future internet

MDPI

Article

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

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

[1] Department of Information Engineering and Computer Science, Feng Chia University, Taichung 402, Taiwan; linmy@mail.fcu.edu.tw (M.-Y.L.); jun.audis5@gmail.com (P.-C.W.)
[2] Department of Information Management, Chaoyang University of Technology, Taichung 413, Taiwan
* 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 the 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$_{\text{+st+TSA}}$, 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

Citation: Lin, M.-Y.; Wu, P.-C.; Hsueh, S.-C. Optimizing Session-Aware Recommenders: A Deep Dive into GRU-Based Latent Interaction Integration. *Future Internet* **2024**, *16*, 51. https://doi.org/10.3390/fi16020051

Academic Editors: María N. Moreno García and Fernando De la Prieta Pintado

Received: 10 December 2023
Revised: 21 January 2024
Accepted: 31 January 2024
Published: 1 February 2024

1. Introduction

Recommendation mechanisms have emerged as vital tools for the filtering of 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 the 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 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.

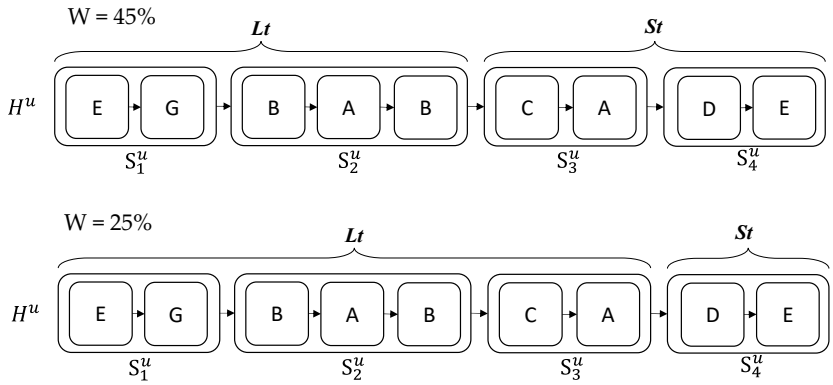
Our work leverages the session-aware recommender system (SARS) over the session-based recommender system (SBRS) to address two critical shortcomings: overlooking contextual information and rigid definitions of long-term and short-term preferences. In our approach, we focus on user–item interactions, such as clicks or purchases, represented by item IDs. These interactions inform our prediction model, which aims to identify the next item a user is likely to engage with. We consider contextual information, divided into intra-context (additional information accompanying interactions) and inter-context (information spanning current and past sessions). One challenge in this method is ensuring the relevance of contextual information to the sequence of interactions. Additionally, another challenge is maintaining the significance of sequential data in our recommendations, considering the complexity of user interactions over time.

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

To address the rigidity of defining long-term and short-term user preferences based on session duration, we propose a flexible window-based approach. This method utilizes a designated percentage of a user's interaction history to categorize preferences. For example, as illustrated in Figure 1, by setting the window (W) at 45%, we consider 45% of the interaction records closest to the most recent interaction as indicative of short-term preferences. The remaining interactions are categorized as long-term preferences. This approach more accurately reflects the user's actual preferences by dynamically adjusting the range between long-term and short-term interactions.

While SBRS effectively captures user preferences within individual sessions, its capacity for personalized recommendations is somewhat limited by its focus on single-session data. This approach may result in less-tailored recommendations due to the absence of long-term preference analysis. Acknowledging this, our research extends to SARS, which encompasses both short-term and long-term user preferences, thus offering a more holistic view of user intent. Our method integrates latent-contextual information, closely linked to interaction records, which is often overlooked in traditional SARS. By considering both

long-term and short-term preferences and employing a window-based approach, we aim to enhance the recommendation performance significantly.



Session: 1 day; *Lt*: long term; *St*: short term; *W*: Window size (ratio of short term)
H^u: User u's interaction history; S_i^u: u's i-th session ($1 \le i \le 4$); A, B, C, …, G: interaction item

Figure 1. Long-term vs. short-term preferences by varying window scope.

In summary, the contributions of this study include as follows:

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.

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 data 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 a 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-contextual information as an auxiliary method in recommendation systems has been explored but not extensively developed.

Our work builds on session-aware recommendation, integrating latent-contextual 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 revised 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-contextual 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 $H^u = \{ S_1^u, S_2^u, \ldots, S_t^u \}$. These records consist of t sessions, each ordered chronologically by the time of interaction. Within each session, denoted as $S_j^u = \left\{ i_1^{u,j}, i_2^{u,j}, \ldots, i_v^{u,j} \right\}$, 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 $\{i_{r1}, i_{r2}, \ldots, i_{rk}\}$ that the user is most likely to interact with next. These predicted items are typically ranked in descending order of interaction probability. Table 1 presents the primary symbols utilized in this paper. It also briefly introduces additional terms such as $R_{i_v^{u,j}}$, $C_{i_v^{u,j}}$, and $F_{i_v^{u,j}}$, which are elaborated upon in the subsequent sections. These terms pertain to the embedding [18] and preference representations crucial to our methodology.

Table 1. Notations.

Notation	Description
$I = \{i_1, i_2, \ldots, i_M\}$	Set of items, i_x is the item ID ($1 \leq x \leq M$)
$U = \{u_1, u_2, \ldots, u_N\}$	Set of users, u_y is the user ID ($1 \leq y \leq N$)
$H^u = \{S_1^u, S_2^u, \ldots, S_k^u\}$	User u's historical interactions, S_j^u is the j-th session ($1 \leq y \leq k$)
$S_j^u = \left\{ i_1^{u,j}, i_2^{u,j}, \ldots, i_v^{u,j} \right\}$	Interaction items in the j-th session, $i_p^{u,j}$ is the p-th item ($1 \leq p \leq v$)
$R_{i_v^{u,j}}$	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.

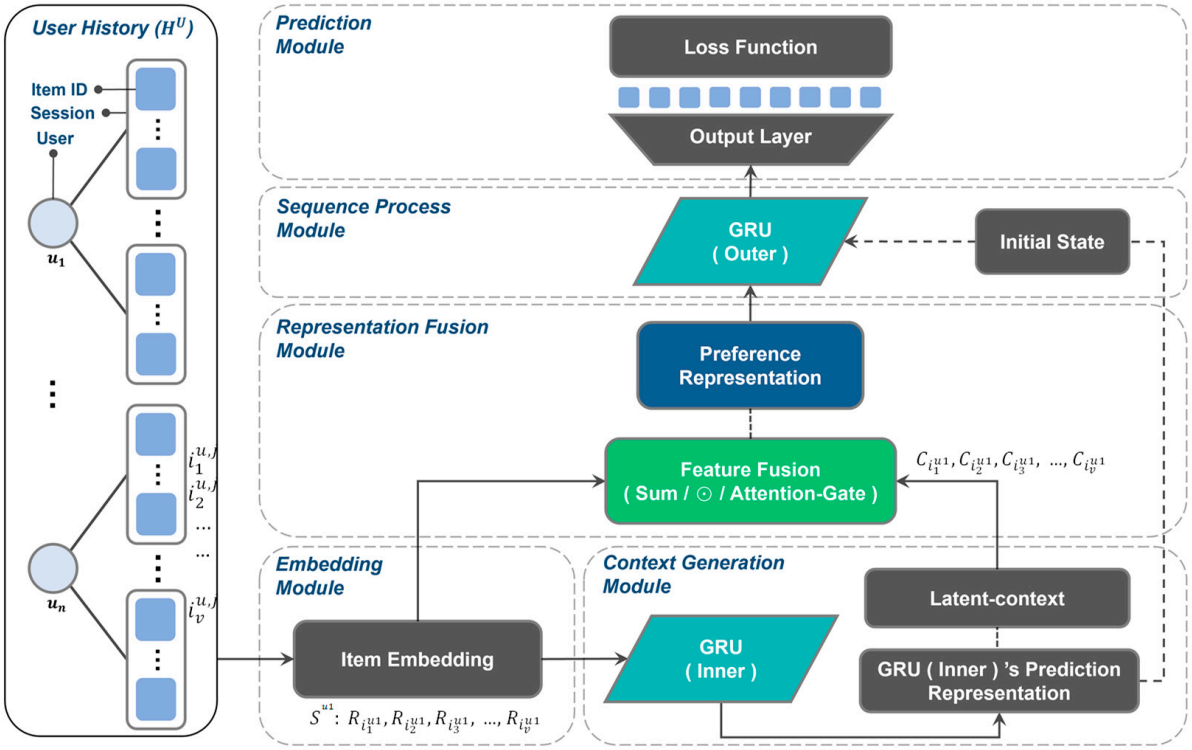


Figure 2. LCII Architecture.

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 contextual information that is considered for recommendations. As the 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 the GRU (Inner) is treated as this potential context information.

The representation fusion module performs feature fusion to generate the preference representation. Feature fusion involves extracting session features by merging item embeddings with latent-contextual representation. The basic LCII performs feature fusion only and uses the result directly as the preference representation. Note that Section 3.2.3 will describe two strategies using additional 'Term Fusion' to generate the preference representation. The first strategy, named LCII-Pre, performs 'Term Fusion' before feature fusion. The second strategy, named LCII-Post, performs 'Term Fusion' after feature fusion. The representation fusion of LCII-Pre is shown in Figure 3, while that of LCII-Post is shown in Figure 4.

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

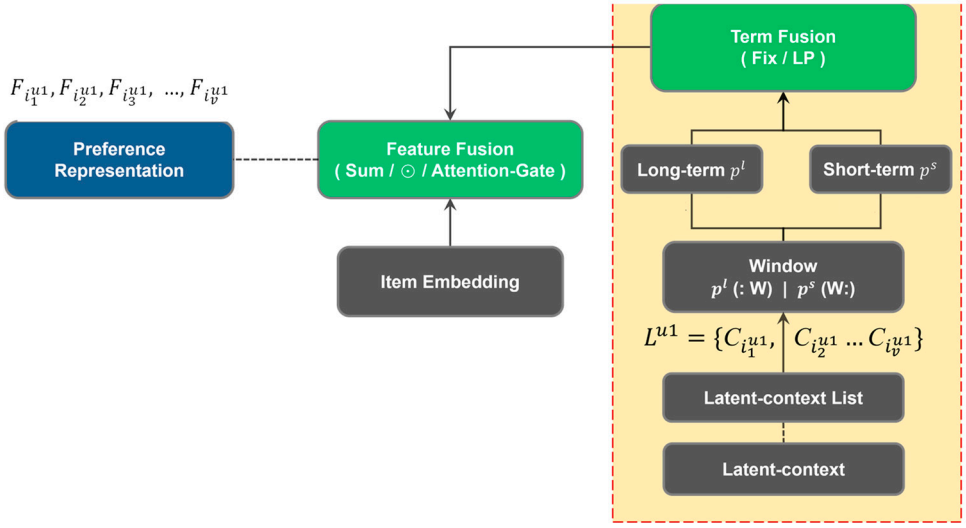


Figure 3. Representation Fusion module—LCII-Pre.

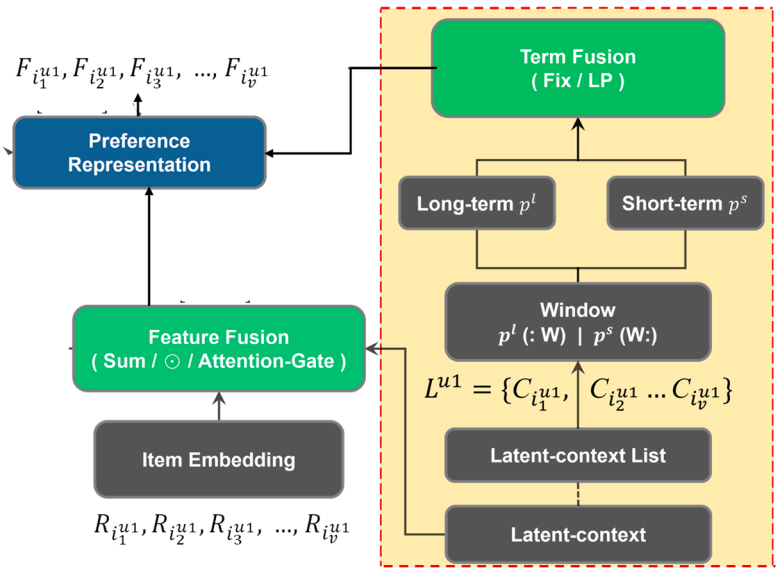


Figure 4. Representation Fusion module—LCII-Post.

3.2.1. Embedding Module

The item embedding module starts by generating a learnable matrix of size M × d, where M is the total number of items and d is the embedding size, using a uniform random number generator. The item IDs are then converted into vectors using this matrix, which is continuously updated during the model's training process. This process involves the integration of context generation, representation fusion, sequence processing, and prediction modules, ensuring that the item embedding is not an isolated process but an integral part of the entire model training. The context generation module then takes these embeddings to extract latent contexts, which are further divided into long-term and short-term preferences in the subsequent modules. This design ensures a seamless and integrated approach to model training, where each module contributes to refining the final output. In Figure 2, this is shown as: a user session S_j^{u1} of v items $i_1^{u1,j}$, $i_2^{u1,j}$, ..., $i_v^{u1,j}$ is embedded as $R_{i_1^{u1,j}}, R_{i_2^{u1,j}}, \ldots, R_{i_v^{u1,j}}$.

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 module 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 $C_{i_1^{u1}}$, $C_{i_2^{u1}}$, $C_{i_3^{u1}}$, …, $C_{i_v^{u1}}$. 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

The item embedding and the latent context produced by the two modules are subsequently utilized as inputs for the representation fusion module. The aim of this module is to combine the two representations to derive a unified user preference representation. As shown in Figure 5(i), feature fusion generates its representation by combining the item embedding (denoted by x_t) and the latent context embedding (denoted by c_t) at the t-th interaction for the whole interaction history, using one of three ways: vector addition, element-wise vector multiplication, and the modified attention-gate mechanism [9]. Vector addition sums corresponding vector elements. Element-wise vector multiplication entails the multiplication of corresponding elements in the vectors. The modified attention-gate mechanism [9] merges elements through multiplication of the weighted item vector denoted by $\alpha_x x$ (where x = (x_1, x_2, …, x_v)) and the weighted (latent) context vector denoted by $\alpha_c c$ (where c = (c_1, c_2, …, c_v)). The attention weights for items and latent context, denoted by α_x and α_c, respectively, are computed using a common relevance-based attention formula.

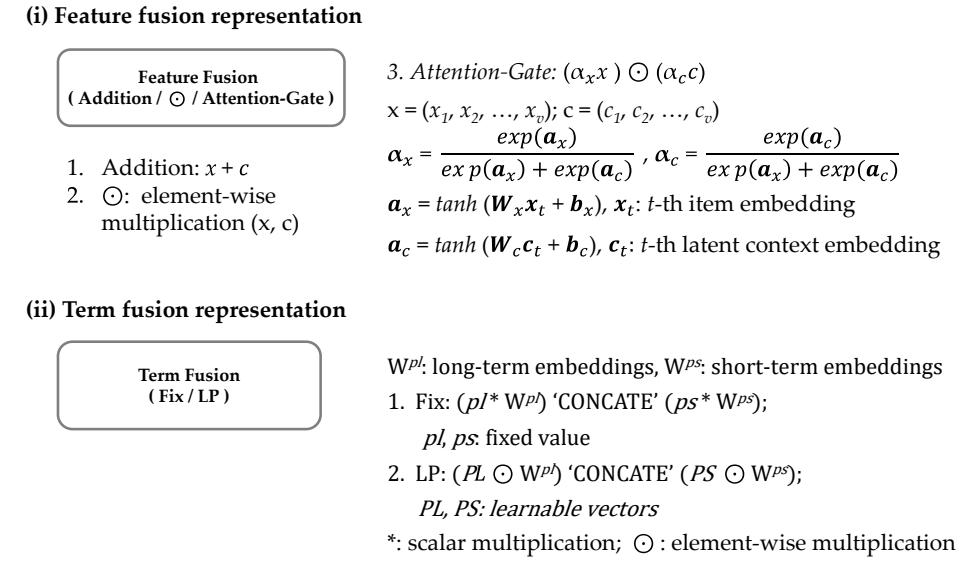


Figure 5. Representation Fusion module—LCII.

In the basic LCII model, the feature fusion representation is directly used as the preference representation. The *LCII-Pre* and *LCII-Post* strategies use additional 'term fusion' to generate the preference representation, as described below.

Figure 5(ii) shows that term fusion is accomplished by either fix fusion or LP (learned-parameters) fusion. First, we determine the window size to separate the long-term and short-term preferences, using the latent-context list as a source. For example, setting the window of 30% for a sequence of 19 items will use the last 30 percent of the latent-context embedding as short-term preference, thus (C_{14}, C_{15}, . . ., C_{19}) as short-term preference and (C_1, C_2, . . ., C_{13}) as long-term preference. Second, the long-term weight and the short-term weight are then decided either by a fixed setting (denoted as Fix) or by learned-parameters (denoted by LP). Let W^{pl} and W^{ps} respectively be the **l**ong-term embedding and **s**hort-term embedding. Fix (term) fusion applies preset values by multiplying W^{pl} with a preset *pl* value and W^{ps} with a preset *ps* value, and then concatenates these multiplied embeddings to form the term fusion representation.

LP (term) fusion dynamically learns the appropriate weights for the embeddings. It begins by initializing the long-term vector *PL* and the short-term vector *PS* with random values. During the training process, these vectors are continuously updated through iterations. The fusion involves element-wise multiplication of *PL* with the long-term embedding W^{pl}, and *PS* with the short-term embedding W^{ps}. After the multiplications, these results are concatenated to create the final term-fusion representation.

Figure 3 illustrates the representation fusion module in *LCII-Pre*. In this configuration, the module incorporates item embedding with the term fusion representation, instead of solely using the latent-context embedding. This is achieved by employing a pre-combining strategy, where the latent-context embedding is processed in the term fusion prior to the feature fusion process.

Figure 4 demonstrates the representation fusion module in *LCII-Post*. Contrasting with *LCII-Pre*, the *LCII-Post* applies a post-combining strategy. This involves processing the latent-context embedding in the term fusion after the feature fusion process. The module operates by first engaging in feature fusion with item embedding, followed by the subsequent incorporation of the latent-context embedding. This sequential integration offers a distinct approach to synthesizing user interaction data into a cohesive preference representation. The preference representation serves as the input to the Outer GRU in the subsequent module, as described next.

3.2.4. Sequence Processing Module

The preference representation, along with the Inner GRU's predictive representation, is input into the Outer GRU for prediction purposes. The embeddings from the preference representation are then sequentially fed into the Outer GRU. The first element of the Inner GRU's predictive representation is used as the initial state for the Outer GRU's hidden state. This approach, detailed in study [16], specifically addresses the cold-start problem in RNNs [10,16,19]. Assigning an initial value to the hidden state of the recurrent neural network allows the model to learn and predict more efficiently [16]. The output from the sequence processing module is subsequently used for prediction.

3.2.5. Prediction Module

The prediction module first derives a user preference representation during the sequence processing phase. This representation undergoes adjustment by a feedforward neural network to align with the format of the original interaction records. The model's training involves a comparison of this representation against actual data (ground truth) using the softmax cross-entropy loss function. This loss function is crucial for training the model, by minimizing the difference between the predicted outputs and actual data. For evaluating the model's accuracy post-training, we employ standard accuracy metrics during the testing phase. These metrics assess the model's performance in ranking items by relevance in actual user sessions. By following this approach, we ensure a clear distinction

between the training phase, where cross-entropy is used for model optimization, and the testing phase, where accuracy is evaluated based on the model's ability to generate relevant item recommendations using the top-k approach.

4. Experimental Results

4.1. Datasets and Pre-Processing

A total of 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 ($60 \times 60 \times 24$ s); 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	9733	1196	6330
Number of items	46,959	3328	4332
Number of actions	700,960	158,498	49,164

4.2. Evaluation Metrics and Parameter Settings

In our experiment, we adopted evaluation metrics such as Recall@K, MRR@K, and NDCG@K. For Recall@K, we considered top-K predictions with K values set at 5, 10, and 20. The experimental settings for each dataset were optimized as follows: For Amazon, the window ratio was 65 with an equal preference ratio of 0.5/0.5 for long-term and short-term preferences, iterating 100 times at a learning rate of 0.001. Steam's settings included a window ratio of four, a preference ratio of 0.2/0.8, 200 iterations, and a learning rate of 0.001. For MovieLens 1M, the window ratio was 30 with a preference ratio of 0.8/0.2, iterating 200 times at a learning rate of 0.01. The attention-gate was used for feature fusion, and a fixed ratio was used for term fusion. Common hyperparameters were an embedding/GRU hidden unit size of 80, one inner/outer GRU layer, a batch size of 100, and a dropout rate of 0.8. We used the Adam optimizer and cross entropy loss function.

The feature fusion method utilizes the attention-gate, and the term fusion method employs a fixed ratio. In our study, we set the embedding/GRU hidden unit size to 80 as a common parameter. This choice was informed by our findings that increasing the embedding size beyond 80 does not significantly improve performance. This approach also considers the training cost of the model. Other hyperparameters, set as common parameters, include 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$_{+ST+TSA}$ [26]: This is the most recent session-aware method, building upon the foundation set by [24], enhanced with ST (session token) and TSA (temporal self-attention) strategies.
- MCLRec [27]: One of the latest studies of session recommendation based on contrastive learning.

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.

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	<u>0.2393</u>	<u>0.2194</u>	<u>0.2043</u>	<u>0.2475</u>	<u>0.2205</u>	<u>0.2224</u>	<u>0.2545</u>	<u>0.2210</u>	<u>0.2270</u>
Attention-Gate	**0.2407**	0.2157	**0.2060**	**0.2487**	0.2168	**0.2244**	**0.2553**	0.2173	**0.2300**

The attention-gate mechanism mitigates conflicts in fusion between data 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.

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, is [0/1].

4.4.4. Overall Comparisons

The overall experimental results are presented in Tables 5–7, respectively, for datasets Amazon, MoveiLens 1M, and Steam. Most-Popular and Item-kNN are not included when their performance is 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$_{\text{Fix}}$ for MovieLens 1M, and LCII-Post$_{\text{Fix}}$ for Steam. In the MovieLens 1M dataset, LCII-Pre$_{\text{Fix}}$ 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$_{\text{LP}}$ 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$_{+\text{ST}+\text{TSA}}$	0.1231	-	0.1060	0.1343	-	0.1096	0.1409	-	0.1113
MCLRec	0.1933	-	0.1601	0.1956	-	0.1597	0.1873	-	0.1534
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$_{+\text{ST}+\text{TSA}}$	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, M.-Y.L.; formal analysis, M.-Y.L.; investigation, M.-Y.L. and P.-C.W.; methodology, M.-Y.L., P.-C.W. and S.-C.H.; supervision, M.-Y.L.; writing—original draft, P.-C.W.; writing—review and editing, S.-C.H. All authors have read and agreed to the published version of the manuscript.

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.

Data Availability Statement: The data presented in this study are available at https://github.com/Junwu0615/LCII-Rec-Model (accessed on 1 December 2023).

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

References

1. Cui, Q.; Wu, S.; Huang, Y.; Wang, L. A Hierarchical Contextual Attention-Based GRU Network for Sequential Recommendation. *arXiv* **2017**, arXiv:1711.05114. Available online: https://arxiv.org/abs/1711.05114 (accessed on 23 September 2022).
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, San Diego, CA, USA, 7–9 May 2015. Available online: https://arxiv.org/abs/1511.06939 (accessed on 20 October 2022).
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), Boston, MA, USA, 15–19 September 2016; pp. 241–248. [CrossRef]
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), Torino, Italy, 22–26 October 2018; pp. 843–853. [CrossRef]

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), Singapore, 6–10 November 2017; pp. 1419–1428. [CrossRef]
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), Amsterdam, The Netherlands, 27 September–1 October 2021; pp. 282–291. [CrossRef]
7. Guo, Y.; Zhang, D.; Ling, Y.; Chen, H. A Joint Neural Network for Session-Aware Recommendation. *IEEE Access* **2020**, *8*, 74205–74215. [CrossRef]
8. Gu, W.; Dong, S.; Zeng, Z. Increasing Recommended Effectiveness with Markov Chains and Purchase Intervals. *Neural Comput. Appl.* **2014**, *25*, 1153–1162. [CrossRef]
9. Phuong, T.M.; Thanh, T.C.; Bach, N.X. Neural Session-Aware Recommendation. *IEEE Access* **2019**, *7*, 86884–86896. [CrossRef]
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), Madrid, Spain, 11–15 July 2022; pp. 2639–2644. [CrossRef]
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. [CrossRef]
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, Hong Kong, China, 1–5 May 2001; pp. 285–295.
13. Linden, G.; Smith, B.; York, J. Amazon.com Recommendations: Item-To-Item Collaborative Filtering. *IEEE Internet Comput.* **2003**, *7*, 76–80. [CrossRef]
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 Trans. Ind. Inform.* **2018**, *14*, 2011–2022. [CrossRef]
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 Trans. Cybern.* **2018**, *48*, 1216–1228. [CrossRef] [PubMed]
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), Como, Italy, 27 August 2017; pp. 24–31. [CrossRef]
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, Yunlin, Taiwan, 1–2 December 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), Vietri sul Mare, Italy, 13–16 September 2016; pp. 1–6. [CrossRef]
19. Cui, Q.; Wu, S.; Liu, Q.; Zhong, W.; Wang, L. MV-RNN: A Multi-View Recurrent Neural Network for Sequential Recommendation. *IEEE Trans. Knowl. Data Eng.* **2020**, *32*, 317–331. [CrossRef]
20. Steam Dataset. Available online: https://cseweb.ucsd.edu/~jmcauley/datasets.html#steam_data (accessed on 20 October 2022).
21. MovieLens Dataset. Available online: https://grouplens.org/datasets/movielens/ (accessed on 20 October 2022).
22. Amazon Dataset. Available online: http://jmcauley.ucsd.edu/data/amazon (accessed on 20 October 2022).
23. Kang, W.C.; McAuley, J. Self-Attentive Sequential Recommendation. In Proceedings of the IEEE International Conference on Data Mining (ICDM), Singapore, 17–20 November 2018; pp. 197–206. [CrossRef]
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), Beijing, China, 3–7 November 2019; pp. 1441–1450. [CrossRef]
25. Taylor, W.L. Cloze Procedure: A New Tool for Measuring Readability. *J. Q.* **1953**, *30*, 415–433. [CrossRef]
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), Minneapolis, MN, USA, 2–7 June 2019; pp. 4171–4186.
27. Qin, X.; Yuan, H.; Zhao, P.; Fang, J.; Zhuang, F.; Liu, G.; Liu, G.; Sheng, V. Meta-optimized Contrastive Learning for Sequential Recommendation. In Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23), Taipei, Taiwan, 23–27 July 2023; pp. 89–98.