Selective Sequence Sampling via (m, k)-Firm Policy for Sequential Recommendation*

Hye-Jin Jeong and Sang-Min Choi†

Gyeongsang National University, Jinju-si, Republic of Korea
{cucumber_love, jerassi}@gnu.ac.kr

Abstract

In sequential recommendation, sliding window-based augmentation methods generate multiple subsequences to enrich training data. However, this process often results in the excessive inclusion of irrelevant user preferences and semantically redundant subsequences, which can increase computational cost. To address these issues, we propose a novel sampling strategy that selects only those subsequences containing a sufficient proportion of meaningful interactions. Our approach adapts the (m, k)-firm deadline concept from process scheduling and extends it to the construction of training data in recommender systems. Experimental results validate the effectiveness of the proposed method and highlight the potential of selective sequence sampling as an explainable method.

1 Introduction

In the modern e-commerce, personalizing the user experience is important to increasing sales and customer engagement [21]. To address this need, recommender systems, a specialized field of information retrieval, analyze a user's past interactions to identify behavioral patterns and suggest items they are likely to prefer. Among these, sequential recommender systems (SRS) make it possible to capture the temporal dynamics of user preferences [13, 16]. With the increasing scale of online platforms and the accumulation of users' historical data, efficient pre-processing has become increasingly important [30, 6]. To enhance the recommendation accuracy of SRS, previous studies have explored various pre-processing techniques, including data augmentation [5], session segmentation [23, 27], and feature enrichment [15, 9].

Among these, we focus on data augmentation and session segmentation, which have been advanced to address the challenges of handling user interactions of varying lengths. Data augmentation techniques [4, 1] aim to compensate for insufficient user information in short sequences, thereby improving the model's learning capability. However, these approaches may inadvertently degrade recommendation performance, as the augmentation process can introduce noise unrelated to user preferences.

On the other hand, long sequences present the challenge of analyzing mixed user preferences within the interactions. To address this, studies have proposed segmenting the original sequence into subsequences [24, 7]. In contrast to the simple truncation of long sequences, a method commonly employed in existing frameworks[1] [2], segmentation enables learning without the loss of user information. However, it is possible to disrupt the temporal continuity of preferences since in this method, each slide can be considered independently for training steps.

*Proceedings of The 2025 IFIP WG 8.4 International Symposium on E-Business Information Systems Evolution (EBISION 2025), Article No. 20, December 16-18, 2025, Sapporo, Japan.

†Corresponding author

[1] https://github.com/RUCAIBox/RecBole

[2] https://github.com/massquantity/LibRecommender

The sliding window approach [12] partitions long sequences into overlapping sub-sequences and generates training data directly from the user's original interactions. This design preserves the temporal coherence of user preferences and prevents noise introduced by synthetic data. Also, it works efficiently for short sequences, as it enriches user preference representations by generating densely overlapped subsequences, thereby augmenting the number of training samples and enabling the model to comprehensively learn behavioral patterns. Nevertheless, the sliding window based sampling approach imposes a significant burden on the system by generating semantically redundant subsequences due to overlapping windows and meaningless subsequences that are not related to user preferences. In particular, the smaller the step size, the more meaningless subsequences are generated. Consequently, the sliding window method, which does not account for the relevance of subsequences to user preferences, leads to potential degradation in recommendation accuracy as well as an increase in computational cost.

Accordingly, previous SRS studies have explored sampling methods for select sequences that are meaningful for model training. AutoSAM [29] introduces an automated sampling framework based on reinforcement learning to non-uniformly sample users' historical interactions. Additionally, LLM4DSR [26] leverages large language models for semantic inference to generate corrected sequences, thereby improving sequential recommendation accuracy. However, these sampling approaches remain stochastic and nondeterministic. As a result, while they can achieve high recommendation accuracy, they lack explainability of the model outputs. Because the sampled sequences vary across different runs, it becomes difficult to establish a stable link between the training data and the model's predictions [19]. Acknowledging the growing importance of explainability alongside accuracy in recommeder systems [31], our sampling strategy departs from existing stochastic methods by explicitly defining and incorporating criteria for meaningful interactions

Our idea was motivated by the (m, k)-firm deadline [8], which was originally introduced in process scheduling. This has a rule that a least m tasks out of k tasks must be completed. To incorporate this into our sampling procedure, we introduces an explicit policy that is our formal definition of meaningful interactions, which guides the selective sampling of high-quality training data. During the sliding process, a subsequence of length k is selected for the training set if it contains at least m meaningful interactions. Through this selective sampling method, we investigated whether we could guarantee both explainability and high accuracy with fewer data samples compared to the traditional sliding window methodology.

Experimental results show that existing sliding window method includes substantial noise, which can hinder model learning. That can be seen from the results showing that randomly removing subsequences from sliding window outputs leads to higher recommendation accuracy than using all subsequences. Moreover, the proposed (m, k)-firm sampling method achieves improved performance compared to the random removing strategy, highlighting the benefits of selective sequence training.

The key contributions in our study are summarized as follows:

- We introduce a novel sampling approach for SRS training sequences by applying the (m, k)-firm deadline, a concept frequently used in scheduling tasks, to select meaningful subsequences.

- By selectively using meaningful sequences, our method reduces computational cost while achieving higher accuracy compared to the original sliding window approach.

- The explicit criterion for sequence selection ensures the explainability of recommendation results.

- Beyond simple data pre-processing, our study illustrates the potential for interdisciplinary research by integrating scheduling principles into the recommendation sampling process.

2 Related Work

2.1 Session Segmentation, Data Augmentation, and Sliding-window

Various SRS studies have explored effective pre-processing of excessively long or short user sequences to achieve both training efficiency and recommendation accuracy [6].

Session segmentation is a method for dividing long user sequences according to specific criteria. By segmenting sequences based on time intervals [22, 14] or user behavior logs [11], models can learn local preferences from the resulting subsequences. Also, data augmentation techniques address the sparsity problem of short sequence data and enable richer representation learning, thereby improving training performance. There are methods to extend sequences by either inserting similar items into user sequences [1] or by repeatedly filling short sequences with the original sequences, rather than zero-padding [4]. In [18], a transformer models is pretrained to predict past items from a users' sequence. Then, the pretrained model generates virtual past items that are inserted at the beginning of the sequence. As a result, the extended sequences allow the model to better capture user preferences, even on short sequences. However, it has the limitation of potentially introducing irrelevant information or distorting the original flow of interaction. A sliding window technique, hybridizing the segmentation and augmentation methods, divides the original sequence into overlapping subsequences [12]. In the next section, we will examine recent studies that have aimed to improve upon the limitations of the sliding window approach.

2.2 Selective Sampling Methods

The sliding window approach can produce a large number of irrelevant or semantically redundant subsequences. This not only increases computational costs but also risks degrading model performance resulting from noise interaction. To overcome these limitations, recent research has focused on selective sampling methodologies. AutoSAM [29] addresses the limitation of sampling methods, which are non-differentiable and thus difficult to optimize with deep learning model. Accordingly, it employs a reinforcement learning-based sampling approach to stochastically select which behaviors to sample from a given user sequence. This sampler does not treat all behaviors uniformly, instead, it prioritizes the selction of important behaviors.

Also, to remove noise interation in sequences degrading recommendation accuracy, a study of [26] utilized an LLM.

The authors introduced an uncertainty estimation module, to enhance reliability of LLM outputs against issues such as non-logical ouputs or hallucinations. This module uses only high-confidence response of LLM to correct the sequences, which are then leveraged for subsequent model training. These approaches have demonstrated the feasibility of achieving high recommendation accuracy with a smaller dataset by selectively sampling data from user sequences.

2.3 Explainability of Recommender Systems

In services where user satisfaction is paramount, such as recommender systems, not only high accuracy but also the explainability of model results has become a crucial factor determining

service quality [31]. However, the sampling methods mentioned in the previous section are stochastic and non-deterministic.

These characteristic leads to lower explainability, a common limitation of deep learning models. They can't provide a clear rationale for how certain behaviors are identified as important or why others are treated as noise [2]. Furthermore, these probabilistic methods introduce a reliability issue due to fluctuating results, as a different sampling of the same data is selected each time [20].

Recent research on Expainable AI (XAI) provides explanations for model outputs by leveraging feature attributions, which indicate the correlation between data points and model outputs [28]. Therefore, it is essential for enhancing explainability to understand how the knowledge derived from specific training samples is incorporated into the model, allowing us to trace back its reasoning. A study by [25] demonstrated that the performance of explainable models diminishes as the level of noise in the dataset increases, highlighting the need for improved data pre-processing to enhance model explainability. In [3], this study enhanced the accuracy and persuasiveness of recommendation explanations by retrieving additional side information from the training dataset in the absence of sufficient user reviews. Also, to address the inconsistency between predicted ratings and generated explanations, [17] utilized an LLM to predict ratings. They then combined the rating vectors with user-item interaction and fed this combined sequence back into the LLM to generate explanations. In this context, we propose a sampling technique that not only enhances recommendation performance and computational efficiency but also guarantees explainability through clear rules.

3 Methods

3.1 Sliding Window Sampling

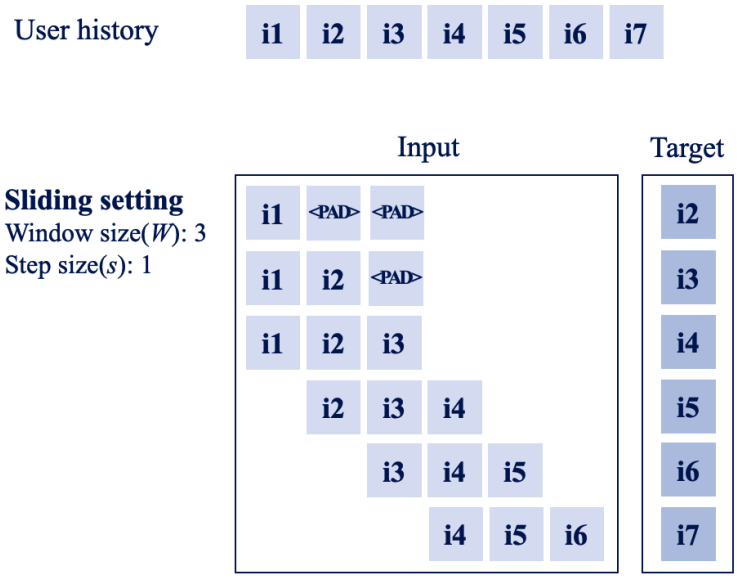

Figure 1: Procedure of sliding window sampling

To construct training data for sequential recommendation, the sliding window strategy is commonly employed on a user's interaction history. The key idea is to generate multiple subsequences from a user's history, enabling the model to effectively learn temporal dependencies.

- **Basic Idea**

  As illustrated in Figure 1, given a user's interaction sequence $S = \langle i_1, i_2, \cdots, i_T \rangle$, a fixed window size W, and a step size of s, this strategy slides a window across the sequence to generate training data. A key aspect of the sliding window method is its ability to ensure the model learns from initial interactions. This is particularly for addressing **cold-start**, where a user's available interaction history is short. To maintain a uniform input length required for batch processing in neural networks, padding is applied to these shorter subsequences using a special $\langle PAD \rangle$ token until they reach length W.

  Specifically, for the example shown in the Figure 1 with a $W = 3$ and $s = 1$:

  - The first input sequence is $\langle i_1 \rangle$. It is padded to become $\langle i_1, PAD, PAD \rangle$, with the target item being i_2.
  - The second input is $\langle i_1, i_2 \rangle$, which is padded to $\langle i_1, i_2, PAD \rangle$, with i_3 as the target.
  - Once the subsequence length reaches the window size W, the window slides one step at a time without padding. For instance, the input $\langle i_1, i_2, i_3 \rangle$ yields the target i_4, and is followed by $\langle i_2, i_3, i_4 \rangle$ with the target i_5, and so on.

  This process is repeated for all positions in the sequence, enabling the model to learn predictive patterns from both short- and long-term user contexts within the historical data.

- **Implementation Pipeline**

  The implementation follows four major steps:

  - **Pre-processing**: All user-item interactions are sorted in ascending chronological order within each user ID.
  - **Window Expansion**: For each user, a starting index *seq_start* is maintained. At each position i, the subsequence $\langle i_{seq\_start}, \cdots, i_{i-1} \rangle$ is constructed. If the subsequence length exceeds a predefined maximum length L, the window slides forward by increasing *seq_start*.
  - **Sample Construction**: Each training consists of:
    - Input Sequence: $\langle i_{seq\_start}, \cdots, i_{i-1} \rangle$
    - Target item: i_i

    The length of each subsequence is also stored, allowing the model to process variable-length inputs.
  - **Batch Transformation**: All generated subsequences are padded to a uniform length and organized into batch tensors for training.

- **Strengths and Limitations**

The sliding window sampling strategy provides rich training signals by exposing the model to subsequences of varying lengths, thereby facilitating the learning of temporal dependencies. However, this method also suffers from inefficiencies: many redundant or irrelated subsequences are generated, leading to unnecessary data expansion and higher computational cost.

3.2 (m, k)-Firm Deadline

- **Basic Idea**

  The (m, k)-firm deadline is a scheduling mechanism commonly used in real-time systems. It imposes the constraint that, within any sequence of k consecutive tasks, at least m task must be completed before their respective deadlines. The following points summarize key characteristics of the (m, k)-firm deadline.

  – While not all tasks are required to meet the deadline, the system's performance and stability are maintained if the proportion of tasks meeting the deadline satisfies m/k.

  – This model balances strict real-time requirements with a degree of flexibility: when $m = k$, all tasks must meet the deadlines(hard real-time), whereas $m < k$ allows occasional deadline misses while maintaining a guaranteed level of QoS.

  In a typical real-time system scenario, multiple request streams(e.g., user sessions, model shards, traffic classes) share one or few server resources. Each stream maintains an internal FIFO(First-In, First-Out) queue, and only the head requests of each queue are candidates for service. The final selection among candidates is determined based on priority, with ties resolved according to a predefined system rule. Depending on the application, requests that have already missed their deadlines may be either dropped or forcibly processed.

- **Distance-Based Priority(DBP)**

  DBP extends (m, k)-firm deadline by providing a mechanism to dynamically evaluate the urgency of individual events within each stream.

  – The recent k outcomes of a stream are represented as a binary state vector(1 = meet, 0 = miss).

  – The priority distance of a stream is defined as the minimum number of consecutive failures required to violate (m, k) constraint. Smaller distances indicate higher urgency and are assigned higher priority.

  – For instance, a (2, 3)-firm stream may have distance value of 0, 1, 2, with smaller values serviced first to prevent dynamic failure transitions. In this situation

  – In a (2, 3)-firm scenario, the sequence $\langle 1, 0, 1\rangle$ and $\langle 0, 1, 1\rangle$ both have the same number of meets within k, but their distances are different. The former has a distance of 1, while the latter has a distance of 2, meaning the former is assigned a higher priority. This is because the sequence $\langle 1, 0, 1\rangle$ is closer to violating the constraint and thus requires more urgent attention to prevent a dynamic failure.

  DBP allows the scheduler to react adaptively to the stream's current state, reducing the dynamic failure probability compared to fixed or uniform priority schemes.

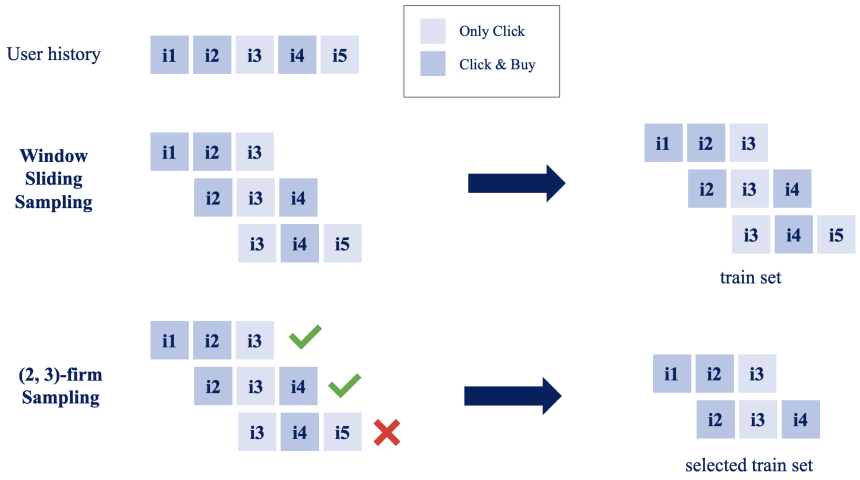

Figure 2: Comparison of Window Sliding Sampling and (m, k)-firm Sampling

3.3 Our method: (m, k)-Firm Sampling

- **Basic Idea**

  We propose (m, k)-firm sampling, a novel sampling strategy for SRS that extends the concept of the (m, k)-firm deadline. This method is designed to tackle a fundamental challenge within user logs. Specifically, these logs are often long and contain a mixture of actions that reflect varying degrees of user intent. For instance, high-commitment actions like a *Buy* event are strongly correlated with genuine intent, while low-commitment actions like simple clicks can be ambiguous.

  Our approach, therefore, is built on the principle of explicitly distinguishing between meaningful and meaningless interactions to improve the signal-to-noise ratio of the training data. The following provides details.

  – Each subsequence generated according to the conventional sliding window method is sampled by applying the (m, k)-firm criterion, ensuring that the subsequence does not violate the constraint.

  – Here, k denotes the maximum length of the generated subsequence, and m represents the minimum number of meaningful interaction (e.g., purchases, high ratings) that each subsequence must include.

  – In the early stage, when the generated subsequence is shorter than k, this policy is not applied. This is to allow the model to learn effectively even from short initial sequences.

  Figure 2 illustrates the sampling process in detail. In the given user history, only interactions i_1, i_2 and i_4 are $Click\&Buy$ events out of total interactions. With a window size of 3 and a step size of 1, the conventional window sliding sampling methods generates three overlapping subsequences. However, our proposed sampling method, in (2, 3)-firm, discards the last subsequence $\langle i_3, i_4, i_5 \rangle$, because it fails to meet the policy's condition of having at least two meaningful interactions within the window. As a result, our sampling

methodology selectively generates training data based on this explicit rule, ensuring that only qualified subsequences are used for training.

- **Formalization**:

  We define S_i as the subsequence generated at time i, as follows:

$$S_i = \begin{cases} \{x_1, \ldots, x_i\}, & \text{if } i \leq K, \\ \{x_{i-K+1}, \ldots, x_i\}, & \text{if } i > K \text{ and } \sum_{j=i-K+1}^{i} f(x_j) \geq M, \\ \varnothing, & \text{otherwise.} \end{cases}$$

  , where x denotes the interaction at time i, K is the window size, and M is the minimum number of meaningful interactions each subsequence must satisfy.

  Here, the indicator function $f(x_j)$ is defined as

$$f(x_j) = \begin{cases} 1, & \text{if interaction } x_j \text{ is meaningful,} \\ 0, & \text{otherwise.} \end{cases}$$

- **Implementation**

  Except for the selection process, the implementation follows the conventional sliding window approach, as described in Section 3.1

4 Experiment

4.1 Dataset

We conducted our experiments on the MovieLens-100K dataset, which consists of user-item interactions annotated with *user_ID*, *item_ID*, *rating*, and *timestamp*. To ensure sufficient learning signals, we filtered out users with fewer than 20 interactions and retained only those with richer activity histories.

4.2 Experimental Setup

As a baseline, we employed a Transformer-based sequential recommendation model implemented using the RecBole [32] framework on top of PyTorch. On this model, we applied both the conventional sliding window methods and our (m, k)-firm sampling approach for comparison.

We adopted the Adam optimizer with a learning rate of 0.0005 and a batch size of 1024. The maximum number of training epochs was set to 300, with a dropout rate of 0.5 applied to mitigate overfitting. To ensure reproducibility, all experiments were run under the same GPU environment with a fixed random seed of 556.

The dataset was split into training, validation, and test sets in a ratio of 0.8/0.1/0.1, preserving the chronological order of interactions per user. Model performance was evaluated using three standard ranking metrics: *Recall@K* and *NDCG@K* [10], which provide complementary perspectives on recommendation accuracy.

4.3 Experimental Result

We conducted two sets of experiments to evaluate the effectiveness of the proposed (m, k)-firm sampling policy. The first experiment focuses on validating the performance gains of the (m, k)-firm policy against baseline sampling strategies, while the second experiment examines how different window sizes with identical failure tolerance rates affect the number of generated subsequences and overall recommendation performance.

4.3.1 Effectiveness of the (m, k)-firm Policy

Table 1: Policy Comparison

	(1) original	(2) random remove	(3) (m,k)-firm sampling
recall@3	0.1652	0.1827 (+10.59%)	**0.1867 (+13.01%)**
recall@5	0.2296	0.2511 (+9.36%)	**0.2546 (+10.88%)**
recall@10	0.3461	0.3696 (+6.78%)	**0.3705 (+7.04%)**
recall@20	0.4866	**0.5015 (+3.06%)**	0.4949 (+1.70%)
NDCG@3	0.3138	0.2919 (-6.97%)	**0.3147 (+0.28%)**
NDCG@5	0.3078	0.2948 (-4.22%)	**0.3137 (+1.91%)**
NDCG@10	0.3249	0.3197 (-1.60%)	**0.3309 (+1.84%)**
NDCG@20	0.3629	0.3578 (-1.40%)	**0.3637 (+0.22%)**
augmented seq	98,344	86,570	86,570

The first experiment is designed to verify the effectiveness of the proposed (m, k)-firm policy. We compared three sampling strategies:

- (1) conventional sliding window sampling
- (2) random drop to match the number of sequences generated by the (m, k)-firm sampling
- (3) the proposed (m, k)-firm sampling

Table 1 shows the overall results of our experiments. The *augmented seq* refers to the total number of sampled sequences, as shown in the last row of Table 1; the numbers in parentheses indicate the relative performance improvement compared to the original. The best-performing values are highlighted in **bold**.

As can be observed from the table, the conventional method yielded the lowest recommendation accuracy. The random drop strategy achieved slightly better results, suggesting that reducing redundancy can help to some extent. In contrast, the(m, k)-firm sampling generally outperformed the baselines, confirming the effectiveness of incorporating temporal reliability constraints into the sampling process.

4.3.2 Impact of Window Size and Failure Rate

Table 2: Tolerance Comparison

	tolerance:0.2		tolerance:0.4	
	(1, 5)	(2, 10)	(2, 5)	(4, 10)
recall@3	**0.1651**	0.1572	**0.1867**	0.1816
recall@5	**0.2307**	0.2304	**0.2546**	0.2514
recall@10	0.349	**0.3554**	**0.3705**	0.3732
recall@20	0.4762	0.4889	0.4949	**0.5093**
NDCG@3	**0.3138**	0.3022	**0.3147**	0.3246
NDCG@5	**0.3105**	0.3027	**0.3137**	0.3193
NDCG@10	0.3244	0.3266	**0.3309**	0.3368
NDCG@20	0.3567	0.362	0.3637	**0.3725**
augmented seq	96,745	98,017	86,570	90,020

The second experiment analyzed the effect of window size under the same failure tolerance ratio. Specifically, we compared the pairs(1, 5) vs. (2, 10) and (2, 5) vs. (4, 10). Although both (2, 5) and (4, 10) share the same failure tolerance of 0.4, the longer window in (4, 10) provides a looser constraint, allowing more subsequences to be retained.

The results, summarized in Table 2, show that the performance differences were not large. However, we observed that (1, 5) generally outperformed (2, 10), while (4, 10) achieved higher accuracy than (2, 5). These findings indicate that system performance is influenced not only by the failure rate itself but also by the structural characteristics of the window, including its length and the resulting number of valid subsequences.

5 Conclusion

In this work, we introduce a novel (m, k)-firm sampling strategy for sequential recommendation, inspired by real-time scheduling theory. We hypothesized that traditional window sliding methods generate a high volume of sequences that include noise and redundancy, potentially hindering model training. Our experiments compared the proposed (m, k)-firm strategy against both the standard window sliding (baseline) and a random sampling approach. The results demonstrated that both sampling methods significantly outperformed the baseline, which used all subsequences. This finding experimentally validates our hypothesis that the high volume of data from standard window sliding contains noise.

Our (m, k)-firm strategy generally outperformed random sampling, there was one specific instance where random sampling achieved slightly higher performance. However, this random approach lacks a systematic or explainable basis for its selection. In contrast, our (m, k)-firm strategy provides an intuitive and interpretable policy to selectively filter subsequences based on the proportion of meaningful interactions. By applying this explicit policy, our method not only reduces computational costs but also introduces explainability at the pre-processing stage.

Although our method is in an early stage of research, it establishes a foundational, interpretable framework for sampling. We believe that this systematic approach has the potential to consistently surpass non-systematic methods like random sampling with future refinement. In the near future, we plan to investigate the applicability of the proposed (m, k)-firm sampling

strategy across a broader range of sequential recommendation models to verify its generalizability. Additionally, by incorporating the Distance-Based Priority (DBP) concept, we aim to design a more dynamic and adaptive sampling process. Although our current experiments focus on relatively small datasets with short sequences, we expect the proposed method to be particularly effective on larger datasets with longer user histories, which we intend to explore in future research.

6 Acknowledgments

This research was supported by the Regional Innovation System & Education(RISE) program through the RISE Center, Gyeongsangnam-do, funded by the Ministry of Education(MOE) and the Gyeongsangnam-do Provincial Government, Republic of Korea.(2025-RISE-16-001)

References

[1] Marcia Barros, André Moitinho, and Francisco M Couto. Seen: Sequential enriched datasets for sequence-aware recommendations. *Scientific data*, 9(1):478, 2022.

[2] Supriyo Chakraborty, Richard Tomsett, Ramya Raghavendra, Daniel Harborne, Moustafa Alzantot, Federico Cerutti, Mani Srivastava, Alun Preece, Simon Julier, Raghuveer M Rao, et al. Interpretability of deep learning models: A survey of results. In *2017 IEEE smartworld, ubiquitous intelligence & computing, advanced & trusted computed, scalable computing & communications, cloud & big data computing, Internet of people and smart city innovation (smartworld/SCALCOM/UIC/ATC/CBDcom/IOP/SCI)*, pages 1–6. IEEE, 2017.

[3] Hao Cheng, Shuo Wang, Wensheng Lu, Wei Zhang, Mingyang Zhou, Kezhong Lu, and Hao Liao. Explainable recommendation with personalized review retrieval and aspect learning. *arXiv preprint arXiv:2306.12657*, 2023.

[4] Yizhou Dang, Yuting Liu, Enneng Yang, Guibing Guo, Linying Jiang, Xingwei Wang, and Jianzhe Zhao. Repeated padding for sequential recommendation. In *Proceedings of the 18th ACM Conference on Recommender Systems*, pages 497–506, 2024.

[5] Yizhou Dang, Enneng Yang, Guibing Guo, Linying Jiang, Xingwei Wang, Xiaoxiao Xu, Qinghui Sun, and Hong Liu. Uniform sequence better: Time interval aware data augmentation for sequential recommendation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 4225–4232, 2023.

[6] Yizhou Dang, Enneng Yang, Yuting Liu, Guibing Guo, Linying Jiang, Jianzhe Zhao, and Xingwei Wang. Data augmentation for sequential recommendation: A survey. *arXiv preprint arXiv:2409.13545*, 2024.

[7] Farzad Eskandanian and Bamshad Mobasher. Modeling the dynamics of user preferences for sequence-aware recommendation using hidden markov models. In *FLAIRS*, pages 425–430, 2019.

[8] Moncef Hamdaoui and Parameswaran Ramanathan. A dynamic priority assignment technique for streams with (m, k)-firm deadlines. *IEEE transactions on Computers*, 44(12):1443–1451, 2002.

[9] Dongsoo Jang, Seok-Kee Lee, and Qinglong Li. Its-rec: A sequential recommendation model using item textual information. *Electronics*, 14(9):1748, 2025.

[10] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[11] Yongzhi Jin, Kazushi Okamoto, Kei Harada, Atsushi Shibata, and Koki Karube. Similarity-based supervised user session segmentation method for behavior logs. *arXiv preprint arXiv:2508.16106*, 2025.

[12] Swanand Joshi, Yesu Feng, Ko-Jen Hsiao, Zhe Zhang, and Sudarshan Lamkhede. Sliding window training-utilizing historical recommender systems data for foundation models. In *Proceedings of the 18th ACM Conference on Recommender Systems*, pages 835–837, 2024.

[13] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pages 197–206. IEEE, 2018.

[14] Minje Kim, Wooseung Kang, Gun-Woo Kim, Chie Hoon Song, Suwon Lee, and Sang-Min Choi. End-to-end time interval-wise segmentation for sequential recommendation. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*, pages 1169–1174, 2025.

[15] Jiacheng Li, Ming Wang, Jin Li, Jinmiao Fu, Xin Shen, Jingbo Shang, and Julian McAuley. Text is all you need: Learning language representations for sequential recommendation. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1258–1267, 2023.

[16] Jiacheng Li, Yujie Wang, and Julian McAuley. Time interval aware self-attention for sequential recommendation. In *Proceedings of the 13th international conference on web search and data mining*, pages 322–330, 2020.

[17] Shijie Liu, Ruixin Ding, Weihai Lu, Jun Wang, Mo Yu, Xiaoming Shi, and Wei Zhang. Coherency improved explainable recommendation via large language model. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 12201–12209, 2025.

[18] Zhiwei Liu, Ziwei Fan, Yu Wang, and Philip S Yu. Augmenting sequential recommendation with pseudo-prior items via reversely pre-training transformer. In *Proceedings of the 44th international ACM SIGIR conference on Research and development in information retrieval*, pages 1608–1612, 2021.

[19] Kun Ma, Cong Xu, Zeyuan Chen, and Wei Zhang. Pattern-wise transparent sequential recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 2025.

[20] Branislav Pecher, Ivan Srba, and Maria Bielikova. On the effects of randomness on stability of learning with limited labelled data: A systematic literature review. *CoRR*, 2023.

[21] J Ben Schafer, Joseph A Konstan, and John Riedl. E-commerce recommendation applications. *Data mining and knowledge discovery*, 5(1):115–153, 2001.

[22] Jinseok Seol, Youngrok Ko, and Sang-Goo Lee. Parameter-efficiently leveraging session information in deep learning based session-aware sequential recommendation. *IEEE Access*, 2025.

[23] Jinseok Jamie Seol, Youngrok Ko, and Sang-goo Lee. 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*, pages 2639–2644, 2022.

[24] Weiqi Shao, Xu Chen, Jiashu Zhao, Long Xia, Jingsen Zhang, and Dawei Yin. Sequential recommendation with user evolving preference decomposition. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, pages 253–263, 2023.

[25] Sairamvinay Vijayaraghavan and Prasant Mohapatra. Stability of explainable recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 947–954, 2023.

[26] Bohao Wang, Feng Liu, Changwang Zhang, Jiawei Chen, Yudi Wu, Sheng Zhou, Xingyu Lou, Jun Wang, Yan Feng, Chun Chen, et al. Llm4dsr: Leveraing large language model for denoising sequential recommendation. *arXiv preprint arXiv:2408.08208*, 2024.

[27] Zihan Wang, Gang Wu, and Yan Wang. Effectively using long and short sessions for multi-session-based recommendations. *arXiv preprint arXiv:2205.04366*, 2022.

[28] Haoyi Xiong, Xuhong Li, Xiaofei Zhang, Jiamin Chen, Xinhao Sun, Yuchen Li, Zeyi Sun, and Mengnan Du. Towards explainable artificial intelligence (xai): A data mining perspective. *arXiv preprint arXiv:2401.04374*, 2024.

[29] Hao Zhang, Mingyue Cheng, Qi Liu, Zhiding Liu, and Enhong Chen. Towards automatic sampling of user behaviors for sequential recommender systems. *arXiv preprint arXiv:2311.00388*, 2023.

[30] Qian-Ming Zhang, An Zeng, and Ming-Sheng Shang. Extracting the information backbone in online system. *PloS one*, 8(5):e62624, 2013.

[31] Yongfeng Zhang, Xu Chen, et al. Explainable recommendation: A survey and new perspectives. *Foundations and Trends® in Information Retrieval*, 14(1):1–101, 2020.

[32] Wayne Xin Zhao, Yupeng Hou, Xingyu Pan, Chen Yang, Zeyu Zhang, Zihan Lin, Jingsen Zhang, Shuqing Bian, Jiakai Tang, Wenqi Sun, Yushuo Chen, Lanling Xu, Gaowei Zhang, Zhen Tian, Changxin Tian, Shanlei Mu, Xinyan Fan, Xu Chen, and Ji-Rong Wen. Recbole 2.0: Towards a more up-to-date recommendation library. In *CIKM*, pages 4722–4726. ACM, 2022.