[25] J. Tang and K. Wang, "Personalized top-n sequential recommendation via convolutional sequence embedding," in *WSDM*, 2018, pp. 565–573.

[26] C. Wu and M. Yan, "Session-aware information embedding for e-commerce product recommendation," in *CIKM*, 2017, pp. 2379–2382.

[27] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *ICLR*, 2015.

[28] J. Li, P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma, "Neural attentive session-based recommendation," in *CIKM*, 2017, pp. 1419–1428.

[29] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *ICONIP*, 2017, pp. 5998–6008.

[30] S. Zhang, Y. Tay, L. Yao, and A. Sun, "Next item recommendation with self-attention." *arXiv: Information Retrieval*, 2018.

[31] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, and M. Sun, "Graph neural networks: a review of methods and applications," *arXiv preprint arXiv:1812.08434*, 2018.

[32] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, "Session-based recommendation with graph neural networks," *arXiv preprint arXiv:1811.00855*, 2018.

[33] J. Huang, W. X. Zhao, H. Dou, J.-R. Wen, and E. Y. Chang, "Improving sequential recommendation with knowledge-enhanced memory networks," in *SIGIR*, 2018, pp. 505–514.

[34] A. P. Singh and G. J. Gordon, "Relational learning via collective matrix factorization," in *SIGKDD*, 2008, pp. 650–658.

[35] A. Krohn-Grimberghe, L. Drumond, C. Freudenthaler, and L. Schmidt-Thieme, "Multi-relational matrix factorization using bayesian personalized ranking for social network data," in *WSDM*, 2012, pp. 173–182.

[36] Z. Zhao, Z. Cheng, L. Hong, and E. H. Chi, "Improving user topic interest profiles by behavior factorization," in *WWW*, 2015, pp. 1406–1416.

[37] B. Loni, R. Pagano, M. Larson, and A. Hanjalic, "Bayesian personalized ranking with multi-channel user feedback," in *RecSys*, 2016, pp. 361–364.

[38] H. Qiu, Y. Liu, G. Guo, Z. Sun, J. Zhang, and H. T. Nguyen, "Bprh: Bayesian personalized ranking for heterogeneous implicit feedback," *Information Sciences*, vol. 453, pp. 80–98, 2018.

[39] G. Guo, H. Qiu, Z. Tan, Y. Liu, J. Ma, and X. Wang, "Resolving data sparsity by multi-type auxiliary implicit feedback for recommender systems," *Knowledge-Based Systems*, vol. 138, pp. 202–207, 2017.

[40] J. Ding, G. Yu, X. He, Y. Quan, Y. Li, T.-S. Chua, D. Jin, and J. Yu, "Improving implicit recommender systems with view data." in *IJCAI*, 2018, pp. 3343–3349.

[41] C. Gao, X. He, D. Gan, X. Chen, F. Feng, Y. Li, T.-S. Chua *et al.*, "Learning recommender systems from multi-behavior data," *arXiv preprint arXiv:1809.08161*, 2018.

[42] Q. Xia, P. Jiang, F. Sun, Y. Zhang, X. Wang, and Z. Sui, "Modeling consumer buying decision for recommendation based on multi-task deep learning," in *CIKM*, 2018, pp. 1703–1706.

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

[44] V. Bogina and T. Kuflik, "Incorporating dwell time in session-based recommendations with recurrent neural networks," in *CEUR Workshop Proceedings*, 2017, pp. 57–59.

[45] B. Hidasi and A. Karatzoglou, "Recurrent neural networks with top-k gains for session-based recommendations," in *CIKM*, 2018, pp. 843–852.

[46] B. Hidasi, M. Quadrana, A. Karatzoglou, and D. Tikk, "Parallel recurrent neural network architectures for feature-rich session-based recommendations," in *RecSys*, 2016, pp. 241–248.

[47] Y. Zhang, H. Dai, C. Xu, J. Feng, T. Wang, J. Bian, B. Wang, and T. Liu, "Sequential click prediction for sponsored search with recurrent neural networks," *national conference on artificial intelligence*, pp. 1369–1375, 2014.

[48] H. Soh, S. Sanner, M. White, and G. Jamieson, "Deep sequential recommendation for personalized adaptive user interfaces," in *IUI*, 2017, pp. 589–593.

[49] C.-Y. Wu, A. Ahmed, A. Beutel, A. J. Smola, and H. Jing, "Recurrent recommender networks," in *WSDM*, 2017, pp. 495–503.

[50] H. Bharadhwaj and S. Joshi, "Explanations for temporal recommendations," *Künstliche Intelligenz*, vol. 32, no. 4, pp. 267–272, 2018.

[51] X. Chen, H. Xu, Y. Zhang, J. Tang, Y. Cao, Z. Qin, and H. Zha, "Sequential recommendation with user memory networks," in *WSDM*, 2018, pp. 108–116.

[52] M. Quadrana, A. Karatzoglou, B. Hidasi, and P. Cremonesi, "Personalizing session-based recommendations with hierarchical recurrent neural networks," in *RecSys*, 2017, pp. 130–137.

[53] T. Donkers, B. Loepp, and J. Ziegler, "Sequential user-based recurrent neural network recommendations," in *RecSys*, 2017, pp. 152–160.

[54] Q. Liu, S. Wu, D. Wang, Z. Li, and L. Wang, "Context-aware sequential recommendation," *ICDM*, pp. 1053–1058, 2016.

[55] Y. Song and J.-G. Lee, "Augmenting recurrent neural networks with high-order user-contextual preference for session-based recommendation," *arXiv preprint arXiv:1805.02983*, 2018.

[56] F. Yu, Q. Liu, S. Wu, L. Wang, and T. Tan, "A dynamic recurrent model for next basket recommendation," in *SIGIR*, 2016, pp. 729–732.

[57] R. Devooght and H. Bersini, "Long and short-term recommendations with recurrent neural networks," in *UMAP*, 2017, pp. 13–21.

[58] M. Ruocco, O. S. L. Skrede, and H. Langseth, "Inter-session modeling for session-based recommendation," in *DLRS*, 2017, pp. 24–31.

[59] T. Shi, Y. Keneshloo, N. Ramakrishnan, and C. K. Reddy, "Neural abstractive text summarization with sequence-to-sequence models," *arXiv preprint arXiv:1812.02303*, 2018.

[60] K.-C. Hsu, S.-Y. Chou, Y.-H. Yang, and T.-S. Chi, "Neural network based next-song recommendation," *arXiv preprint arXiv:1606.07722*, 2016.

[61] F. Yuan, A. Karatzoglou, I. Arapakis, J. M. Jose, and X. He, "A simple convolutional generative network for next item recommendation," *web search and data mining*, pp. 582–590, 2019.

[62] S. Wang, L. Hu, L. Cao, X. Huang, D. Lian, and W. Liu, "Attention-based transactional context embedding for next-item recommendation," in *AAAI*, 2018, pp. 2532–2539.

[63] Q. Liu, Y. Zeng, R. Mokhosi, and H. Zhang, "Stamp: short-term attention/memory priority model for session-based recommendation," in *SIGKDD*, 2018, pp. 1831–1839.

[64] P. Ren, Z. Chen, J. Li, Z. Ren, J. Ma, and M. de Rijke, "Repeatnet: A repeat aware neural recommendation machine for session-based recommendation," *arXiv preprint arXiv:1812.02646*, 2018.
[65] N. Sachdeva, K. Gupta, and V. Pudi, "Attentive neural architecture incorporating song features for music recommendation," in *RecSys*, 2018, pp. 417–421.
[66] T. Bai, J.-Y. Nie, W. X. Zhao, Y. Zhu, P. Du, and J.-R. Wen, "An attribute-aware neural attentive model for next basket recommendation," in *SIGIR*, 2018, pp. 1201–1204.
[67] W. Kang and J. Mcauley, "Self-attentive sequential recommendation." *arXiv: Information Retrieval*, 2018.
[68] J. W. C. P. X. L. W. O. Fei Sun, Jun Liu and P. Jiang, "Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer," *arXiv preprint arXiv:1904.06690*, 2019.
[69] S. Wan, Y. Lan, P. Wang, J. Guo, J. Xu, and X. Cheng, "Next basket recommendation with neural networks." in *RecSys Posters*, 2015.
[70] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, no. 6, pp. 1137–1155, 2003.
[71] P. Wang, J. Guo, Y. Lan, J. Xu, S. Wan, and X. Cheng, "Learning hierarchical representation model for next basket recommendation," in *SIGIR*, 2015, pp. 403–412.
[72] N. Sachdeva, G. Manco, E. Ritacco, and V. Pudi, "Sequential variational autoencoders for collaborative filtering," *WSDM*, 2019.
[73] D.-T. Le, H. W. Lauw, and Y. Fang, "Modeling contemporaneous basket sequences with twin networks for next-item recommendation," in *IJCAI*, 2018, pp. 3414–3420.
[74] Z. Li, H. Zhao, Q. Liu, Z. Huang, T. Mei, and E. Chen, "Learning from history and present: next-item recommendation via discriminatively exploiting user behaviors," in *SIGKDD*, 2018, pp. 1734–1743.
[75] O. Barkan and N. Koenigstein, "Item2vec: neural item embedding for collaborative filtering," in *MLSP*, 2016, pp. 1–6.
[76] Q. Liu, S. Wu, and L. Wang, "Multi-behavioral sequential prediction with recurrent log-bilinear model," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 6, pp. 1254–1267, 2017.
[77] E. Smirnova and F. Vasile, "Contextual sequence modeling for recommendation with recurrent neural networks," in *DLRS*, 2017, pp. 2–9.
[78] C. Zhou, J. Bai, J. Song, X. Liu, Z. Zhao, X. Chen, and J. Gao, "Atrank: An attention-based user behavior modeling framework for recommendation," *arXiv preprint arXiv:1711.06632*, 2017.
[79] X. Huang, S. Qian, Q. Fang, J. Sang, and C. Xu, "Csan: Contextual self-attention network for user sequential recommendation," in *MM*, 2018, pp. 447–455.
[80] P. Loyola, C. Liu, and Y. Hirate, "Modeling user session and intent with an attention-based encoder-decoder architecture," in *RecSys*, 2017, pp. 147–151.
[81] M. Ma, S. Na, C. Xu, and X. Fan, "The graph-based broad behavior-aware recommendation system for interactive news," *arXiv preprint arXiv:1812.00002*, 2018.
[82] T. Bansal, D. Belanger, and A. Mccallum, "Ask the gru : multi-task learning for deep text recommendations," *conference on recommender systems*, pp. 107–114, 2016.
[83] P. Covington, J. Adams, and E. Sargin, "Deep neural networks for youtube recommendations," in *RecSys*, 2016, pp. 191–198.
[84] H. T. Nguyen, M. Wistuba, J. Grabocka, L. R. Drumond, and L. Schmidt-Thieme, "Personalized deep learning for tag recommendation," in *PAKDD*, 2017, pp. 186–197.
[85] Q. Zhang, J. Wang, H. Huang, X. Huang, and Y. Gong, "Hashtag recommendation for multimodal microblog using co-attention network." in *IJCAI*, 2017, pp. 3420–3426.
[86] Y. S. Rawat and M. S. Kankanhalli, "Contagnet: Exploiting user context for image tag recommendation," in *ACMMM*, 2016, pp. 1102–1106.
[87] H. Wang, X. Shi, and D. Y. Yeung, "Collaborative recurrent autoencoder: Recommend while learning to fill in the blanks," *neural information processing systems*, pp. 415–423, 2016.
[88] S. Li, J. Kawale, and Y. Fu, "Deep collaborative filtering via marginalized denoising auto-encoder," in *CIKM*, 2015, pp. 811–820.
[89] H. Wang, N. Wang, and D. Y. Yeung, "Collaborative deep learning for recommender systems," *knowledge discovery and data mining*, pp. 1235–1244, 2015.
[90] J. Wei, J. He, K. Chen, Y. Zhou, and Z. Tang, "Collaborative filtering and deep learning based hybrid recommendation for cold start problem," in *DASC/PiCom/DataCom/CyberSciTech*, 2016, pp. 874–877.
[91] A. Dallmann, A. Grimm, C. Pölitz, D. Zoller, and A. Hotho, "Improving session recommendation with recurrent neural networks by exploiting dwell time," *arXiv preprint arXiv:1706.10231*, 2017.
[92] A. Greenstein-Messica, L. Rokach, and M. Friedman, "Session-based recommendations using item embedding," in *IUI*, 2017, pp. 629–633.
[93] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *EMNLP*, 2014, pp. 1532–1543.
[94] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.
[95] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.
[96] V. Vapnik and A. Vashist, "A new learning paradigm: Learning using privileged information," *Neural networks*, vol. 22, no. 5-6, pp. 544–557, 2009.
[97] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of ir techniques," *ACM Transactions on Information Systems*, vol. 20, no. 4, pp. 422–446, 2002.