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

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

In the field of sequential recommendation, deep learning (DL)-based methods have received a lot of attention in the past few years and surpassed traditional models such as Markov chain-based and factorization-based ones. However, there is little systematic study on DL-based methods, especially regarding to how to design an effective DL model for sequential recommendation. In this view, this survey focuses on DL-based sequential recommender systems by taking the aforementioned issues into consideration. Specifically, we illustrate the concept of sequential recommendation, propose a categorization of existing algorithms in terms of three types of behavioral 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.

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

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

ACM Reference Format:
Hui Fang, Danning Zhang, Yiheng Shu, and Guibing Guo. 2019. Deep Learning for Sequential Recommendation: Algorithms, Influential Factors, and Evaluations. 1, 1 (November 2019), 36 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

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

*Corresponding author

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

XXXX-XXXX/2019/11-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

might not hold in real-world scenarios, where the user's next behavior not only depends on the static long-term preference, but also relies on the current intent to a large extent, which might be probably inferred and influenced by a small set of the most recent interactions. On the other side, the conventional approaches always ignore to consider the sequential dependencies among the user's interactions, leading to inaccurate modeling of the user's preferences. 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 much broader term *sequential* recommendation to describe the task that explores the sequential data. For sequential recommendation, besides capturing users' long-term preferences across different sessions as the conventional recommendation does, it is also extremely important to simultaneously model users' short-term interest in a session (or a short sequence) for accurate recommendation. Regarding the time dependency among different interactions in a session as well as the correlation of behavior patterns among different sessions, traditional sequential recommender systems are particularly interested in employing appropriate and effective machine learning (ML) approaches to model sequential data, such as Markov Chain [15] and session-based KNN [25], which are criticized by their incomplete modeling problem, as they fail to thoroughly model users' long-term patterns by combining different sessions.

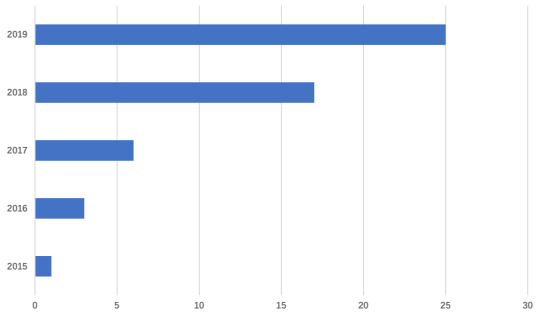


Fig. 1. The number of arXiv articles on DL-based sequential recommendation in recent five years.

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 recommendation, and many DL-based models have achieved state-of-the-art performance [99]. The number of relevant arXiv articles posted 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

[1]We searched on arXiv.org for articles using the related keywords as well as their combinations, such as *sequential recommendation*, *deep learning* and *session* in October 2019.
[2]www.kaggle.com/chadgostopp/recsys-challenge-2015.
[3]labrosa.ee.columbia.edu/millionsong/lastfm.
[4]movielens.org.

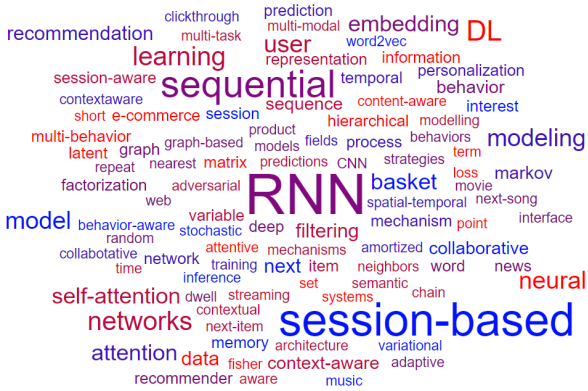


Fig. 2. Word cloud of the keywords for sequential recommendation related articles.

depicts the word cloud of keywords in sequential recommendation related articles, which to some extent reflects the hot topics in the field of sequential recommendation.

More and more new techniques and improved structures have been applied to facilitate the DL-based sequential recommendation. However, we find that the DL-based models still have some common drawbacks. For example, many existing works do not take items and users into consideration at the same important level, i.e., they largely emphasize item representation, but lack of a careful design on user representation. Besides, they merely consider all interactions into one type, instead of distinguishing different types. More importantly, although more advanced techniques have been increasingly adopted, it is still unclear the real progress for this area, as only complex DL structures could not possibly yield better performance [13]. In this case, it becomes extremely necessary to unveil the influential factors leading to a useful DL-based sequential recommender systems. Therefore, we tend to thoroughly discuss the improved techniques and the aforementioned issues in this survey. The contributions of this survey are concluded as follows:

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

1.1 Related Survey

There have been some surveys on either DL-based recommendation or sequential recommendation. For DL-based recommendation, Singhal et al. [68] 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. [99] further discussed the state-of-the-art DL-based recommender systems, including several RNN-based sequential recommendation algorithms. For sequential recommendation, Quadrana et al. [55] proposed a categorization of the recommendation tasks and goals, and summarized existing solutions. Wang et al. [86] illustrated the value and significance of the session-based recommender systems (SBRS), and proposed a hierarchical framework to categorize issues and methods, including some DL-based ones.

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

1.2 Structure of This Survey

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

2 OVERVIEW OF SEQUENTIAL RECOMMENDATION

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

2.1 Concept Definitions

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

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

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

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

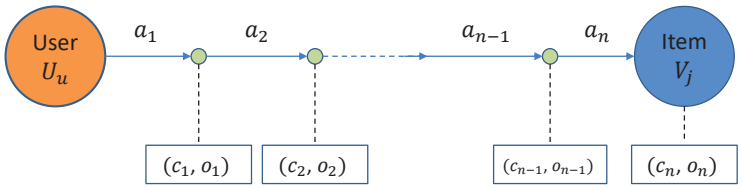


Fig. 3. A schematic diagram of the sequential recommendation. c_i: behavior type, o_i: behavior object. A behavior a_i is represented by a 2-tuple, i.e., $a_i = (c_i, o_i)$. A behavior sequence (i.e., behavior trajectory) is a list of 2-tuples in the order of time.

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

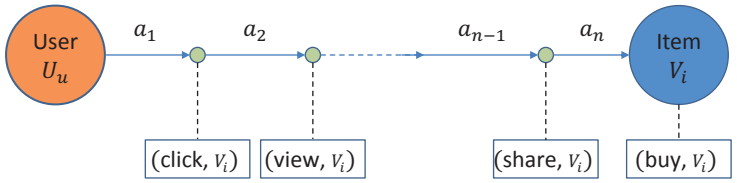


Fig. 4. Experience-based behavior sequence.

Experience-based behavior sequence. In an experience-based behavior sequence (see 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* related keywords, then *clicks* the item of interest on the result pages followed by *viewing* the details of the item. Finally, the user may *share* the item with her friends and *add it to cart* if she likes it. Different behavior types as well as their orders might indicate users' different intentions. For instance, *click* and *view* can only show a user's interest of a low degree, while *share* behavior appears before (or after) *purchase* might imply a user's strong desire (or satisfaction) to obtain (or have) the item. *For this type of behavior sequence, a model is expected to capture a user's underlying intentions indicated by different behavior types. The goal here is to predict the next behavior type that the user will exert given an item.*

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

Interaction-based behavior sequence. An interaction-based behavior sequence could be viewed as a mixture of experience-based and transaction-based behavior sequences (see Fig. 6), i.e., a generalization of previous two types and much closer to the real scenarios. That is to say, it

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

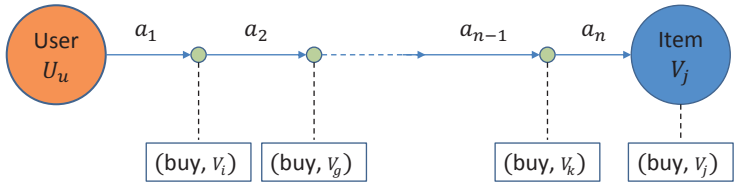


Fig. 5. Transaction-based behavior sequence.

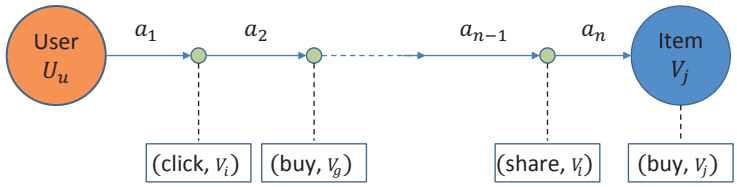


Fig. 6. Interaction-based behavior sequence.

consists of *different behavior objects* and *different behavior types* simultaneously. *In interaction-based behavioral sequence modeling, a recommender system is expected to understand user preferences more realistically, including different user intents expressed by different behavior types and preferences implied by different behavior objects. Its major goal is to predict the next behavior object that a user will interact with.*

2.2 Sequential Recommendation Tasks

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

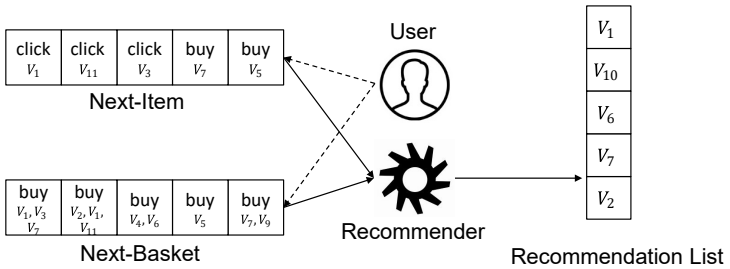


Fig. 7. Next-item and next-basket recommendation.

However, although the input of the aforementioned recommendation tasks is varied, their goals are mostly identical. Specifically, both of them strive to predict the next item(s) for a user, whilst the most popular form of the output is the top-N ranked item list. The rank could be determined by probabilities, absolute values or relative rankings, while in most cases *softmax function* is adopted to generate the output. Tan et al. [76] further proposed an embedding version of the softmax output for fast prediction to accommodate the large volume of items in recommendation.

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

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

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

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

2.3 Related Models

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

2.3.1 Traditional Methods. Conventional popular methods for the sequential recommendation include frequent pattern mining, K-nearest neighbors, Markov chains, matrix factorization, and reinforcement learning [55]. They generally adopt matrix factorization for addressing users' long-term preferences across different sequences, whilst use first-order Markov Chains for capturing users' short-term interest within a sequence [25]. We next introduce the traditional methods as well as the representative algorithms for the sequential recommendation.

Frequent pattern mining. As we know, association rule [49] strives to use frequent pattern mining to mine frequent patterns with sufficient support and confidence. In the sequential recommendation, patterns refer to the sets of items which are frequently co-occurred within a sequence, and then are deployed to make recommendations. Although these approaches are easy to implement, and relatively explicable for users, they suffer from the limited scalability problem as matching patterns for recommendation is extremely strict and time-consuming.

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

K-nearest neighbors (KNN). It includes item-based KNN and session-based KNN for the sequential recommendation. Item-based KNN [15, 42] only considers the last behavior in a given session and recommends items that are most similar to its behavior object (item), where the similarities are usually calculated via the cosine similarity or other advanced measurements [72].

In contrast, session-based KNN [31, 38, 42] compares the entire existing session with all the past sessions to recommend items via calculating similarities using Jaccard index or cosine similarity on binary vectors over the item space. KNN methods can generate highly explainable recommendation. Besides, as the similarities can be pre-calculated, KNN-based recommender systems could generate recommendations promptly. However, this kind of algorithms generally fails to consider the sequential dependency among items.

Markov chains (MC). In the sequential recommendation, Markov models assume that future user behaviors only depend on the last or last few behaviors. For example, [24] merely considered the last behavior with first-order MC, while [23, 25] adopted high-order MCs, which take the dependencies with more previous behaviors into account. Considering only the last behavior or several behaviors makes the MC-based models unable to leverage the dependencies among behaviors in a relatively long sequence and thus fails to capture intricate dynamics of more complex scenarios. Besides, they might also suffer from data sparsity problems.

Factorization-based methods. Matrix factorization (MF) tries to decompose the user-item interaction matrix into two low-rank matrices. For example, BPR-MF [59] optimizes a pairwise ranking objective function via stochastic gradient descent (SGD). Twardowski [79] proposed a MF-based sequential recommender system (a simplified version of Factorization Machines [61]), where only the interaction between a session and a candidate item is considered for generating recommendations. FPMC [60] is a representative baseline for next-basket recommendation, which integrates the MF with first-order MCs. FISM [33] conducts matrix factorization on an item-item matrix, and thus no explicit user representation is learned. On the basis of FISM, FOSSIL [25] tackles the sequential recommendation task by combining similarity-based methods and high-order Markov Chains. It performs better on sparse datasets in comparison with the traditional MC methods and FPMC. The main drawbacks of MF-based methods lie in: 1) most of them only consider the low-order interactions (i.e., first-order and second-order) among latent factors, but ignore the possible high-order interactions; and 2) excepts for a handful of algorithms considering temporal information (e.g., TimeSVD++ [35]), they generally ignore the time dependency among behaviors both within a session and across different sessions.

Reinforcement learning (RL). The essence of RL methods is to update recommendations according to the interactions between users and the recommender systems. When a system recommends an item to a user, a positive reward is assigned if the user expresses her interest on the item (via behaviors such as click or view). It is usually formulated as a Markov decision process (MDP) with the goal of maximizing the cumulative rewards in a set of interactions [66, 101]. With RL frameworks, sequential recommender systems can dynamically adapt to users (changing) preferences. However, similar to DL-based approaches, this kind of works is also lack of interpretability. Besides, more importantly, there is few appropriate platforms or resources for developing and testing RL-based methods in academia .

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

Recurrent neural networks (RNNs). The effectiveness of RNNs in sequence modeling have been widely demonstrated in the field of natural language processing (NLP). In the sequential recommendation, RNN-based models are in the majority of DL-based models [11]. In comparison with the traditional models, RNN-based sequential recommendation models can well capture the dependencies among items within a session or across different sessions. The main limitation of RNNs for the sequential recommendation is that it is relatively difficult to model dependencies in a longer sequence (although could be somehow mitigated by other techniques), and training is burdened with the high cost especially with the increase of sequence length.

Convolutional neural networks (CNNs). CNN is commonly applied to process time series data (e.g., signals) and image data, where a typical structure consists of convolution layers, pooling layers, and feed-forward full-connected layers. It is suitable to capture the dependent relationship across local information (e.g., the correlation between pixels in a certain part of an image or the dependencies between several adjacent words in a sentence). In the sequential recommendation,

CNN-based models can well capture local features within a session, and also could take the time information into consideration in the input layer [77, 78].

Multi-layer perceptrons (MLPs). MLPs refer to feed-forward neural networks with multiple hidden layers, which can thus well learn the nonlinear relationship between the input and output via nonlinear activation functions (e.g., tanh and ReLU). Therefore, MLP-based sequential recommendation models are expected to well capture the complex and nonlinear relationships among users behaviors [90].

Attention mechanisms. Attention mechanism in deep learning is intuited from visual attentions of human-beings (incline to be attracted by more important parts of a target object). It is originated from the work of Bahdanau et al. [1], which proposes an attention mechanism in neural machine translation task to focus on modeling the importance of different parts of the input sentence on the output word. Grounded on the work, *vanilla attention* is proposed by applying the work as a decoder of the RNN, and has been widely used in the sequential recommendation [39]. On the other hand, *self-attention mechanism* (originated in transformer [81] for neural machine translation by Google 2017) has also been deployed in the sequential recommendation. In contrast with vanilla attention, it does not include RNN structures, but performs much better than RNN-based models in recommender systems [98].

Graph neural networks (GNNs). GNN [104] can collectively aggregate information from the graph structure. Due to its effectiveness and superior performance in many applications, it has also obtained increasing interest in recommender systems. For example, Wu et al. [92] first used GNN for session-based recommendation by capturing more complex relationships between items in a sequence, and each session is represented as the composition of the long-term preference and short-term interests within a session using an attention network.

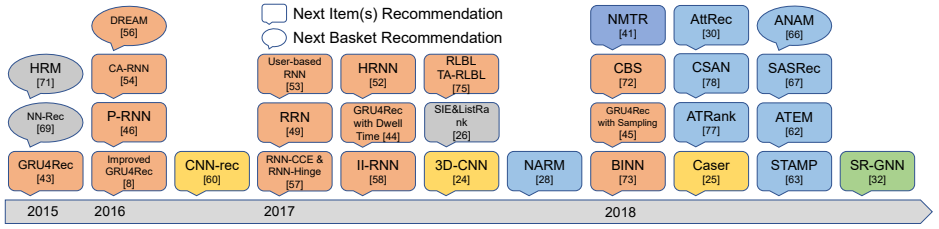


Fig. 8. Some recent and representative DL-based sequential recommendation models. Different colors indicate different DL techniques (grey: MLP; orange: RNN; yellow: CNN; blue: attention mechanism; green: GNN).

2.3.3 Concluding Remarks. Compared with conventional methods, DL-based methods are a much more active research area in the recent years. The MC- and MF-based models assume that a user's next behavior is related to only a few recent behavior(s), while DL methods utilize a much longer sequence for prediction [86], as they are able to effectively learn the *theme* of the whole sequence. Thus, they generally obtain better performance (in terms of accuracy measurement) than traditional models. Meanwhile, DL methods are more robust to sparse data and can adapt to varied length of the input sequence. The representative DL-based sequential recommendation algorithms are presented in Figure 8, which will be introduced in details in next sections.

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

3 SEQUENTIAL RECOMMENDATION ALGORITHMS

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

3.1 Experience-based Sequential Recommendation

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

3.1.1 Conventional models for multi-behavior recommendation. Ajit el al. [67] first proposed a collective matrix factorization model (CMF) to simultaneously factorize multiple user-item interaction matrices (in terms of different behavior types) by sharing the item-side latent matrix (item embedding) across matrices. Other studies [36, 102] extended CMF to handle different user behaviors (e.g., social relationships). Besides, there are also some models addressing multi-behavior recommendation with Bayesian learning. For example, Loni et al. [47] proposed multi-channel BRP to adapt the sampling rule for different behavior types. Qiu et al. [54] further proposed an adaptive sampling method for BPR by considering the co-occurrence of multiple behavior types. Guo et al. [22] 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. [17] developed a margin-based learning framework to model the pairwise ranking relations among purchase, view, and non-view behaviors.

3.1.2 DL-based multi-behavior recommendation. DL techniques have also been applied in multi-behavior recommendation. For example, *NMTR* [20] is proposed to tackle some representative problems of conventional models for multi-behavior recommendation, e.g., lack of behavior semantics, unreasonable embedding learning and incapability in modeling complicated interactions. To capture the sequential relationships between behavior types, NMTR [20] cascades predictions of different behavior types by considering the sequential dependency relationship among different behaviors in practice[6], which thus translates the heterogeneous behavior problem into the experience-based sequential recommendation problem as we have defined.

It should be noted that this cascaded prediction, which could be regarded as pre-training embedding layers of other behavior types before learning a recommendation model for the target behavior, only considers the connections between target behavior and previous behaviors but ignores the ones between target behavior and subsequent behaviors. 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 [20] is generally better than those using the sequential training. Besides, Xia et al. [93] proposed a multi-task model with LSTM to explicitly model users' purchase decision process by predicting the stage and decision of a user at a specific time with the assistance of a pre-defined set of heuristic rules, and thus obtaining more accurate recommendation results.

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

3.2 Transaction-based Sequential Recommendation

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

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

(1) GRU4Rec-related models. Hidasi et al. [27] 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, 26, 76] have been proposed, which also use RNN architectures for modeling behavior sequence. The architecture of GRU4Rec is shown in Figure 9. As introduced in [27], the input of GRU4Rec is a session (behavior sequence), which could be a single item, or a set of items appeared in the session. It uses one-hot encoding to represent the current item, or a weighted sum of encoding to represent the set of items. The core of the model is the GRU layer(s), where the output of each layer is the input for the next layer, but each layer can also be connected to a deeper non-adjacent GRU layer in the network. Feedforward layers are added between the last GRU layer and the output layer. The output is the probability of each candidate item that will appear in the next behavior.

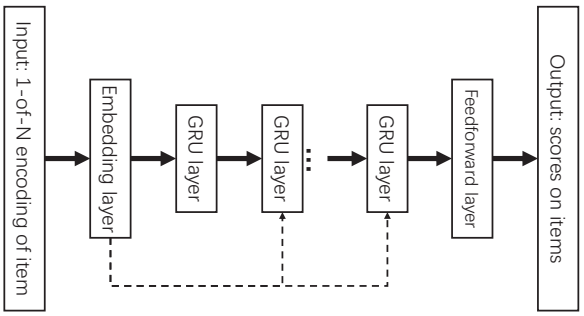


Fig. 9. Architecture of GRU4Rec.

GRU4Rec employs *session-parallel mini-batches* and *popularity-based negative sampling* for training. The reason for using session-parallel mini-batches is to form sessions with equal length while the length of actual sessions can be greatly varied. On the other hand, if simply breaking a session into different parts to force them into equal length, we could not well model the behavior sequence and fail to capture how a session evolves over time [27].

The improved studies strive to improve the model performance from the perspectives of model training and designing more advanced model structures for better learning item information. For example, for **facilitating training**, [76] applied *data augmentation* to enhance training of GRU4Rec. [8] considered the *dwell time* to modify the generation of mini-batch, which has been verified to greatly improve performance. On the other hand, popularity-based sampling suffers from the problem that model learning slows down after all the candidates items have been ranked above popular ones, which could be a relatively serious problem for long-tail items recommendation. Thus, [26] proposed the *additional sampling* (a combination of uniform sampling and popularity

sampling) for negative sampling in GRU4RC, which can enormously improve recommendation accuracy.

For **better modeling item information**, [28] considered additional item information other than IDs (e.g., text descriptions and images) for improving prediction performance. Specifically, they introduced a number of parallel RNN (p-RNN) architectures to model sessions based on click behaviors and other features of the clicked items (e.g., pictures and text descriptions). Moreover, they particularly proposed alternative but more suitable training strategies for p-RNNs: *simultaneous*, *alternating*, *residual*, and *interleaving* training. In simultaneous training (baseline), every parameter of each subnet is trained simultaneously. In alternating training, subnets are trained in an alternating fashion per epoch. In residual training, subnets are trained one by one by the residual error of the ensemble of the previously trained subnets, while interleaving training is alternating training per mini-batch. Furthermore, [31] combined the session-based KNNs with GRU4Rec using the methods of *switching*, *cascading*, and *weighted hybrid*.

(2) With user representation. There are also some studies that aim to better model users' preference. For example, [100] proposed a RNN-based framework for click-through rate (CTR) prediction in sponsor search, which considers the impact of the click dwell time with the assumption that the longer a user stays on an ad page, the more attractive the ad is for the user. In total, three categories of features are considered: ad features (ad ID, position and query text), user features (user ID, user's query) and sequential features (time interval, dwell time and click sequence). [70] took one-hot encoding of items in users' behavior sequences as input of a GRU-based RNN to learn users' historical embeddings. *RRN* [91] is the first recurrent recommender network that attempts to capture the dynamics of both user and item representation. [7] further improved the RRN's interpretability by devising a time-varying neighborhood style explanation scheme, which jointly optimizes prediction accuracy and interpretability of the sequential recommendation.

Considering that simply embedding a user's historical information into a single vector may lose the per-item or feature-level correlation information between a user's historical sequences and long-term preference, Chen et al. [10] thus proposed a memory-augmented neural network for the sequential recommendation. The model explicitly stores and updates every user's historical information by leveraging an external memory matrix.

HRNN [56][7] uses GRU to model users and sessions respectively. The session-level GRU considers a user's activities within a session and thus generates recommendations, while the user-level GRU models the evolution of a user's preference across sessions. Given that the length of sessions of different users are varied, it deploys user parallel mini-batch training, which is extended from session parallel mini-batch of GRU4Rec. Donkers et al. [18] further proposed a user-based GRU framework (including linear user-based GRU, rectified linear user-based GRU, and attentional user-based GRU) to integrate user information for giving better user representations.

(3) Context-aware sequential recommendation. Most of the previous models have ignored the huge amount of context information in real-word scenarios. In this case, [44] summarized two types of contexts: *input contexts* and *transition contexts*. Input contexts refer to the ones by which users conduct their behaviors, e.g., location, time and weather, whilst transition contexts mean the transitions between two adjacent input elements in historical sequences (e.g., time intervals between the adjacent behaviors). It further designed the context-aware recurrent neural networks (*CA-RNN*) to simultaneously model the sequential and contextual information. Besides, [71] proposed *ARNN* to consider the user-side contexts, e.g., age, gender and location. Specifically, *ARNN* extracts high-order user-contextual preferences using a product-based neural network, which is capable of being incorporated with any existing RNN-based sequential recommendation models.

[7] github.com/mquad/hgru4rec.

(4) Other models. Other than the aforementioned three categories, there are other RNN-based models (e.g., *DREAM* [94]) for transaction-based sequential recommendation in the literature. For example, [16] used RNN for the collaborative filtering task and considered two different objective functions in the RNN model: categorical cross-entropy (*CCE*) and *Hinge*, where CCE has been widely used in language modeling, and Hinge is extended from the objective function of SVMs. [62] deployed a multi-layer GRU network to capture sequential dependencies and user interest from both the inter-session and intra-session levels. In view of that existing studies assume that there is only an implicit purpose for users in a session, Wang et al. [88] proposed a mixture-channel purpose routing networks (MCPRNs) to capture the possible multi-purposes of users in a session (a channel implies a latent purpose). MCPRNS consists of a purpose router (PRN) and a multi-channel recurrent framework with purpose-specific recurrent units.

3.2.2 CNN-based Models. RNN models are limited to model relatively short sequences due to their network structures and relatively expensive computing costs, which can be partially alleviated by CNN models [65]. For example, *3D-CNN* [78] designs an embedding matrix to concatenate the embedding of item ID, name, and category. *Caser* [77] 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* [29] is highly similar to Caser in terms of user embedding and horizontal convolution, but it does not deploy vertical convolution. *NextItNet* [95] is a generative CNN model with the residual block structure for the sequential recommendation. It is capable of capturing both long and short-term item dependencies.

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

(1) Vanilla attention mechanisms. *NARM*[8] [39] is an encoder-decoder framework for transaction-based sequential recommendation. In the local encoder, RNN is combined with vanilla attention to capture the major purposes (or interest) of a user in the current sequence. With the attention mechanism, NARM is able to eliminate noises from unintended behaviors, such as accidental (unintended) clicks. [87] applied the vanilla attention mechanism to weight each item in a sequence to reduce the negative impact of unintended interactions. Liu et al. [46] proposed a short-term attention/memory priority model, which uses vanilla attention to calculate attention scores of items in a sequence as well as the attention correlations between previous items and the most recent item in the sequence. Ren et al.[58] considered repeat consumption issue, and thus proposed *RepeatNet* which evaluates the recommendations from both the repeat mode and the explore mode, which refer to the old item from a user's history and the new item, respectively. [63] incorporated vanilla attention with a Bi-GRU network to model user's short-term interest for music recommendation. [2] proposed a unified attribute-aware neural attentive model, which applies vanilla attention mechanism on feature level.

(2) Self-attention mechanisms. Self-attention mechanisms have also obtained increasing interest in the sequential recommendation. For example, Zhang et al. [98] utilized the self-attention mechanism to infer the item-item relationship from the user's historical interactions. With self-attention, it is capable of estimating weights of each item in the user's interaction trajectories to learn more accurate representations of the user's short-term intention, while it uses a metric

[8] github.com/lijingsdu/sessionRec_NARM.

learning framework to learn the user's long-term interest. Similarly, SASRec [34] adopts a self-attention layer to balance short-term intent and long-term preference, and seeks to identify items relevant to the next behavior from the user's historical behavior sequences. BERT4Rec [19] is the improved version of SASRec, which introduces the transformer architecture for the sequential recommendation and trains the bidirectional model to model sequential data using Cloze task. Besides, to overcome the drawbacks of RNN-based sequential recommender systems, such as not supporting parallelism and only modeling one-way transitions between consecutive items, SANSR [74] incorporates the transformer framework [81] to speed up the training process and learn the relations between items in the session regardless the distance and the direction.

As we know, most of the previous studies on the sequential recommendation focus on recommendation accuracy, but ignore the *diversity* of recommendation results, which is also a quite important measurement for effective recommendation. With respect to this issue, Chen et al. [9] proposed an intent-aware sequential recommendation algorithm, which uses the self-attention mechanism to model a user's multi-intents in a given session.

3.2.4 Other Models. There are also some other DL-structures (e.g., MLP [85], GNN and autoencoder) that have been adopted in the sequential recommendation. For example, *NN-rec* [82] is the first work considering neural network for next-basket recommendation, which is inspired by the NLP model (NLPM) [6]. Wu et al. [92] first used GNN for session-based recommendation to capture complex transitions among items. In this model, each session are modeled as a directed graph, and are proceeded by a gated graph neural network to obtain session representations (local session embedding and global session embedding). *GACOforRec* [96] utilizes graph convolutional neural networks to learn the item order within a session as well as the spatiality within the network to handle a users' short-term intents, while it designs ConvLSTM to capture the user's long-term preference. Besides, considering that different behaviors may have different impacts, it proposes a new pair of attention mechanisms which consider the different propagation distances in the graph convolutional network to obtain the different weights. Sachdeva et al. [64] explored the variational autoencoder for modeling a user's preference through her historical 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 behavior sequence consists of both different behavior types and different behavior objects. Thus, the recommendation models are expected to capture both the sequential dependencies among different behaviors, different items as well as behaviors and items, respectively. Next, we summarize the related models according to the deployed DL techniques.

3.3.1 RNN-based Models. RNN-based models still take the majority role in this task [48]. For example, [79] proposed a RNN-based model without explicitly learning user representation. Given the task of predicting the next item expected to appear in terms of a target behavior type, Le et al. [37] firstly divided a session into a target sequence and a supporting sequence according to the target behavior type. Its basis idea is that the target behavior type (e.g., purchase) contains the most efficient information for the prediction task, and the remaining behaviors (e.g., click) can thus be utilized as the supporting sequences that can facilitate the next-item prediction task for the target behavior type. Besides, in order to better model the dependencies among different behavior types, some studies would also assume that there is a cascading relationship (as in Section 3.1.2) among different types of behaviors (i.e., different behavior types are sequentially ordered). For example, Li et al. [41] proposed a model that consists of two main components: *neural item embedding* [4] and *discriminative behavior learning*. For behavior learning, it utilizes all types of behavior (e.g.,

click, purchase and collect) to capture a user's present consumption motivation. Meanwhile, it selects purchase related behaviors (e.g, purchase, collect and add-to-cart) from user's historical experience to model the user's underlying long-term preference. Considering that RNN cannot well handle users' short-term intent in a sequence whereas log-bilinear model (LBL) cannot capture users' long-term preference, Liu et al. [45] combined RNN with LBL to construct two models (*RLBL* and *TA-RLBL*) for modeling multi-behavioral sequences. TA-RLBL is an extension of RLBL [45] which considers the continuous time difference information between input behavior objects and thus further improve the peroformance of RLBL. [69] took context information (e.g., behavior type) into consideration by modifying the structures of RNN.

3.3.2 Other Models. There are also some other DL techniques applied in the interaction-based sequential recommendation, including attention mechanisms, MLPs and Graph-based models. For example, [103] proposed *ATRank*[9] which adopts both *self-attention* and *vanilla attention* mechanisms. Considering the heterogeneity of behaviors, ATRank models the influence among behaviors via self-attention, while it uses vanilla attention to model the impact of different behaviors on the recommendation task. *CSAN* [30] is the improved version of ATRank by also considering side information and polysemy of behavior types. Wu et al. [90] proposed a deep listNet ranking framework (MLP-based) to jointly consider user's clicks and views. Ma et al. [50] proposed a graph-based broad-aware network (*G-BBAN*) for news recommendation, which considers multiple user behaviors, behavioral sequence representations, and user representation.

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

Task	DL Model	Papers
Experienced-based	MLP	[20]
	RNN	[93]
Transaction-based	RNN	[8–10, 18, 27, 44, 56, 71, 74, 76, 94] [7, 16, 26, 28, 31, 62, 70, 91, 100]
	CNN	[29, 77, 78, 95]
	MLP	[82, 85]
	Attention mechanism	[2, 9, 19, 34, 39, 46, 58, 63, 87, 98]
	GNN	[92, 96]
Interaction-based	RNN	[37, 41, 45, 48, 69, 79]
	MLP	[90]
	Attention mechanism	[30, 48, 103]
	GNN	[50]

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

[9] github.com/jinze1994/ATRank.

recommendation: (1) the behavior type and the item in a behavior 2-tuple (c_i, o_i) are mostly equally treated. For example, ATRank [103] and CSAN [30] adopt the same attention score for the item and the corresponding behavior type; (2) different behavior types are not distinguished successfully. For example, [79] 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, [90] used pooling operation to model multi-type behavior in a sequence. In view of these issues, more advanced and effective approaches are needed for much more effective sequential recommendation, especially for the task of interaction-based sequential recommendation. In the next two sections, we will further summarize and evaluate the factors that might impact the performance of a DL-based model, which is expected to better guide future research.

4 INFLUENTIAL FACTORS ON DL-BASED MODELS

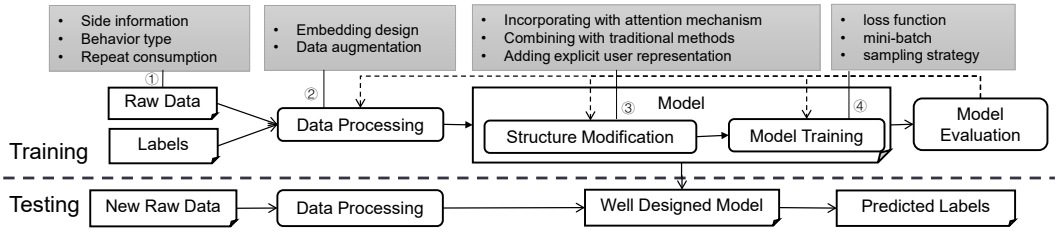


Fig. 10. Influential factors of DL-based models.

Figure 10 shows the *training* and *testing* process of a sequential recommender system. In the training, the input includes raw data and label information, which are then fed into the data processing module, mainly including *feature extraction* and *data augmentation*. Feature extraction refers to converting raw data into structured data, while data augmentation is normally used to deal with data sparsity and cold-start problems, especially in DL-based models. 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 and Table 2) 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.

4.1.1 Side Information. Side information has been well recognized to be effective in facilitating recommendation performance [75]. It refers to information about items (other than IDs), e.g., *images*, *text descriptions*, and reviews, or information related to transactions (behaviors) like *dwell time*. Text and image information about items have been widely explored in DL-based collaborative filtering systems [3, 12, 52, 57, 97], as well as in some DL-based sequential recommender systems [28, 30]. For example, p-RNN [28] uses a parallel RNNs framework to process the item IDs, images and texts. Specifically, the first parallel architecture trains a GRU network (i.e., subnet) for item

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

Module	Factor	Method	Papers
Input	side information	utilize image/text	[28, 30]
		utilize dwell time	[8, 14, 100]
	behavior type	simple behavior embedding	[103]
		divide session into groups for different purposes	[37, 41]
	repeat consumption	consider repeat behavior	[58]
Data processing	embedding design	item embedding	[21]
		w-item2vec	[41]
		session embedding	[90]
	data augmentation		[76, 78]
Model structure	incorporating attention mechanism	only attention mechanism	[2, 30, 98, 103]
		incorporating vanilla attention mechanism with other DL methods	[34, 39, 46, 87]
	combining with conventional methods	KNN	[31]
		metric learning	[98]
	adding explicit user representation	user embedded models	[77, 85]
		user recurrent models	[7, 10, 18, 41, 56, 91]
Model training	negative sampling	uniform	[26, 27]
		popularity-based	[26, 27]
		additional	[26]
		sample size	[26]
	mini-batch creation	session parallel	[27]
		item boosting	[8]
		user parallel	[56]
	loss function	TOP1	[27]
		TOP1-max & BPR-max	[26]
		CCE &Hinge	[16]

representation on the basis of each kind of information, respectively. The model concatenates the hidden layers of the subnets and generates the output. The second architecture has a shared hidden state to output weight matrix. The weighted sum of the hidden states is used to produce the output instead of being computed by separate subnets. In the third structure called parallel interaction, the hidden state of the item feature subnet is multiplied by the hidden state of the ID subnet in an element-wise manner before generating the final outcome. CSAN [30] 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 [28, 40, 83] and cold-start [30, 84, 89, 103] problems.

On the other hand, side information like *dwell time* partially imply a user's degrees of interest on different items. For example, when a user browses a web page for an item, the longer she stays, we can infer that the more she is interested in. Bogina et al. [8] applied item boosting according to the dwell time for generating mini-batch in training. In particular, assuming that a predefined threshold of dwell time is t_d seconds, if the dwell time on an object i in a session is within the range of $[2t_d, 3t_d)$, then the parallel mini-batch of this session will contain 2 repeated behaviors regarding to i, i.e., the presence of i in the session increases. This strategy (referred as *item boosting*) can be considered as to re-measure the importance of behavior objects in terms of the corresponding dwell time. Zhang et al. [100] treated dwell time as a sequence feature, and concatenated it with other features (e.g., query text). Similarly, Dallmann et al. [14] proposed an extension to existing RNN approaches by adding user dwell time. Experiments in [8] show that incorporating dwell time with GRU4Rec [27] 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 [30, 103], and different behavior types imply users' different intents. For instance, a purchase action is a better indicator of a user's preference on an item than a click behavior. Therefore, it is critical to treat different behavior types differently [20, 37, 79, 103]. For example, CBS [37] divides a sequence into target sequence and supporting one in terms of behavior types, where the target sequence is related to the behavior type (e.g., purchase) that has the most efficient information for prediction. Similarly, BINN [20] utilizes all behavior types (e.g., click, purchase and collect) to capture a user's present interest whereas models the user's long-term preference using only purchase related information (e.g, purchase,add-to-cart and collect). [103] learns the representation of each behavior type and then concatenate them with the corresponding item embedding vectors. Experiments generally support that purchase behavior can more accurately capture a user's long-term preferences, whilst other behavior types can facilitate the learning of short-term interests [20, 37, 79, 103].

4.1.3 Repeat Consumption. Repeat consumption refers to that an item is repeatedly appeared in a user's historical sequences, which is mostly ignored in the sequential recommendation. Only *RepeatNet* proposed by Ren et al.[58][10] has ever considered this issue, and their results confirm that the consideration of repeat consumption patterns in DL 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 the sequential recommendation, embedding methods are used to represent information about an item, a user, or a session. For example, Greenstein et al. [21] adopts the word embedding methods GloVe [53] and Word2Vec [51] (CBOW) for *item embedding* in e-commerce applications. Li et al. [41] further proposed w-item2vec (inspired by item2vec [4]) on the basis of the Skip-gram model and thus formed unified representations of items. Wu et al. [90] designed a session embedding for pre-training by considering different user search behaviors

[10] github.com/PengjieRen/RepeatNet.

such as clicks and views, the target item embedding and the user embedding together to have a comprehensive session understanding (i.e., session representation).

4.2.2 Data Augmentation. In the sequential recommendation, in some scenarios there might be no user profiles or historical information for a new user, or a user who does not log in, i.e., cold-start problems. In this case, data augmentation becomes an important technique. For example, Tan et al. [76] proposed an augmentation method, where prefixes of the original input sessions are treated as new training sequences as shown in Figure 11. That is, given the original session (l_1, l_2, l_3, l_4), we can generate 3 training sequences: (l_1, l_2), (l_1, l_2, l_3), (l_1, l_2, l_3, l_4), while a recommendation algorithm predicts the last item for each training sequence. With this method, a session is repeatedly utilized during training, which is demonstrated to improve 14.7% over GRU4Rec on MRR@20 [76]. Besides, the *dropout* method [73] is further adopted to prevent the over-fitting problem (see Figure 11, a circle with the dotted line is the dropout behavior in each sequence). Besides, considering that items after a target item may also contain valuable information, these items are thus viewed as privileged information as [80] to facilitate the learning process.

Similarly, with regard to two behavior types (i.e., add-to-cart and click), 3D-CNN [78] 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.

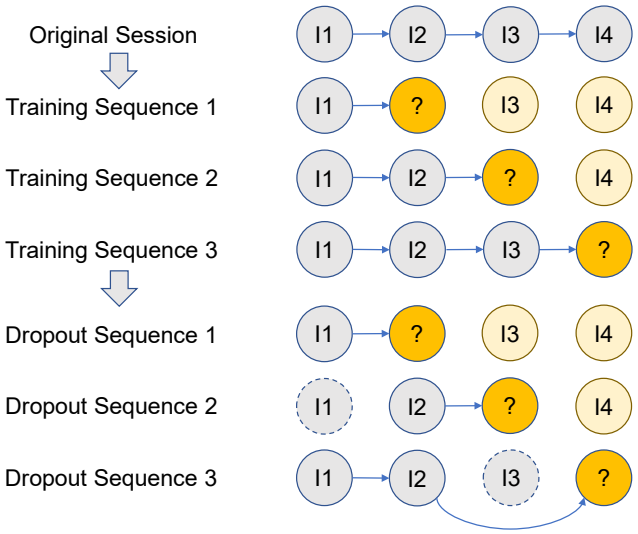


Fig. 11. Data argumentation. The orange circles represent the predicted items; the dotted circles represent the item that is deleted in the dropout method, and light orange circles make up privileged information.

4.3 Model Structure

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

4.3.1 Incorporating Attention Mechanisms. In Section 2.3.2, we discuss that there are mainly *vanilla attention* and *self-attention*. Overall, we can incorporate attention mechanism with other DL models,

or just build attention models to address the sequential recommendation problems. For the first scenario, NARM [39], ATEM [87] and STAMP [46] incorporate the *vanilla attention mechanism* with RNN or MLP, aiming to capture user's main purpose in a given session. Experiments verify that their performance surpassed GRU4Rec by 25%, 92% and 30% respectively. SASRec [34] 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 [77], respectively. For the second scenario, for example, AttRec [98] simply deploys a self-attention mechanism to capture users' short-term interest, where its performance exceeds Caser by 8.5% on HR@50. ATRank [103] and CSAN [30] combine self-attention with vanilla attention for the sequential recommendation. Attention mechanisms can be further employed to capture attribute-level importance level of items for modeling users' interest. For example, ANAM [2] 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 the DL-based models, while mostly only using self-attention mechanisms can have better performance than some DL-based models without attention mechanisms.

4.3.2 Combining with Conventional Methods. DL-based models can also be combined with traditional methods to bootstrap their performance on the sequential recommendation tasks. For example, [31] combined a session-based KNN with GRU4Rec [27] in three different ways (i.e., switching, cascading, and weighted hybrid), showing that the best combination can exceed original GRU4Rec by 9.8% in some applications. AttRec [98] combines self-attention (for short-term interest learning) and metric learning (for long-term preference modeling), and the performance exceeds Caser[77] by 8.5% on HR@50.

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

User embedded models. This fold of models explicitly learns user representations [77, 85] via embedding methods, but not in a recurrent process as item representation. They can facilitate the performance of the sequential recommendation models [77]. However, such models might suffer from the cold-start user problem since the long-term interest of a user with little historical information cannot be well learned. Another issue is that, user representation via user embedded models is learned in a relatively static way, which cannot capture users evolved and dynamic preferences. In this view, user recurrent models are more effective, which also learn user representations in a recurrent way as item representation learning.

User recurrent models. They treat both user and item representations as recurrent components in the DL-based models, which can better capture users' evolving preferences, including memory-augmented neural network [10], RNN-based models [18, 41, 56] and recurrent neural networks [7, 91]. For example, [18, 41, 56] used RNN framework to learn users' long-term interest from their historical behavior sequences. Experiments verify that considering a user's long-term interest is critically valuable for personalized recommendation, e.g., HRNN [56] exceeds GRU4Rec by 3.5% with explicit user representation in some scenarios.

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

4.4 Model Training

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

4.4.1 Negative Sampling. *Popularity-based sampling* and *uniform sampling* have been widely used in recommendation. Popularity-based sampling assumes that the more popular an item is, the more possibly that a user knows about it. In this case, if a user does not interact with it previously, it is more likely that the user dislikes it. [26] further proposed a novel sampling strategy (called *additional sampling*) by combining these two sampling strategies, which takes the advantages but overcomes the shortcomings of both strategies in negative sampling. In the additional sampling strategy, negative samples are selected with a probability proportional to supp_i^{α}, where supp_i is the support of item i and α is a parameter ($0 \leq \alpha \leq 1$). The cases of $\alpha = 0$ and $\alpha = 1$ are equivalent to uniform and popularity-based sampling respectively. Experiment results show that additional sampling can surpass both the popularity-based sampling and uniform sampling methods under certain scenarios (e.g., loss functions). Besides, *the size of negative samples* can also affect the performance of sequential recommendation models.

4.4.2 Mini-batch Creation. *Session parallel mini-batch training* was proposed in [27] to accommodate sessions of varying lengths and strive to capture the dynamics of sessions over time. In particular, sessions are firstly arranged in time order. Then, the first event (behavior) of the first X sessions (X is the number of sessions) is used to form the input of the first mini-batch (whose desired output is the second event of the active sessions). The second mini-batch is formed from the second event of the X sessions, and so on and so forth. If any of the X sessions reaches its ending, the next available session out of the X sessions is placed in the corresponding place to continually form the mini-batch. Session parallel mini-batch has two variants: the first one is *item boosting* [8]. 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 [56] designs user-parallel mini-batch (i.e., parallel sessions belong to different users) to model the evolution of users' preferences across sessions.

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

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

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

where $\sigma(.)$ is a sigmoid function, r_i and r_j are the ranking scores for sample i and j respectively. Following the same notations, **BPR** (Bayesian Personalized Ranking) [59] is defined as:

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

TOP1 and BPR loss functions might suffer from the gradients vanishing problems for DL-based models (e.g., in GRU4Rec [26]). In this view, ranking-max loss function family is proposed [26] to address this issue, where the ranking score is only compared to the negative sample which is most relevant to the target sample, i.e., the one has the highest ranking score. Accordingly, we have **TOP1-max** and **BPR-max**, which are formulated as Equations 4 and 5 respectively. In this case, TOP1-max and BRP-max can be considered as the weighted version of TOP1 and BPR. Previous research validates that the two loss functions largely improve the performance for RNN-based sequential recommendation models [26].

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

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

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

In addition to the ranking-based loss functions, *CCE* and *Hinge* loss functions have also been applied in the sequential recommendation [16]. **CCE** is defined as:

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

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

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

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

5 EMPIRICAL STUDIES ON INFLUENTIAL FACTORS

In this section, we conduct experiments[11] 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[12], which contains click and buy behaviors from an online shop. Only the click data is used in our evaluations. *RSC19* is published by RecSys Challenge 2019[13], which contains hotel search sessions from a global hotel platform. *RSC19 (user)* is a subset of *RSC19*, where we select users with more than 10 sessions, and consider the last session of each user as the 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

[11] The source codes and datasets of the experiments will be publicized on GitHub.
[12] www.kaggle.com/chadgostopp/recsys-challenge-2015.
[13] www.recsyschallenge.com/2019/.

manually divide the behavior sequence of each user into sessions every 30 minutes. The statistic information of these datasets are summairzed in Table 3.

Table 3. The statistic information of the four datasets.

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

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

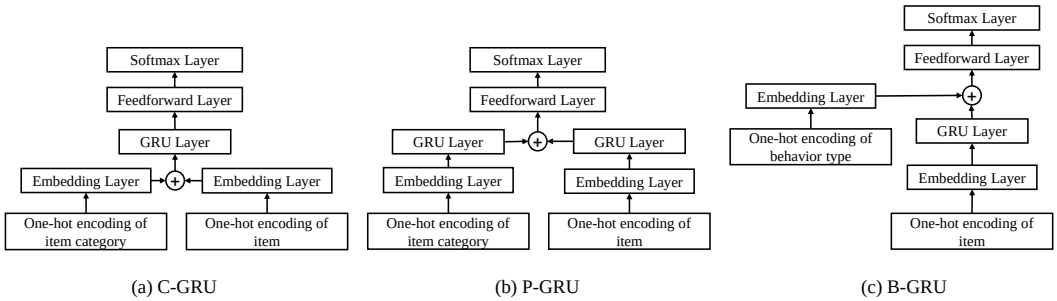
Table 4. Other parameters settings for different scenarios.

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
Model	**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
Model	**LastFM**		
	Batch Size	Lr	RNN Size
Default	50	0.001	50
User Recurrent	200	0.02	50

5.1.2 Model settings. We choose GRU4Rec [27] (Figure 9) as our *basic* model, and then consider the influential factors in Figure 10 to check their effects on the basic model. The main reason of using GRU4Rec is that lots of algorithms in the literature make improvement on it, or recognize it as a representative and competitive baseline for the sequential recommendation tasks. The *default* parameters for the **basic** model is no data augmentation, no user representation, BRP-max loss function, uniform 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 being particularly figured out, other models also use these default settings.

For the input module, we choose two kinds of **side information**: *item category* and *dwell time*. For the item category, 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 [8], 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 vectors and behavior type one-hot vectors as input and converts them into embedding vectors, where item embedding vectors are fed into a GRU model, whose output is concatenated with behavior type embedding vectors for MLP layers. Besides, B-GRU uses uniform negative sampling method with a sample size of 32. The structures of **C-GRU**, **P-GRU** and **B-GRU** are shown in Figure 12.



(a) C-GRU (b) P-GRU (c) B-GRU

Fig. 12. The network structures of C-GRU, P-GRU and B-GRU.

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

For the model structure module, we consider three structures: **NARM** [39] (*incorporating the basic model with the attention mechanism*), **weighted model** in [31] (*combining the DL model with KNN*) and **adding an explicit user representation** in two ways (i.e., the model in [56] and the second one adding a user embedding part based on user IDs, which is concatenated with the output of GRU in the basic model). Besides, for the last one, we have a small dataset from RSC19 (**RSC19 (user)** in Table 3) which selects users who have substantial amount of historical data for user 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 [26]), 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 4, where **Lr** refers to learning rate of these DL-based 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 [32]) 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 [98] 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 [77]. 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):

$$\begin{aligned} \text{AP@}k(m) &= \frac{1}{m_g}\sum_{i=1}^{k}\frac{P(i)I_m(i)}{i}, \\ \text{MAP@}k &= \frac{1}{|M|}\sum_{m\in M}\text{AP@}k(m) \end{aligned} \tag{8}$$

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.

$$\text{NDCG@}k = \frac{\text{DCG@}k}{\text{IDCG@}k}, \text{DCG@}k = \sum_{g\in G}\frac{s_g I_k(g)}{\log_2(i_g+1)} \tag{9}$$

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* refers to 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 appears 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 specific requirements.

5.2 Experiment Results

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

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

Side information effects. Tables 5 and 6 show the results of the two types of the side information on DL-based model respectively. As shown in Table 5, 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 6, **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

Table 5. Results of incorporating item category or behavior type.

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

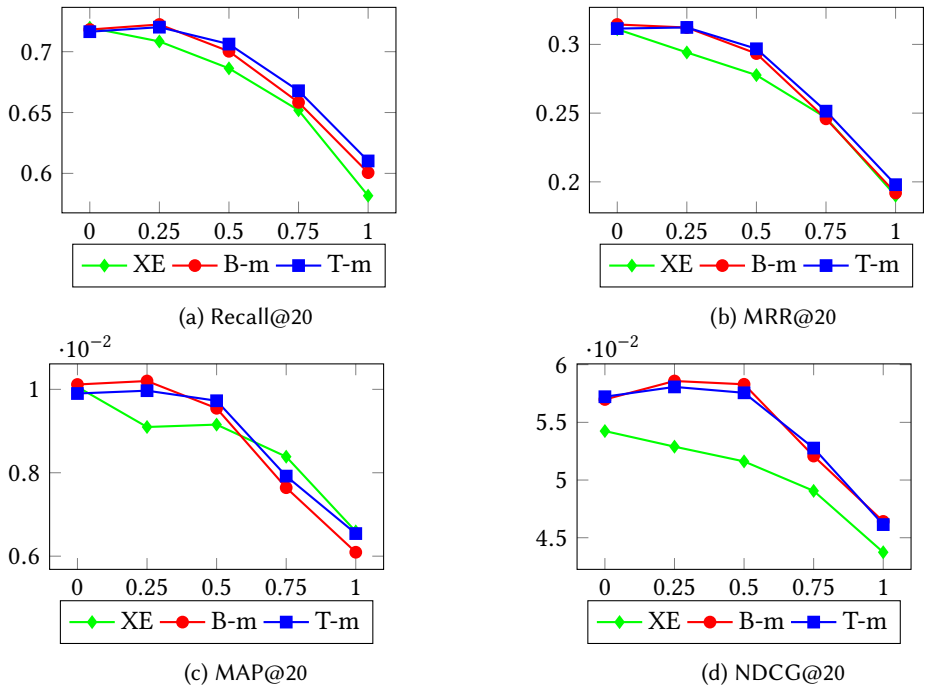


Fig. 13. The impact of α on additional sampling strategy on RSC15 (XE: cross-entropy; B-m: BPR-max; T-m: TOP1-max).

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 present in Table 5. 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 the majority part (62%). In

Table 6. Results of different factors.

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

this case, without a careful design, considering other supporting behaviors for the target behavior might probably incorporate noisy information for the prediction task.

5.2.2 Data Processing. Here, we check the experimental results in regard to the data processing part.

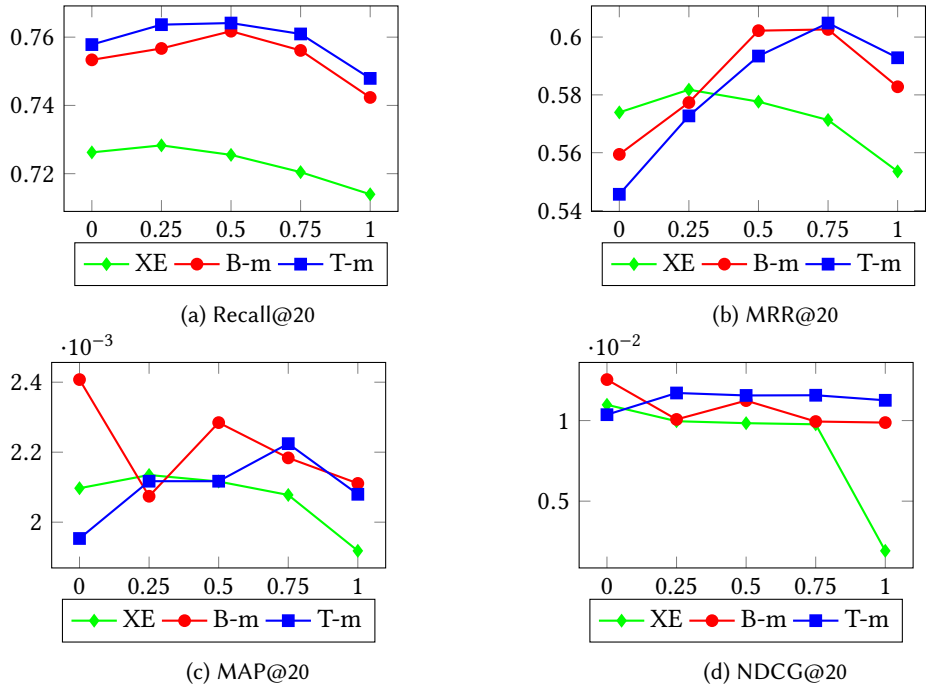


Fig. 14. The impact of α on additional sampling strategy on RSC19 (XE: cross-entropy; B-m: BPR-max; T-m: TOP1-max).

Data augmentation effects. As shown in Table 6, 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.

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

Incorporating attention mechanism effects.

As shown in Table 6, incorporating **attention mechanism** enhances the performance of the model almost for all the scenarios (except NDCG).

Combining with conventional method effects. 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 that the way of combining traditional models with DL-models can have a significant effect on the sequential recommendation.

User representation effects. 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 3, the number of sessions is much greater than that of users on these two datasets; 3) according to the number of items and behaviors shown in Table 3, the average support of items is much smaller on these two datasets. On the other hand, in terms of NDCG@20 and MAP@20, user representation

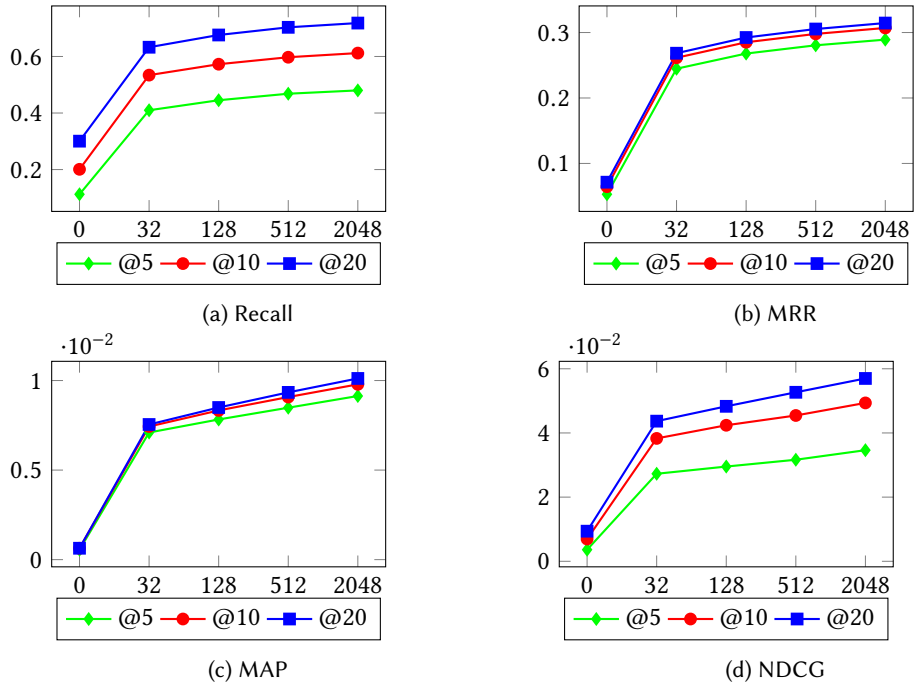


Fig. 15. The effect of sample size on RSC15.

models greatly outperform the basic model, while the user recurrent model performs better than the user embedded model. In this case, in the 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.

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

Sampling method effects. Figures 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 steadily declines with the increase of 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 computational costs. Therefore, in real world

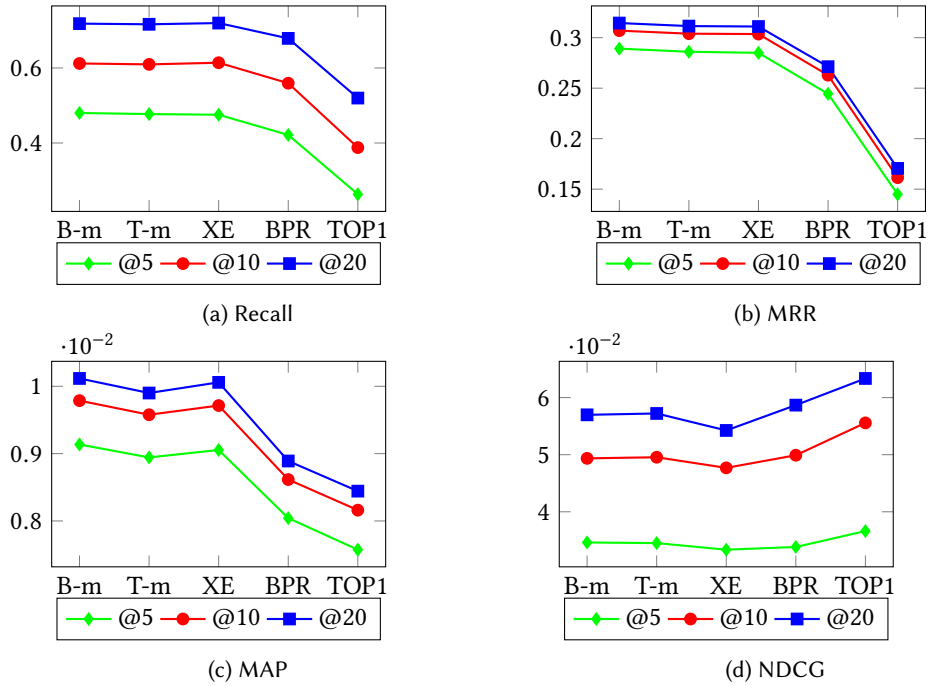


Fig. 16. Model performance for different loss functions on RSC15 (B-m: BPR-max; T-m: Top1-max; XE: cross-entropy).

applications, we need to keep a balance between model performance and training time considering the size of negative samples.

Loss function effects. As depicted in Figures 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.

5.2.5 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) when allowed, and carefully design the corresponding modules; 2) well consider the connections between other behavior types with the target behavior, and be careful about the possible noisy information involved in final recommendation when model these connections; 3) always incorporate data argumentation, TOP1-max, BPR-max and cross-entropy loss functions for training, keep a balance between model performance and computational cost with regard to the size of negative sample; and 4) for any DL-based models, consider to further improve their performance with attention mechanism, by possibly combing with the traditional sequential learning models, and a well designed explicit user representation module.

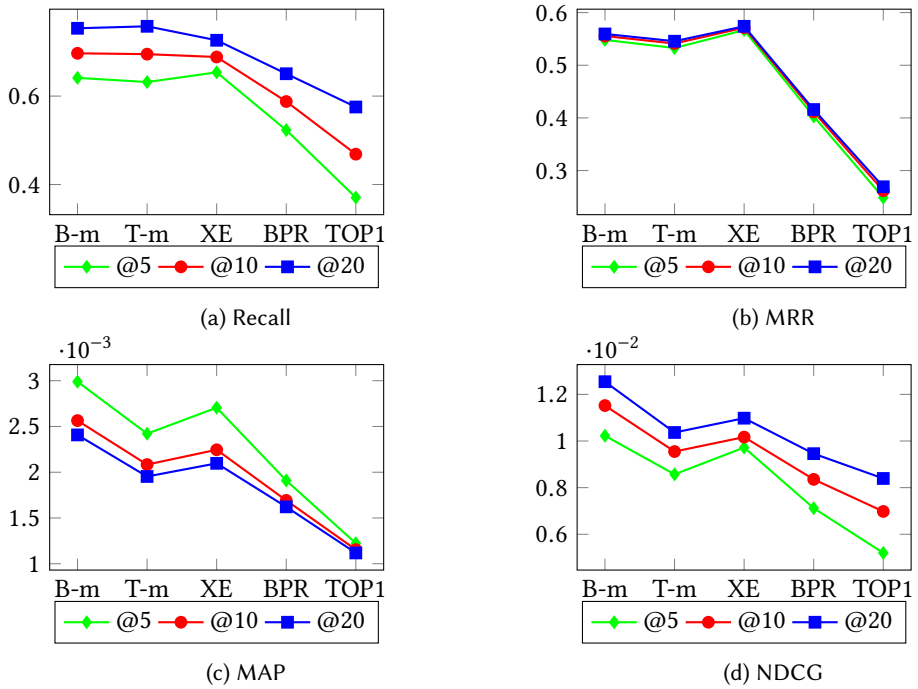


Fig. 17. Model performance for different loss functions on RSC19 (B-m: BPR-max; T-m: Top1-max; XE: cross-entropy).

6 FUTURE DIRECTIONS AND CONCLUSIONS

6.1 Future Directions

As we have discussed, DL techniques have greatly promoted the sequential recommendation studies, but also are accompanied by some challenging issues. Thus, we summarize the following open issues which can be 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 [27] 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 [95] (CNN-based model) or further refer to the comparison framework proposed in [49]. There is a growing consensus that only complex deep learning structures could not always guarantee better and more importantly robust recommender systems [13].

More designs on embedding methods. Most previous studies adopt the embedding methods from NLP. However, in the sequential recommendation, it is rather challenging to pre-train an embedding model (e.g., word2vec) as the information is constantly changing while the words and their connections in NLP are relatively fixed and static. On the other hand, behaviors in the sequential recommendation are also very complex than words as they involve both behavior objects and types. Furthermore, the incorporation of embedding vectors in existing sequential recommendation models are also in a relatively simple way. In this case, more advanced and particular designs of

embedding methods are needed for the sequential recommendation. For example, [43] designed a sequential embedding approach which also takes the dependencies relationships among items and their attributes into consideration for the sequential recommendation.

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

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 the sequential recommendation for interaction-based sequential recommendation tasks. Our empirical evaluation also indicates that well considering another behavior type for a target type is very challenging. In this case, more DL-models can be designed by considering the connections between polymorphic behavior types and thus for better recommendation performance in the sequential recommendation. For example, Qiu et al.[54] proposed a Bayesian personalized ranking model for heterogeneous behavior types (BPRH) that incorporates the target behavior, auxiliary behavior, and negative behavior into a unified model, and the idea might also be applicable for the sequential recommendation.

Learning behavior sequences in real time. Every behavior of the user might reflect a possible interest transfer, in this case, recommendater 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.[101] combined MF, RNN, and GAN in film recommendations to dynamically provide movie recommendations. Shih et al.[66] 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 the sequential recommendation tasks in all areas. Future research can be conducted to design specific models for particular areas by capturing the characteristics of these areas, which is more valuable for real-world applications.

6.2 Conclusions

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

can be viewed a testbed. Finally, we discussed the challenges and provided new potential directions for the research on DL-based sequential recommendation.

REFERENCES

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

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

[3] Trapit Bansal, David Belanger, and Andrew Mccallum. 2016. Ask the GRU : multi-task learning for deep text recommendations. *conference on recommender systems* (2016), 107–114.

[4] Oren Barkan and Noam Koenigstein. 2016. Item2vec: neural item embedding for collaborative filtering. In *MLSP*. 1–6.

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

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

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

[8] Veronika Bogina and Tsvi Kuflik. 2017. Incorporating dwell time in session-based recommendations with recurrent Neural networks. In *CEUR Workshop Proceedings*. 57–59.

[9] Wanyu Chen, Pengjie Ren, Fei Cai, and Maarten de Rijke. 2019. Improving End-to-End Sequential Recommendations with Intent-aware Diversification. *CoRR* abs/1908.10171 (2019).

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

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

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

[13] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. 2019. Are We Really Making Much Progress? A Worrying Analysis of Recent Neural Recommendation Approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems (RecSys '19)*. ACM, New York, NY, USA, 101–109. https://doi.org/10.1145/3298689.3347058

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

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

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

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

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

[19] Jian Wu Changhua Pei Xiao Lin Wenwu Ou Fei Sun, Jun Liu and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. *arXiv preprint arXiv:1904.06690* (2019).

[20] Chen Gao, Xiangnan He, Dahua Gan, Xiangning Chen, Fuli Feng, Yong Li, Tat-Seng Chua, et al. 2018. Learning recommender systems from multi-behavior data. *arXiv preprint arXiv:1809.08161* (2018).

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

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

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

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

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

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

[27] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. (2016).
[28] Balázs Hidasi, Massimo Quadrana, Alexandros Karatzoglou, and Domonkos Tikk. 2016. Parallel recurrent neural network architectures for feature-rich session-based recommendations. In *RecSys*. 241–248.
[29] Kai-Chun Hsu, Szu-Yu Chou, Yi-Hsuan Yang, and Tai-Shih Chi. 2016. Neural network based next-song recommendation. *arXiv preprint arXiv:1606.07722* (2016).
[30] Xiaowen Huang, Shengsheng Qian, Quan Fang, Jitao Sang, and Changsheng Xu. 2018. CSAN: Contextual self-attention network for user sequential recommendation. In *MM*. 447–455.
[31] Dietmar Jannach and Malte Ludewig. 2017. When recurrent neural networks meet the neighborhood for session-based recommendation. In *RecSys*. 306–310.
[32] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems* 20, 4 (2002), 422–446.
[33] Santosh Kabbur, Xia Ning, and George Karypis. 2013. Fism: factored item similarity models for top-n recommender systems. In *SIGKDD*. 659–667.
[34] Wangcheng Kang and Julian Mcauley. 2018. Self-attentive sequential recommendation. *arXiv: Information Retrieval* (2018).
[35] Yehuda Koren. 2009. Collaborative filtering with temporal dynamics. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 447–456.
[36] Artus Krohn-Grimberghe, Lucas Drumond, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2012. Multi-relational matrix factorization using bayesian personalized ranking for social network data. In *WSDM*. 173–182.
[37] Duc-Trong Le, Hady W Lauw, and Yuan Fang. 2018. Modeling contemporaneous basket sequences with twin networks for next-item recommendation. In *IJCAI*. 3414–3420.
[38] Lukas Lerche, Dietmar Jannach, and Malte Ludewig. 2016. On the value of reminders within e-commerce recommendations. In *UMAP*. 27–35.
[39] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *CIKM*. 1419–1428.
[40] Sheng Li, Jaya Kawale, and Yun Fu. 2015. Deep collaborative filtering via marginalized denoising auto-encoder. In *CIKM*. 811–820.
[41] Zhi Li, Hongke Zhao, Qi Liu, Zhenya Huang, Tao Mei, and Enhong Chen. 2018. Learning from history and present: next-item recommendation via discriminatively exploiting user behaviors. In *SIGKDD*. 1734–1743.
[42] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon.com recommendations: item-to-item collaborative filtering. *IEEE Internet Computing* 7, 1 (2003), 76–80.
[43] Kuan Liu, Xing Shi, and Prem Natarajan. 2018. A Sequential Embedding Approach for Item Recommendation with Heterogeneous Attributes. arXiv:cs.IR/1805.11008
[44] Qiang Liu, Shu Wu, Diyi Wang, Zhaokang Li, and Liang Wang. 2016. Context-aware sequential recommendation. *ICDM* (2016), 1053–1058.
[45] Qiang Liu, Shu Wu, and Liang Wang. 2017. Multi-behavioral sequential prediction with recurrent log-bilinear model. *IEEE Transactions on Knowledge and Data Engineering* 29, 6 (2017), 1254–1267.
[46] Qiao Liu, Yifu Zeng, Refuoe Mokhosi, and Haibin Zhang. 2018. STAMP: short-term attention/memory priority model for session-based recommendation. In *SIGKDD*. 1831–1839.
[47] Babak Loni, Roberto Pagano, Martha Larson, and Alan Hanjalic. 2016. Bayesian personalized ranking with multi-channel user feedback. In *RecSys*. 361–364.
[48] Pablo Loyola, Chen Liu, and Yu Hirate. 2017. Modeling user session and intent with an attention-based encoder-decoder architecture. In *RecSys*. 147–151.
[49] Malte Ludewig and Dietmar Jannach. 2018. Evaluation of session-based recommendation algorithms. *arXiv preprint arXiv:1803.09587* (2018).
[50] Mingyuan Ma, Sen Na, Cong Xu, and Xin Fan. 2018. The graph-based broad behavior-aware recommendation system for interactive news. *arXiv preprint arXiv:1812.00002* (2018).
[51] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).
[52] Hanh TH Nguyen, Martin Wistuba, Josif Grabocka, Lucas Rego Drumond, and Lars Schmidt-Thieme. 2017. Personalized deep learning for tag recommendation. In *PAKDD*. 186–197.
[53] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. 1532–1543.
[54] Huihuai Qiu, Yun Liu, Guibing Guo, Zhu Sun, Jie Zhang, and Hai Thanh Nguyen. 2018. BPRH: Bayesian personalized ranking for heterogeneous implicit feedback. *Information Sciences* 453 (2018), 80–98.

[55] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. 2018. Sequence-aware recommender systems. *arXiv preprint arXiv:1802.08452* (2018).
[56] Massimo Quadrana, Alexandros Karatzoglou, Balázs Hidasi, and Paolo Cremonesi. 2017. Personalizing session-based recommendations with hierarchical recurrent neural networks. In *RecSys*. 130–137.
[57] Yogesh Singh Rawat and Mohan S Kankanhalli. 2016. ConTagNet: Exploiting user context for image tag recommendation. In *ACMMM*. 1102–1106.
[58] Pengjie Ren, Zhumin Chen, Jing Li, Zhaochun Ren, Jun Ma, and Maarten de Rijke. 2018. RepeatNet: A repeat aware neural recommendation machine for session-based recommendation. *arXiv preprint arXiv:1812.02646* (2018).
[59] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *UAI*. 452–461.
[60] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *WWW*. 811–820.
[61] Steffen Rendle, Zeno Gantner, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. Fast context-aware recommendations with factorization machines. In *SIGIR*. 635–644.
[62] Massimiliano Ruocco, Ole Steinar Lillestøl Skrede, and Helge Langseth. 2017. Inter-session modeling for session-based recommendation. In *DLRS*. 24–31.
[63] Noveen Sachdeva, Kartik Gupta, and Vikram Pudi. 2018. Attentive neural architecture incorporating song features for music recommendation. In *RecSys*. 417–421.
[64] Noveen Sachdeva, Giuseppe Manco, Ettore Ritacco, and Vikram Pudi. 2019. Sequential Variational Autoencoders for Collaborative Filtering. *WSDM* (2019).
[65] Tian Shi, Yaser Keneshloo, Naren Ramakrishnan, and Chandan K Reddy. 2018. Neural abstractive text summarization with sequence-to-sequence models. *arXiv preprint arXiv:1812.02303* (2018).
[66] Shun-Yao Shih and Heng-Yu Chi. 2018. Automatic, personalized, and flexible playlist generation using reinforcement learning. *arXiv preprint arXiv:1809.04214* (2018).
[67] Ajit P Singh and Geoffrey J Gordon. 2008. Relational learning via collective matrix factorization. In *SIGKDD*. 650–658.
[68] Ayush Singhal, Pradeep Sinha, and Rakesh Pant. 2017. Use of deep learning in modern recommendation system: A summary of recent works. *arXiv preprint arXiv:1712.07525* (2017).
[69] Elena Smirnova and Flavian Vasile. 2017. Contextual sequence modeling for recommendation with recurrent neural networks. In *DLRS*. 2–9.
[70] Harold Soh, Scott Sanner, Madeleine White, and Greg Jamieson. 2017. Deep sequential recommendation for personalized adaptive user interfaces. In *IUI*. 589–593.
[71] Younghun Song and Jae-Gil Lee. 2018. Augmenting recurrent neural networks with high-order user-contextual preference for session-based recommendation. *arXiv preprint arXiv:1805.02983* (2018).
[72] Gabriele Sottocornola, Panagiotis Symeonidis, and Markus Zanker. 2018. Session-based news recommendations. In *WWW'18*. 1395–1399.
[73] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.
[74] Shiming Sun, Yuanhe Tang, Zemei Dai, and Fu Zhou. 2019. Self-Attention Network for Session-Based Recommendation With Streaming Data Input. *IEEE* (2019).
[75] Zhu Sun, Qing Guo, Jie Yang, Hui Fang, Guibing Guo, Jie Zhang, and Robin Burke. 2019. Research commentary on recommendations with side information: A survey and research directions. *Electronic Commerce Research and Applications* 37 (2019), 100879.
[76] Yong Kiam Tan, Xinxing Xu, and Yong Liu. 2016. Improved recurrent neural networks for session-based recommendations. In *DLRS*. 17–22.
[77] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*. 565–573.
[78] Trinh Xuan Tuan and Tu Minh Phuong. 2017. 3D convolutional networks for session-based recommendation with content features. In *RecSys*. 138–146.
[79] Bartłomiej Twardowski. 2016. Modelling contextual information in session-aware recommender systems with neural networks. In *RecSys*. 273–276.
[80] Vladimir Vapnik and Akshay Vashist. 2009. A new learning paradigm: Learning using privileged information. *Neural networks* 22, 5-6 (2009), 544–557.
[81] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *ICONIP*. 5998–6008.
[82] Shengxian Wan, Yanyan Lan, Pengfei Wang, Jiafeng Guo, Jun Xu, and Xueqi Cheng. 2015. Next basket recommendation with neural networks.. In *RecSys Posters*.

[83] Hao Wang, Xingjian Shi, and Dit Yan Yeung. 2016. Collaborative recurrent autoencoder: Recommend while learning to fill in the blanks. *neural information processing systems* (2016), 415–423.
[84] Hao Wang, Naiyan Wang, and Dit Yan Yeung. 2015. Collaborative deep learning for recommender systems. *knowledge discovery and data mining* (2015), 1235–1244.
[85] Pengfei Wang, Jiafeng Guo, Yanyan Lan, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2015. Learning hierarchical representation model for next basket recommendation. In *SIGIR*. 403–412.
[86] Shoujin Wang, Longbing Cao, and Yan Wang. 2019. A survey on session-based recommender systems. *arXiv preprint arXiv:1902.04864* (2019).
[87] Shoujin Wang, Liang Hu, Longbing Cao, Xiaoshui Huang, Defu Lian, and Wei Liu. 2018. Attention-based transactional context embedding for next-item recommendation. In *AAAI*. 2532–2539.
[88] Shoujin Wang, Liang Hu, Yan Wang, Quan Z. Sheng, Mehmet A. Orgun, and Longbing Cao. 2019. Modeling Multi-Purpose Sessions for Next-Item Recommendations via Mixture-Channel Purpose Routing Networks. In *IJCAI*. 3771–3777.
[89] Jian Wei, Jianhua He, Kai Chen, Yi Zhou, and Zuoyin Tang. 2016. Collaborative filtering and deep learning based hybrid recommendation for cold start problem. In *DASC/PiCom/DataCom/CyberSciTech*. 874–877.
[90] Chen Wu and Ming Yan. 2017. Session-aware information embedding for e-commerce product recommendation. In *CIKM*. 2379–2382.
[91] Chao-Yuan Wu, Amr Ahmed, Alex Beutel, Alexander J Smola, and How Jing. 2017. Recurrent recommender networks. In *WSDM*. 495–503.
[92] Shu Wu, Yuyuan Tang, Yanqiao Zhu, Liang Wang, Xing Xie, and Tieniu Tan. 2018. Session-based Recommendation with Graph Neural Networks. *arXiv preprint arXiv:1811.00855* (2018).
[93] Qiaolin Xia, Peng Jiang, Fei Sun, Yi Zhang, Xiaobo Wang, and Zhifang Sui. 2018. Modeling consumer buying decision for recommendation based on multi-task deep learning. In *CIKM*. 1703–1706.
[94] Feng Yu, Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. A dynamic recurrent model for next basket recommendation. In *SIGIR*. 729–732.
[95] Fajie Yuan, Alexandros Karatzoglou, Ioannis Arapakis, Joemon M Jose, and Xiangnan He. 2019. A simple convolutional generative network for next item recommendation. *web search and data mining* (2019), 582–590.
[96] M. Zhang and Z. Yang. 2019. GACOforRec: Session-Based Graph Convolutional Neural Networks Recommendation Model. *IEEE Access* (2019).
[97] Qi Zhang, Jiawen Wang, Haoran Huang, Xuanjing Huang, and Yeyun Gong. 2017. Hashtag recommendation for multimodal microblog using co-attention network.. In *IJCAI*. 3420–3426.
[98] Shuai Zhang, Yi Tay, Lina Yao, and Aixin Sun. 2018. Next item recommendation with self-attention. *arXiv: Information Retrieval* (2018).
[99] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 5.
[100] Yuyu Zhang, Hanjun Dai, Chang Xu, Jun Feng, Taifeng Wang, Jiang Bian, Bin Wang, and Tieyan Liu. 2014. Sequential click prediction for sponsored search with recurrent neural networks. *national conference on artificial intelligence* (2014), 1369–1375.
[101] Wei Zhao, Wenyou Wang, Jianbo Ye, Yongqiang Gao, Min Yang, Zhou Zhao, and Xiaojun Chen. 2017. Leveraging long and short-term information in content-aware movie recommendation. *arXiv preprint arXiv:1712.09059* (2017).
[102] Zhe Zhao, Zhiyuan Cheng, Lichan Hong, and Ed H Chi. 2015. Improving user topic interest profiles by behavior factorization. In *WWW*. 1406–1416.
[103] Chang Zhou, Jinze Bai, Junshuai Song, Xiaofei Liu, Zhengchao Zhao, Xiusi Chen, and Jun Gao. 2017. ATRank: An attention-Based user behavior modeling framework for recommendation. *arXiv preprint arXiv:1711.06632* (2017).
[104] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2018. Graph neural networks: a review of methods and applications. *arXiv preprint arXiv:1812.08434* (2018).