Article

Optimizing Session-Aware Recommenders: A Deep Dive into GRU-Based Latent Interaction Integration

Ming-Yen Lin , Ping-Chun Wu , Sue-Chen Hsueh *

Posted Date: 12 December 2023

doi: 10.20944/preprints202312.0746.v1

Keywords: recommender system; session-aware recommendation; latent-context information; long-term and short-term preference; gated recurrent unit

Article

Optimizing Session-Aware Recommenders: A Deep Dive into GRU-Based Latent Interaction Integration

Ming-Yen Ln [1], Ping-Chun Wu [2] and Sue-Chen Hsueh [3,*]

[1] Dept. of Information Engineering and Computer Science, Feng Chia University, Taiwan; linmy@mail.fcu.edu.tw
[2] Dept. of Information Engineering and Computer Science, Feng Chia University, Taiwan; jun.audis5@gmail.com
[3] Dept. of Information Management, Chaoyang University of Technology, Taiwan; schsueh@cyut.edu.tw
* Correspondence: schsueh@cyut.edu.tw

Abstract: This study introduces session-aware recommendation models, leveraging GRU (Gated Recurrent Unit) and attention mechanisms for advanced latent interaction data integration. A primary advancement is enhancing latent context, a critical factor for boosting recommendation accuracy. We address existing models' rigidity by dynamically blending short-term (most recent) and long-term (historical) preferences, moving beyond static period definitions. Our approaches, pre-combination (LCII-Pre) and post-combination (LCII-Post) with fixed (Fix) and flexible learning (LP) weight configurations, are thoroughly evaluated. We conducted extensive experiments to assess our models' performance on public datasets such as Amazon and MovieLens 1M. Notably, on the MovieLens 1M dataset, LCII-Pre$_{\text{Fix}}$ achieved a 1.85% and 2.54% higher Recall@20 than II-RNN and BERT4Rec+st+TSA, respectively. On the Steam dataset, LCII-Post$_{\text{LP}}$ outperformed these models by 18.66% and 5.5%. Furthermore, on the Amazon dataset, LCII showed a 2.59% and 1.89% improvement in Recall@20 over II-RNN and CAII. These results affirm the significant enhancement our models bring to session-aware recommendation systems, showcasing their potential for both academic and practical applications in the field.

Keywords: recommender system; session-aware recommendation; latent-context information; long-term and short-term preference; gated recurrent unit

1. Introduction

Recommendation mechanisms have emerged as vital tools for filtering information in various aspects of life. They are widely used in commercial platforms, including e-commerce sites like Amazon. Our preferences and purchases change over time. To ensure recommended results align more closely with actual needs, the Sequential Recommendation System (SRS) has gained prominence [1]. SRS emphasizes continuous interaction records with time-series characteristics, based on the assumption that dependencies exist between interactions. Therefore, all user interaction records are essential for a comprehensive understanding. Traditionally, research in this area often used the Recurrent Neural Network (RNN) as the network architecture, yielding positive results [2–4].

The sequential recommendation method utilizes a series of interaction records as its reference basis. To avoid learning incorrect information from irrelevant interaction records, the concept of a session has been introduced. Interaction records within a defined period are considered part of the same session, with interactions processed separately based on the session. This led to the development of both session-based and session-aware recommendations.

The Session-Based Recommender System (SBRS) [2,5,6] aims to reflect users' actual thinking and behavior patterns. It considers only the interaction records within a short period, meaning recommendations are based solely on one session's data. This approach's limitation is its reliance on a narrow data range, leading to less personalized recommendations. It is, however, beneficial for users seeking recommendation system convenience without needing to register or log in.

To facilitate personalized recommendations, the Session-Aware Recommender System (SARS) was developed [7]. SARS involves recommendations comprising multiple sessions [8–11]. Personalized recommendations typically rely on the user's current interaction to infer their short-term behavioral intentions. While short-term behaviors significantly influence future interests and are thus considered as short-term preferences, sessions too distant from the short-term period are classified as long-term preferences. Users have both long-term and short-term preferences, integrated into their overall intention preference for recommendations [1,11]. This method's limitations include overlooking context information and a rigid definition of short-term preferences based on the last session, which can limit recommendation adaptability.

The correlation between a product and the current product represents potential information that should be considered in understanding the real intentions of consumers. Additionally, previous studies have often limited the definition of short-term preferences to the last session [11]. This approach can lead to recommendations that are too rigid and inflexible. This rigidity arises because a session's definition is based on time intervals, as mentioned earlier. If a user's intentions span a period that extends beyond the confines of a single session, this behavior pattern may not accurately represent the user's true intentions.

Our work adopts SARS over SBRS, addressing two unresolved shortcomings: the neglect of context information and the inflexible definition of long-term and short-term preferences. We propose a method that considers context information, typically divided into intra-context and inter-context categories. Intra-context refers to additional messages accompanying interactions, while inter-context, based on the session, pertains to information between the current and past sessions. One challenge with context information is its potential irrelevance to the interaction sequence. Another is its tendency to diminish the emphasis on sequential data in recommendations.

Addressing the relevance of auxiliary context information to sequential interaction records, the hidden state in the Gated Recurrent Unit (GRU) is suggested to be used as the model's context information [1]. This method ensures the generated context information, based on past interaction records, is relevant. It also mitigates the issue of the RNN framework forgetting older messages in long sequence data. Therefore, our paper proposes using the prediction representation of GRU as the latent-context information, derived from learning based on past sequential interaction records.

We also address the inflexibility in defining long-term and short-term preferences, due to the time-based division of sessions. Specifically, we use a long-term and short-term window (W) to flexibly capture user preferences. As shown in Figure 1, setting the window to 45% involves considering 45% of the messages closest to the current message in the 9 interaction records as short-term preferences, with the remainder as long-term preferences. This window method more flexibly identifies the user's actual preferences and divides the range of long-term and short-term preferences accordingly.

Since SBRS lacks the capability for personalized recommendations, we expand our research using SARS. While previous studies considered context information, some limitations were not effectively addressed. Thus, we incorporate latent context information, more closely related to interaction records. Traditional SARS does not consider long-term and short-term preference information along with context information. Our approach, in contrast, considers both as key foundations for improving recommendation performance. Additionally, we use windows to appropriately address the problem of defining long-term and short-term preferences.

In summary, the contributions of this study include:

1. Innovative Combination: We combine GRU networks with attention mechanisms to enhance session-aware recommender systems, focusing on both short-term and long-term user preferences.

2. Latent-Context GRU Model: Introducing the Latent-Context GRU model, a novel approach for capturing latent interaction information, demonstrating improved performance in session-aware recommendation tasks.

3. Comprehensive Evaluation: Rigorous evaluation against current state-of-the-art methods provides a detailed analysis of its effectiveness and potential for further development.

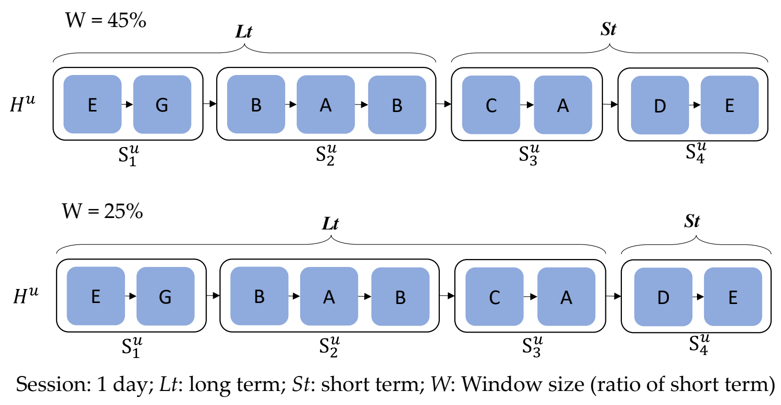

W = 45%
Lt
St
H^u
E
G
B
A
B
C
A
D
E
S_1^u
S_2^u
S_3^u
S_4^u
W = 25%
Lt
St
Session: 1 day; Lt: long term; St: short term; W: Window size (ratio of short term)

Figure 1. Long-Term vs. Short-Term Preferences by Varying Window Scope.

2. Related Work

In traditional recommendation systems, prevalent methods include POP (Most-Popular) and Item-based Nearest Neighbor (Item-kNN) [12]. Subsequent developments have introduced methods like Collaborative Filtering (CF) [13], Matrix Factorization (MF) [14,15], and Markov Chain (MC) [8]. These methods utilize user click data as the basis for recommendations, where clicks during browsing represent varying preferences or interests of users, to predict their next item of interest.

Sequential recommendation, a primary branch of recommendation systems, is further divided into session-based and session-aware recommendations. HCA [1], a hierarchical neural network model, focuses on enhancing short-term interests by capturing the complex correlations between adjacent messages within each time frame. As RNN tends to gradually forget past information due to long-term dependencies, this method helps the system retain information.

GRU4Rec [2], is a recommendation system model employing the RNN approach. This model continually learns from past features through RNN, combined with the timing of data clicks, to construct a highly effective recommendation method at that time. Compared to traditional methods, RNN adds a sequential consideration, with experimental results underscoring its effectiveness and establishing neural methods' status in recommendation systems.

Neural Session-Aware Recommendation (NSAR) [9] operates under session-aware recommendation, incorporating all sessions into the model for learning and predicting the next item. This work discusses the timing of feature integration, positing that different integration opportunities significantly affect recommendation performance, hence proposing two strategies: pre-combine and post-combine.

Inter-Intra RNN (II-RNN) [16] is a two-layer RNN architecture model which can effectively enhance recommendation performance and expedite feature learning. This is because the final prediction representation of the inner network is passed to the initial hidden state of the outer network. The outer network thus does not start learning from scratch, a method whose effectiveness is proven by this research. CAII [17] is another two-layer GRU model which utilizes session information, including item ID, image characteristics, and item price, to compare these features and achieve a balanced CAII. The CAII-P strategy emerged as the best solution in this research, suggesting that image features do not substantially enhance recommendation quality. This also indicates that session information is not directly correlated with sequential data, a key motivation for our study.

The discussed research highlights that personalized recommendations cannot be solely based on session-based methods. Hence, session-aware recommendation research has been extended. Additionally, the use of latent context information as an auxiliary method in recommendation systems has been explored but not extensively developed.

Our work builds on session-aware recommendation, integrating latent-context information and long-term and short-term preference features. We adopt HCA's [1] approach of using GRU prediction representation as an auxiliary feature for model learning, treating such information as latent-context. Drawing from NSAR's [9] pre-combine and post-combine feature combination strategies, we revise the Attention-Gate formula for feature fusion calculations. Our proposed method's main framework follows the inner and outer GRU architecture of II-RNN [16], with a significant difference: while II-RNN inputs only item IDs, our method also incorporates latent content information and designs for processing long-term and short-term preferences.

3. Proposed Method

3.1. Problem Statements

Consider a set of users U, where each user u possesses a historical interaction record $H^u = \{S_1^u, S_2^u, \dots, S_t^u\}$. These records consist of t sessions, each ordered chronologically by the time of interaction. Within each session, denoted as $S_j^u = \{i_1^{u,j}, i_2^{u,j}, \dots, i_v^{u,j}\}$, there are v items with which the user has interacted, also arranged in the order of interaction within that session. The objective is to predict a set of items $\{i_{r1}, i_{r2}, \dots, i_{rk}\}$ that the user is most likely to interact with next. These predicted items are typically ranked in descending order of interaction probability. Table 1 lists the symbols used in this paper, with additional terms introduced in the subsequent sections.

Table 1. Notations.

Notation	Description
$I = \{i_1, i_2, \dots, i_M\}$	Set of items, i_x is the item ID ($1 \le x \le M$)
$U = \{u_1, u_2, \dots, u_N\}$	Set of users, u_y is the user ID ($1 \le y \le N$)
$H^u = \{S_1^u, S_2^u, \dots, S_k^u\}$	User u's historical interactions, S_j^u is the j-th session ($1 \le y \le k$)
$S_j^u = \{i_1^{u,j}, i_2^{u,j}, \dots, i_v^{u,j}\}$	Interaction items in the j-th session, $i_p^{u,j}$ is the p-th item ($1 \le p \le v$)
$R_{i_v^{u,j}}$	Embedding representation of item $i_v^{u,j}$
$C_{i_v^{u,j}}$	Latent context representation of interactions up to item $i_v^{u,j}$
$F_{i_v^{u,j}}$	Preference representation up to $i_v^{u,j}$

3.2. LCII Architecture

Figure 2 depicts the architecture of the proposed method, named LCII (**L**atent **C**ontext **II**). The architecture consists of five modules: Embedding Module, Context Generation Module, Representation Fusion Module, Sequence Processing Module, and Prediction Module.

Specifically, the Embedding Module first generates a random number table, tailored to the size of the input data. It then converts the item ID into an embedded representation, which becomes the input for the model.

The Context Generation Module generates potential context information that is considered for recommendations. As GRU (Inner) processes the embedded representation sequentially, a predicted representation is generated one-by-one, based on the latent features learned previously, and thus, it is closely related to the interactive Item. The prediction representation generated by GRU (Inner) is treated as this potential context information.

The Representation Fusion Module activates after acquiring potential context information. It employs two distinct strategies for this purpose: Pre-combining and Post-combining. These strategies are used to create the Preference Representation. In essence, these strategies integrate information from two key processes: feature fusion and term fusion. Feature fusion involves extracting session features by merging item embeddings with latent context information, effectively capturing the current session's characteristics. Meanwhile, term fusion focuses on aggregating latent context information that reflects both long-term and short-term user preferences. This dual approach in the Representation Fusion Module ensures a comprehensive integration of immediate session data with overarching user preference trends.

Subsequently, the Sequence Processing Module uses GRU (Outer) for learning and prediction. The GRU (Outer) accepts the first prediction representation from GRU (Inner) as the initial state, receives sequentially the preference representation, and produces the final prediction. The predictions are fed into the Prediction Module. Prediction Module ultimately produces the user's recommendation list and provides the prediction evaluation.

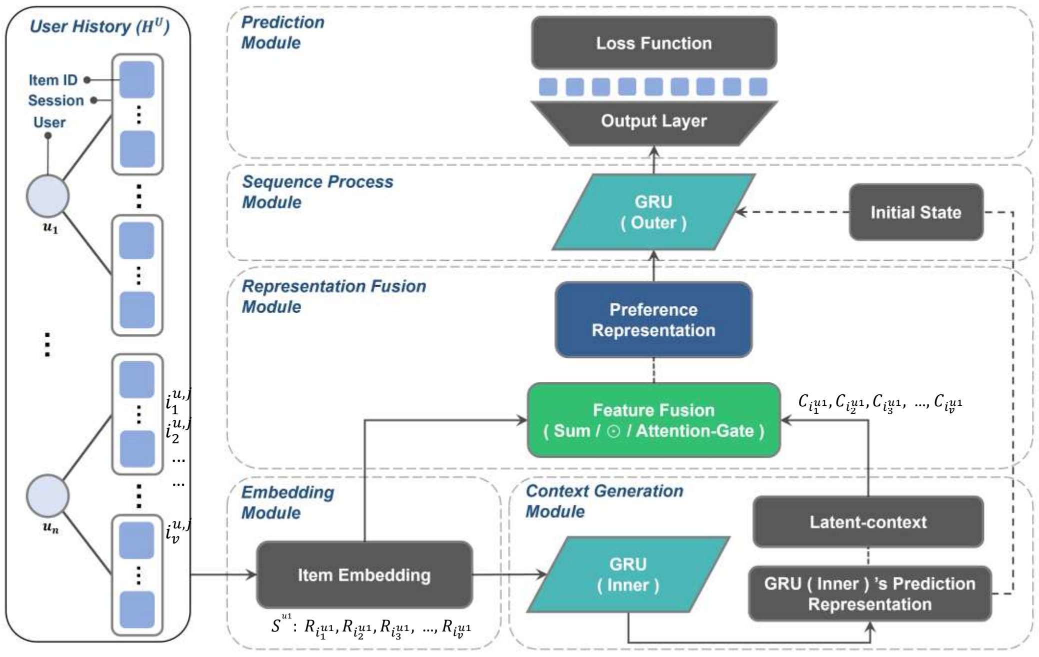

User History (H^U)
Item ID
Session
User
u_1
u_n
$i_1^{u,j}$
$i_2^{u,j}$
$i_v^{u,j}$
Prediction Module
Loss Function
Output Layer
Sequence Process Module
GRU (Outer)
Initial State
Representation Fusion Module
Preference Representation
Feature Fusion (Sum / ⊙ / Attention-Gate)
$C_{i_1^{u1}}, C_{i_2^{u1}}, C_{i_3^{u1}}, \ldots, C_{i_v^{u1}}$
Embedding Module
Item Embedding
S^{u1}: $R_{i_1^{u1}}, R_{i_2^{u1}}, R_{i_3^{u1}}, \ldots, R_{i_v^{u1}}$
Context Generation Module
GRU (Inner)
Latent-context
GRU (Inner) 's Prediction Representation

Figure 2. LCII Architecture.

3.2.1. Embedding Module

We use Item2vec [18] to convert each item ID into an embedding vector. A lookup table for all the items is constructed as an embedding matrix of size $M \times d$, where M is the total number of items and d is the embedding size. A user session S_j^{u1} of v items $i_1^{u1,j}$, $i_2^{u1,j}$, …, $i_v^{u1,j}$ is then embedded, using the lookup table. The resulting embeddings, $R_{i_1^{u1,j}}$, $R_{i_2^{u1,j}}$, …, $R_{i_v^{u1,j}}$, are the Item Embeddings of the user session. Consider when the processing of user u reaches the p-th item, denoted as $i_p^{u,j}$, in the j-th session S_j^u. All items from the first session up to this p-th item collectively form the current set of input item embeddings. These embeddings are then processed to extract their latent contexts. This process results in the division of the embeddings into long-term and short-term preferences, which are handled in the subsequent modules.

3.2.2. Context Generation Module

The output from the Embedding Module constitutes the input for the Context Generation Module. The primary objective of this research is to identify and capture latent interactive information, herein referred to as 'Latent-context'. The process for generating this content information commences with the input of the embedding-transformed representation into a GRU, designated as the GRU (Inner). That is, $R_{i_1^{u1,j}}$, $R_{i_2^{u1,j}}$, …, $R_{i_v^{u1,j}}$, goes through the GRU (Inner) and produces the latent context $C_{i_1^{u1}}, C_{i_2^{u1}}, C_{i_3^{u1}}, \ldots, C_{i_v^{u1}}$. Within this neural network framework, user information is subject to a learning process.

Upon each iteration through the GRU, two distinct outcomes are yielded: the Hidden State and the current Prediction Representation. The Prediction Representation, as derived from the GRU, is conceptualized as Latent-context and is subsequently stored within a dedicated Latent-context list. This list is further employed by another GRU, named GRU (Outer). The rationale for selecting this specific information as Latent-context is based on the methodology of generating the Prediction Representation at any given temporal point. This involves using both the Hidden State and the current interaction record as inputs. The Hidden State represents a cumulative preference profile, developed through the analysis of historical interaction records up to the current moment. Consequently, the Prediction Representation, grounded in these derived preferences, facilitates the generation of user-specific items of potential interest.

3.2.3. Representation Fusion Module

Upon completion of the initial two modules, we acquire two key outputs: the Item Embedding and the Latent Context. These outputs are subsequently utilized as inputs for the Representation Fusion Module. The primary aim of this module is to integrate the two features to derive a unified User Preference Representation.

There are three ways to combine two representations into a new representation having the same number of dimension in this study: Vector Addition, Element-wise Vector Multiplication, and the Attention-Gate Mechanism [9]. Vector Addition involves the straightforward process of summing corresponding vector elements. Conversely, Element-wise Vector Multiplication entails the multiplication of corresponding elements in the vectors. The Attention-Gate Mechanism, adapted from the formula outlined in [9] and modified to exclude the phase content vectors from that study, aims to identify and emphasize the most salient features among the comparison set. This mechanism operates by assigning a calculated weight (scalar) to these features based on their relevance, which is then multiplied back into the original inputs and merged through element-wise multiplication.

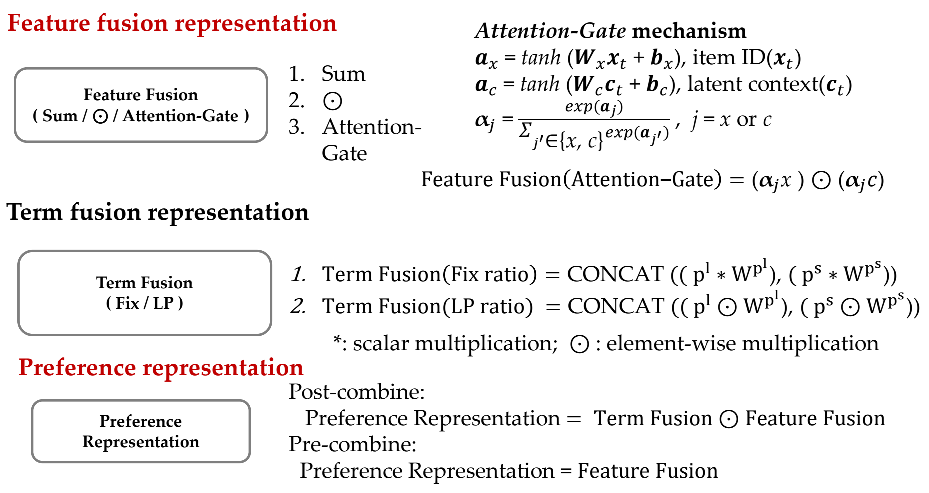
The basic LCII integrates the Item Embedding and the Latent Context using one of the three ways, without the engagement of long and short terms, and generates the User Preference Representation. The basic LCII can be extended with two strategies, both operate with term fusion representation to generate the Preference Representation: Pre-combining and Post-combining.

Within the context of Term Fusion [9], the computation of long-term and short-term preferences uses latent-context list as a source, divides the list based on the setting ratio of the window size. For example, setting the window of 30% for a sequence of 19 items will grab 30% of the nearest neighbor information as a short-term preference (p^s). The results are ($S_{14}^{u1}\dots,\ i_{19}^{u1}$) as short-term preference and ($S_{1}^{u1}\dots,\ i_{13}^{u1}$) as long-term preference (p^l). LCII sets the window size using either the Fixed ratio (Fix) or the Learning Parameters ratio (LP). The Fix sets a fixed window size, which is a pre-determined parameter. The LP learns the window size dynamically from the data. The key point of learning is that the weight matrix is generated based on the size of the range divided by the long-term and short-term windows, so the long-term and short-term preference information within the range can be learned and the weights will be updated and learned from iterations.

Thus, LCII-Pre combines the item embedding with the term fusion representation, again, using one of the three ways three ways (Vector Addition, Element-wise Vector Multiplication, and the Attention-Gate Mechanism). As shown in Figure 2, the LCII combines the term fusion representation with the item embedding into the feature fusion representation. This representation is directly used as the Preference Representation for next module. The strategy used is called *pre-combining,* and this method is referred to as the **LCII-Pre** method.

Alternatively, the LCII may use another strategy to combine term fusion representation to generate the Preference Representation. The item embedding is first combined with the latent context, using one of the three ways (Vector Addition, Element-wise Vector Multiplication, and the Attention-Gate Mechanism), to become the feature fusion representation. The Preference Representation is generated by the Element-wise Vector Multiplication between feature fusion representation and term fusion representation. The strategy used is called *post-combining,* and this method is referred to as the **LCII-Post** method.

This process is graphically depicted in Figure 3, depicting the operation of the fusion module. The representation fusion of LCII-Pre is shown in Figure 4, while that of LCII-Post is shown in Figure 5. The Preference Representation serves as the input to the Outer GRU in the subsequent module, as described next.



Feature fusion representation

Feature Fusion (Sum / ⊙ / Attention-Gate)

1. Sum
2. ⊙
3. Attention-Gate

***Attention-Gate* mechanism**

$\boldsymbol{a}_x = tanh\,(\boldsymbol{W}_x\boldsymbol{x}_t + \boldsymbol{b}_x)$, item ID($\boldsymbol{x}_t$)

$\boldsymbol{a}_c = tanh\,(\boldsymbol{W}_c\boldsymbol{c}_t + \boldsymbol{b}_c)$, latent context($\boldsymbol{c}_t$)

$\boldsymbol{\alpha}_j = \frac{exp(\boldsymbol{a}_j)}{\Sigma_{j'\in\{x,\,c\}} exp(\boldsymbol{a}_{j'})}$, $j = x$ or c

Feature Fusion(Attention−Gate) = $(\boldsymbol{\alpha}_j x) \odot (\boldsymbol{\alpha}_j c)$

Term fusion representation

Term Fusion (Fix / LP)

1. Term Fusion(Fix ratio) = CONCAT (($p^l * W^{p^l}$), ($p^s * W^{p^s}$))
2. Term Fusion(LP ratio) = CONCAT (($p^l \odot W^{p^l}$), ($p^s \odot W^{p^s}$))

*: scalar multiplication; ⊙ : element-wise multiplication

Preference representation

Preference Representation

Post-combine:
Preference Representation = Term Fusion ⊙ Feature Fusion
Pre-combine:
Preference Representation = Feature Fusion

Figure 3. Representation Fusion Module.

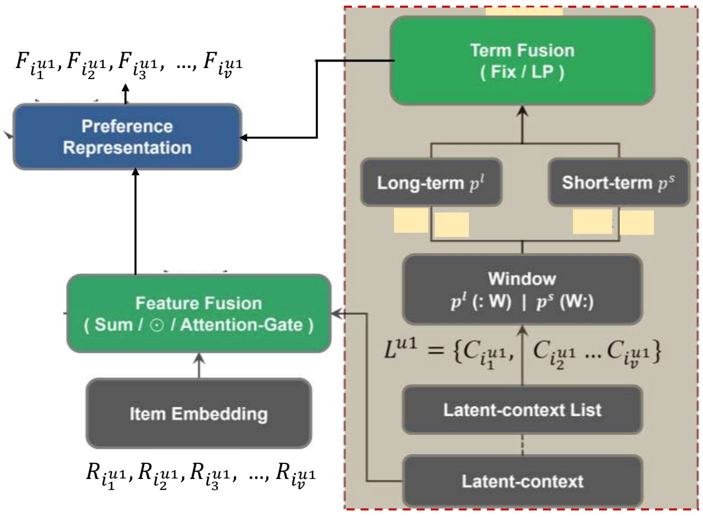

$F_{i_1^{u1}}, F_{i_2^{u1}}, F_{i_3^{u1}}, \ldots, F_{i_v^{u1}}$
Preference Representation
Term Fusion (Fix / LP)
Long-term p^l
Short-term p^s
Window p^l (: W) | p^s (W:)
$L^{u1} = \{C_{i_1^{u1}}, C_{i_2^{u1}} \ldots C_{i_v^{u1}}\}$
Feature Fusion (Sum / ⊙ / Attention-Gate)
Item Embedding
Latent-context List
Latent-context
$R_{i_1^{u1}}, R_{i_2^{u1}}, R_{i_3^{u1}}, \ldots, R_{i_v^{u1}}$

Figure 4. Representation Fusion Module – Pre.

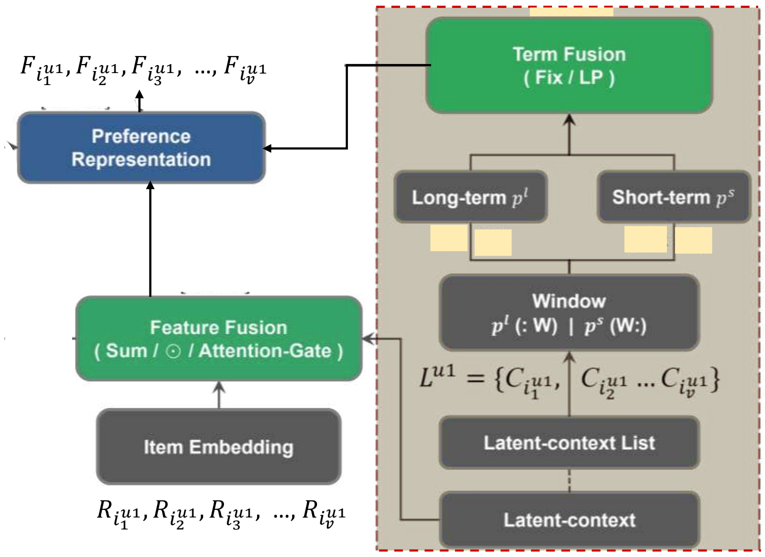

$F_{i_1^{u1}}, F_{i_2^{u1}}, F_{i_3^{u1}}, \ldots, F_{i_v^{u1}}$
Preference Representation
Term Fusion (Fix / LP)
Long-term p^l
Short-term p^s
Window p^l (: W) | p^s (W:)
$L^{u1} = \{C_{i_1^{u1}}, C_{i_2^{u1}} \ldots C_{i_v^{u1}}\}$
Feature Fusion (Sum / ⊙ / Attention-Gate)
Item Embedding
Latent-context List
Latent-context
$R_{i_1^{u1}}, R_{i_2^{u1}}, R_{i_3^{u1}}, \ldots, R_{i_v^{u1}}$

Figure 5. Representation Fusion Module – Post.

3.2.4. Sequence Processing Module

This module involves integrating the Preference Representation with the predictive output of the inner GRU into the outer GRU. A crucial aspect to consider is the discrepancy between the dimensions of the temporary message (output from inner GRU) and the dimensions of the hidden state. To address this, we employ the embedding details of the initial element of the predictive representation to establish the initial state for the GRU's hidden state. This approach, as delineated in the study [16], primarily addresses the 'cold start' challenge commonly encountered in Recurrent Neural Networks (RNNs) at the onset [10,16,19]. By assigning an initial value to the initial Hidden State within the recursive neural network, the model can more rapidly adapt to learning and predicting tasks. Consequently, the use of temporary messages, particularly the adoption of the inner GRU's predictive output as the initial value, plays a pivotal role in expediting the learning of user preferences and enhancing the efficacy of predictions in GRU-based models.

3.2.5. Prediction Module

The prediction module operates as follows: First, it derives a representation from user preferences during the sequence processing phase. This representation is processed by a Feedforward Neural Network, which adjusts the embedding to match the original interaction record's format. The module then compares this representation to the actual data (ground truth), producing scores that rank items by relevance. The model's accuracy is evaluated using the softmax cross-entropy loss function, comparing the predicted and actual data across all user sessions. This process trains the LCII model to make accurate item recommendations. In the final step, the model follows the Top@k recommendation approach to generate a list of recommended items.

4. Experimental Results

4.1. Datasets and Pre-processing

Three well-known datasets were used in the experiments: Steam [20], MovieLens 1M [21], and Amazon [22]. The Steam dataset comprises video game platform review data, featuring 2,567,538 users, 15,474 items, and 7,793,069 interaction records. The MovieLens 1M dataset, known for its movie rating data, includes information about item categories and encompasses 1,000,209 interaction records. The Amazon e-commerce dataset focuses on clothing, shoes, and jewelry shopping data from 2012 to 2014, containing 402,093 users, 140,116 items, and 770,290 interaction records. With regard to the data preprocessing method, Steam and MovieLens 1M datasets were preprocessed following the method described in [10], while the Amazon dataset followed the method in [17].

The preprocessing details are as follows: For MovieLens 1M and Steam, the session time division is set to one day (60*60*24 seconds); the maximum session length is capped at 200 for MovieLens 1M and 15 for Steam. The data filtering conditions are consistent across datasets. For each user, there must be at least two sessions, and each session must have a minimum of two interactions. Each interaction (i.e., item ID) should occur at least five times, and the number of user interaction records (i.e., interaction number of items) should be five or more. Additionally, the ratio of training to testing data is set at 8:2.

For the Amazon dataset, session division is based on interactions occurring at the same time point, with a maximum session length of 20 and a minimum of three sessions. The maximum session lengths for Steam, MovieLens 1M, and Amazon are 15, 200, and 20, respectively. The ratio of training to testing is also 8 to 2. Table 2 lists the characteristics of datasets after pre-processing.

Table 2. Characteristics of Datasets after Pre-processing.

Description	**Amazon**	***MovieLens 1M***	***Steam***
Number of users	9,733	1,196	6,330
Number of items	46,959	3,328	4,332
Number of actions	700,960	158,498	49,164

4.2. Evaluation Metrics and Parameter Settings

In the experiment, this study employs Recall@K, MRR@K, and NDCG@K as the evaluation metrics. The calculation of Recall@K is based on the top K predictions, with the K value set to 5, 10, and 20 as evaluation criteria. The optimal experimental parameter settings for each dataset are as follows. Amazon Configuration: Window ratio is set to 65, and the ratio of long-term to short-term preference in the fixed ratio strategy is 0.5/0.5. The number of iterations is 100, and the learning rate is set at 0.001. Steam Configuration: Window ratio is 4, with the long-term to short-term preference ratio in the fixed proportional strategy set to 0.2/0.8. The number of iterations is 200, and the learning rate is 0.001. MovieLens 1M Configuration: Window ratio is 30, and the long-term to short-term preference ratio in the fixed ratio strategy is 0.8/0.2, with 200 iterations. The learning rate is 0.01.

The feature fusion method utilizes the Attention-Gate, and the term fusion method employs a fixed ratio. Other hyperparameters, set as common parameters, include an embedding/GRU hidden unit size of 80, inner/outer GRU layers of 1, a batch size of 100, and a dropout rate of 0.8. The experimental optimizer is the Adam optimizer, and the loss function used is the Cross Entropy Loss Function.

4.3. Baseline Models

To evaluate the effectiveness of our proposed method, we conducted comparisons with the following well-established methods:

- Most-Popular: This is one of the most commonly used recommendation methods, which suggests the item that appears most frequently in each session.
- Item-kNN [12]: A prevalent recommendation method that relies on the scores generated by users' ratings of items. It employs cosine similarity to recommend items with similar attributes.
- II-RNN [16]: This model is a session-aware model which leverages the architecture of inner and outer GRU as its primary framework for model learning.
- CAII [17]: This session-aware model is an extension of the method outlined in [16]. CAII incorporates context information, such as images and prices, into the model input. For comparison, we use the CAII-P strategy, which has shown the best performance in its experimental results.
- SASRec [23]: The method is based on the Transformer architecture. SASRec models the entire user sequence using a Casual Attention Mask to consider item IDs for recommendations.
- BERT4Rec [24]: A recommendation method based on the BERT architecture, which employs bidirectional Self-Attention to model user behavior sequences. Through the Cloze task [25], also known as the masked language model [26], it learns and predicts recommendations.
- BERT4Rec (+ST+TSA) [10]: This is the most recent session-aware method, building upon the foundation set by [24].

4.4. Performance Comparisons

4.4.1. Performance of Different Feature Fusion Strategies

Given that the Amazon dataset is compatible with all recommendation methods under consideration, we initially selected the feature fusion method based on this dataset. The optimal solution identified here was then applied to subsequent experimental analyses. For these experiments, we utilized the LCII model. Three feature fusion strategies were examined: Sum, Element-wise Multiplication ($\odot$), and the Attention-Gate mechanism. The experimental results are presented in Table 3, where the best result is highlighted in bold and the second-best score is underlined, as will be marked in subsequent charts.

The Attention-Gate mechanism mitigates conflicts in fusion between messages with differing feature content. By leveraging calculated weights, Attention-Gate assesses the influence of each feature on the model, enabling it to prioritize certain feature information over others during different time periods. This approach effectively reduces the dominance of one feature type over another. In terms of prediction accuracy, without considering sorting, Attention-Gate performs the best. When sorting indices are taken into account, Attention-Gate still competes closely with the other two fusion methods. Therefore, based on these experimental results, the Attention-Gate mechanism will be used for experimental analysis in subsequent comparison models.

Table 3. Performance w.r.t. Feature Fusion Strategies.

Feature Fusion	Recall @5	MRR @5	NDCG @5	Recall @10	MRR @10	NDCG @10	Recall @20	MRR @20	NDCG @20
Sum	0.2313	**0.2208**	0.1973	0.2339	**0.2212**	0.2132	0.2369	**0.2214**	0.2164
Element-wise Multiplication	0.2393	0.2194	0.2043	0.2475	0.2205	0.2224	0.2545	0.2210	0.2270
Attention-Gate	**0.2407**	0.2157	**0.2060**	**0.2487**	0.2168	**0.2244**	**0.2553**	0.2173	**0.2300**

4.4.2. Performance of Different Windows

In the experiments, we will utilize the Learning Parameters ratio (LP) in LCII-Post Term Fusion as the experimental model. The experiments in this section aim to validate the effectiveness of the long-term and short-term strategies and to examine the division between them. An inadequately sized window, either too small or too large, may result in suboptimal performance. Finding the right balance between long and short-term considerations is a crucial aspect of this experimental phase.

Table 4 presents the experimental results on the Amazon dataset. When the window is set to 65%, the MRR reaches its optimum, suggesting that this setting effectively brings the real recommendation target closer to the top of the list. In other words, using 65% of the information in the capture window as a short-term preference can effectively prioritize the recommendation target. More experimental results are summarized here: For the MovieLens 1M dataset, the optimal window setting is 30%, with 35% being the second best. For the Steam dataset, the best parameter setting is a window of 4%, followed closely by 3%. These experiments demonstrate that different datasets have unique characteristics for the optimal parameter. Subsequent experiments will therefore employ the optimal window setting for each dataset as a fixed parameter for comparative analysis.

Table 4. Performance w.r.t. Window Size.

Window Size (%)	Recall @5	MRR @5	NDCG @5	Recall @10	MRR @10	NDCG @10	Recall @20	MRR @20	NDCG @20
20	0.2356	0.2118	0.2009	0.2448	0.2131	0.2200	0.2522	0.2136	0.2268
25	0.2364	0.2119	0.2015	0.2451	0.2131	0.2198	0.2537	0.2137	0.2269
30	0.2343	0.2118	0.2009	0.2433	0.2130	0.2190	0.2512	0.2136	0.2263
35	0.2340	0.2098	0.2002	0.2434	0.2111	0.2190	0.2523	0.2117	0.2266
40	0.2357	0.2118	0.2008	**0.2455**	0.2131	0.2197	**0.2540**	0.2137	**0.2275**
45	0.2356	0.2111	0.2013	0.2454	0.2124	0.2200	0.2534	0.2130	0.2274
50	0.2357	0.2120	0.2014	0.2447	0.2133	0.2198	0.2526	0.2138	0.2262
55	**0.2366**	0.2117	0.2009	**0.2455**	0.2128	0.2198	0.2532	0.2134	0.2272
60	0.2359	0.2132	0.2014	0.2445	0.2143	0.2205	0.2524	0.2149	0.2274

65	0.2365	**0.2134**	**0.2017**	0.2451	**0.2145**	**0.2206**	0.2517	**0.2150**	0.2271
70	0.2357	0.2122	0.2013	0.2446	0.2134	0.2201	0.2521	0.2139	0.2268
75	0.2344	0.2100	0.2008	0.2429	0.2111	0.2191	0.2527	0.2118	0.2274
80	0.2344	0.2115	0.2011	0.2432	0.2127	0.2192	0.2519	0.2133	0.2262

4.4.3. Long-term and Short-term Fixed Ratio Performance

In this section, we assess the recommendation performance using a fixed ratio (Fix) for long/short-term preferences. The experiment tested various parameter settings, with combinations of [long-term/short-term] preferences set at [0.8/0.2], [0.5/0.5], and [0.2/0.8]. Additionally, extreme setting combinations of [1/0] and [0/1] were evaluated, where the former exclusively considers long-term preferences and completely disregards short-term ones, while the latter does the opposite.

The experimental results are summarized below. From the Amazon dataset, it is evident that both Post-combine and Pre-combine strategies achieve optimal performance with a fixed ratio of [0.5/0.5]. This finding suggests that the representation model requires a balanced consideration of both short-term and long-term preferences for best results. For the MovieLens 1M dataset, the Post-combine outcome mirrors that of Amazon, with the optimal configuration for Pre-combine being [0.8/0.2]. Notably, focusing solely on short-term preference [0/1] deteriorates model performance. Incorporating long-term information significantly enhances results, yet exclusively considering long-term preference [1/0] is not the most effective strategy.

Furthermore, the Steam dataset results indicate a bias towards short-term preference, implying that information features significantly impact dataset performance. Consequently, the best setting for Post-combine is [0.2/0.8], and for Pre-combine, it is [0/1].

4.4.4. Overall Comparisons

The overall experimental results are presented in Tables 5, 6, 7, respectively, for datasets Amazon, MoveiLens 1M, and Steam. Most-Popular and Item-kNN are not included when their performance are far worse than the others. Our proposed method demonstrates the best performance in most metrics across the three datasets. Specifically, LCII yields the best results for Amazon, LCII-Pre$_{Fix}$ for MovieLens 1M, and LCII-Post$_{Fix}$ for Steam. In the MovieLens 1M dataset, LCII-Pre$_{Fix}$ outperforms II-RNN and BERT4Rec(+ST+TSA) in Recall@20 by 1.85% and 2.54% respectively, translating to relative increases of 7% and 9.9%. In the Steam dataset, LCII-Post$_{LP}$ shows a 18.66% and 5.5% higher performance in Recall@20 compared to II-RNN and BERT4Rec(+ST+TSA), corresponding to relative improvements of 39.88% and 9.2%. For Amazon's Recall@20, LCII surpasses II-RNN and CAII by 2.59% and 1.89%, respectively, indicating performance boosts of 11.3% and 8%.

Table 5. Overall Performance w.r.t. Amazon Dataset.

Model	Recall @5	MRR @5	NDCG @5	Recall @10	MRR @10	NDCG @10	Recall @20	MRR @20	NDCG @20
Most-Popular	0.0041	0.0022	0.0026	0.0075	0.0025	0.0039	0.0144	0.0030	0.0065
Item-kNN	0.2172	0.1796	0.1820	0.2239	0.1804	0.2044	0.2260	0.1806	0.2143
II-RNN	0.2238	0.2139	0.1926	0.2264	0.2143	0.2080	0.2294	0.2145	0.2117
CAII-P	0.2295	0.2115	0.1929	0.2342	0.2121	0.2081	0.2364	0.2123	0.2120
SASRec	0.0908	-	0.0821	0.1007	-	0.0853	0.1102	-	0.0877

BERT4Rec	0.1093	-	0.0900	0.1205	-	0.0936	0.1327	-	0.0967
BERT4Rec$_{+ST+TSA}$	0.1231	-	0.1060	0.1343	-	0.1096	0.1409	-	0.1113
LCII	**0.2407**	**0.2157**	**0.2060**	**0.2487**	**0.2168**	**0.2244**	**0.2553**	**0.2173**	**0.2300**
LCII-Pre$_{LP}$	0.2360	0.2064	0.2017	0.2450	0.2077	0.2204	0.2529	0.2082	0.2275
LCII-Pre$_{Fix}$	0.2368	0.2119	0.2019	0.2467	0.2133	0.2219	0.2538	0.2138	0.2281
LCII-Post$_{LP}$	0.2357	0.2127	0.2008	0.2446	0.2140	0.2205	0.2528	0.2145	0.2279
LCII-Post$_{Fix}$	0.2371	0.2147	0.2029	0.2442	0.2156	0.2212	0.2519	0.2162	0.2273

Table 6. Overall Performance w.r.t. MoveiLens 1M Dataset.

Model	Recall @5	MRR @5	NDCG @5	Recall @10	MRR @10	NDCG @10	Recall @20	MRR @20	NDCG @20
II-RNN	0.4196	0.3955	0.3945	0.4401	0.3982	0.4158	0.4679	0.4001	0.4277
SASRec	0.4945	-	0.4160	0.5218	-	0.4675	0.5551	-	0.4760
BERT4Rec	0.4938	-	0.4622	0.5267	-	0.4729	0.5634	-	0.4820
BERT4Rec$_{+ST+TSA}$	0.5423	-	0.5175	0.5692	-	0.5262	0.5995	-	0.5338
LCII	0.5904	0.5353	0.5430	0.6192	0.5391	0.5778	0.6514	0.5414	0.5925
LCII-Pre$_{LP}$	0.5860	0.5329	0.5409	0.6151	0.5368	0.5745	0.6450	0.5388	0.5885
LCII-Pre$_{Fix}$	0.5923	0.5360	0.5453	0.6200	0.5397	0.5805	0.6527	0.5420	0.5965
LCII-Post$_{LP}$	0.5936	0.5364	0.5449	**0.6240**	0.5404	0.5807	**0.6545**	0.5426	0.5945
LCII-Post$_{Fix}$	**0.5961**	**0.5410**	**0.5506**	0.6232	**0.5446**	**0.5837**	0.6526	**0.5466**	**0.5978**

Table 7. Overall Performance w.r.t. Stream Dataset.

Model	Recall @5	MRR @5	NDCG @5	Recall @10	MRR @10	NDCG @10	Recall @20	MRR @20	NDCG @20
II-RNN	0.1160	**0.0632**	0.0183	0.1761	0.0711	0.0440	0.2636	0.0770	0.0981
SASRec	0.0920	-	**0.0553**	0.1522	-	**0.0747**	0.2400	-	0.0968
BERT4Rec	0.0686	-	0.0409	0.1396	-	0.0637	0.2308	-	0.0866
BERT4Rec$_{+ST+TSA}$	0.0870	-	0.0513	0.1472	-	0.0707	0.2567	-	0.0981
LCII	0.1144	0.0598	0.0214	0.1809	0.0685	0.0501	0.2746	0.0749	0.1088
LCII-Pre$_{LP}$	0.0921	0.0467	0.0191	0.1518	0.0545	0.0442	0.2373	0.0604	0.0959
LCII-Pre$_{Fix}$	**0.1183**	0.0624	0.0231	**0.1852**	**0.0712**	0.0527	**0.2821**	**0.0778**	**0.1138**

LCII-Post$_{LP}$	0.0922	0.0479	0.0194	0.1483	0.0553	0.0433	0.2354	0.0612	0.0946
LCII-Post$_{Fix}$	0.1141	0.0606	0.0209	0.1800	0.0694	0.0482	0.2699	0.0755	0.1041

5. Conclusions

Our study introduces a groundbreaking approach to session-aware recommendations, emphasizing the integration of latent-context features with both long-term and short-term user preferences. The novel methodologies, particularly the LCII-Post and LCII-Pre models, have demonstrated their superiority over traditional recommendation models in our experiments with the MovieLens 1M and Steam datasets. This success highlights the potential of our approach in enhancing the accuracy and relevance of recommendations.

The practical implications of our findings are noteworthy. The LCII model, in particular, emerges as a robust solution for real-world application scenarios. Its balance of performance efficiency and lower execution time costs makes it an attractive option for deployment in various recommendation system environments.

Looking towards future research, there are several promising avenues to explore. The scalability of our proposed models to larger and more diverse datasets represents a significant area for further investigation. Additionally, the integration of more nuanced contextual information, such as user demographics or temporal usage patterns, could further refine the accuracy and applicability of our recommendations.

Author Contributions: Conceptualization, Ming-Yen Lin; Formal analysis, Ming-Yen Lin; Investigation, Ming-Yen Lin and Ping-Chun Wu; Methodology, Ming-Yen Lin, Ping-Chun Wu and Sue-Chen Hsueh; Supervision, Ming-Yen Lin; Writing – original draft, Ping-Chun Wu; Writing – review & editing, Sue-Chen Hsueh.

Funding: This research was funded by National Science and Technology Council, Taiwan, under grant number MOST109-2221-E-035-064 and by Feng Chia University, Taiwan, under grant number 22H00310.

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

References

1. Cui, Q.; Wu, S.; Huang, Y.; Wang, L. A Hierarchical Contextual Attention-Based GRU Network for Sequential Recommendation. arXiv:1711.05114. 2017. Available online: https://arxiv.org/abs/1711.05114 (accessed on 2022/9/23).
2. Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; Tikk, D. Session-Based Recommendations with Recurrent Neural Networks. In Proceedings of the International Conference on Learning Representations, 2015. Available online: https://arxiv.org/abs/1511.06939 (accessed on 2022/10/20).
3. Hidasi, B.; Karatzoglou, A.; Quadrana, M.; Tikk, D. Parallel Recurrent Neural Network Architectures for Feature-rich Session-Based Recommendations. In Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16), 2016; pp. 241-248, doi: 10.1145/2959100.2959167.
4. Hidasi, B.; Karatzoglou, A. Recurrent Neural Networks with Top-k Gains for Session-Based Recommendations. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management (CIKM '18), 2018; pp. 843-853, doi: 10.1145/3269206.3271761.
5. Li, J.; Ren, P.; Chen, Z.; Ren, Z.; Lian, T.; Ma, J. Neural Attentive Session-Based Recommendation. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM '17), 2017; pp. 1419-1428, doi: 10.1145/3132847.3132926.
6. Song, W.; Wang, S.; Wang, Y.; Wang, S. Next-Item Recommendations in Short Sessions. In Proceedings of the 15th ACM Conference on Recommender Systems (RecSys '21), 2021; pp. 282-291, doi: 10.1145/3460231.3474238.
7. Guo, Y.; Zhang, D.; Ling, Y.; Chen, H. A Joint Neural Network for Session-Aware Recommendation. IEEE Access 2020, 8, 74205-74215, doi: 10.1109/ACCESS.2020.2984287.
8. Gu, W.; Dong, S.; Zeng, Z. Increasing Recommended Effectiveness with Markov Chains and Purchase Intervals. Neural Computing and Applications 2014, 25, 5, 1153-1162.
9. Phuong, T.M.; Thanh, T.C.; Bach, N.X. Neural Session-Aware Recommendation. IEEE Access 2019, 7, 86884-86896, doi: 10.1109/ACCESS.2019.2926074.

10. Seol, J.J.; Ko, Y.; Lee, S.G. Exploiting Session Information in BERT-Based Session-Aware Sequential Recommendation. In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '22), 2022; pp. 2639-2644, doi: 10.1145/3477495.3531910.
11. Gu, G.Y.; Yanxiang, L.; Chen, H. A Neighbor-Guided Memory-Based Neural Network for Session-Aware Recommendation. IEEE Access 2020, 8, 120668-120678, doi: 10.1109/ACCESS.2020.3006360.
12. Sarwar, B.; Karypis, G.; Konstan, J.; Reidl, J. Item-Based Collaborative Filtering Recommendation Algorithms. In Proceedings of the 10th international conference on World Wide Web, 2001; pp. 285-295.
13. Linden, G.; Smith, B.; York, J. Amazon.com Recommendations: Item-To-Item Collaborative Filtering. IEEE Internet Computing 2003, 7, 1, 76-80, doi: 10.1109/MIC.2003.1167344.
14. Luo, X.; Zhou, M.; Li, S.; Shang, M. An Inherently Nonnegative Latent Factor Model for High-Dimensional and Sparse Matrices from Industrial Applications. IEEE Transactions on Industrial Informatics 2018, 14, 5, 2011-2022, doi: 10.1109/TII.2017.2766528.
15. Luo, X.; Zhou, M.; Li, S.; Xia, Y.; You, Z.H.; Zhu, Q.; Leung, H. Incorporation of Efficient Second-Order Solvers Into Latent Factor Models for Accurate Prediction of Missing QoS Data. IEEE Transactions on Cybernetics 2018, 48, 4, 1216-1228, doi: 10.1109/TCYB.2017.2685521.
16. Ruocco, M.; Skrede, O.S.L.; Langseth, H. Inter-Session Modeling for Session-Based Recommendation. In Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems (DLRS 2017), 2017; pp. 24-31, doi: 10.1145/3125486.3125491.
17. Hsueh, S. C.; Shih, M. S.; Lin, M. Y. Context Enhanced Recurrent Neural Network for Session-Aware Recommendation. In Proceedings of the 28th International Conference on Technologies and Applications of Artificial Intelligence, 2023.
18. Barkan, O.; Koenigstein, N. ITEM2VEC: Neural Item Embedding for Collaborative Filtering. In Proceedings of the IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP), 2016; pp. 1-6, doi: 10.1109/MLSP.2016.7738886.
19. Cui, Q.; Wu, S.; Liu, Q.; Zhong, W.; Wang, L. MV-RNN: A Multi-View Recurrent Neural Network for Sequential Recommendation. IEEE Transactions on Knowledge and Data Engineering 2020, 32, 2, 317-331, doi: 10.1109/TKDE.2018.2881260.
20. Steam dataset. Available online: https://cseweb.ucsd.edu/~jmcauley/datasets.html#steam_data (accessed on 2022/10/20).
21. MovieLens dataset. Available online: https://grouplens.org/datasets/movielens/ (accessed on 2022/10/20).
22. Amazon dataset. Available online: http://jmcauley.ucsd.edu/data/amazon (accessed on 2022/10/20).
23. Kang, W.C.; McAuley, J. Self-Attentive Sequential Recommendation. In Proceedings of the IEEE International Conference on Data Mining (ICDM), 2018; pp. 197-206, doi: 10.1109/ICDM.2018.00035.
24. Sun, F.; Liu, J.; Wu, J.; Pei, C.; Lin, X.; Ou, W.; Jiang, P. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management (CIKM '19), 2019; pp. 1441-1450, doi: 10.1145/3357384.3357895.
25. Taylor, W.L. Cloze Procedure: A New Tool for Measuring Readability. Journalism Bulletin 1953, pp. 415-433, doi: 10.1177/107769905303000401.
26. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the North American Association for Computational Linguistics (NAACL), 2019; pp. 4171-4186.