International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18 (2026) pp. 1-13
© Cerebration Science Publishing

Article

Dynamic User Modeling and Lightweight Knowledge Signals for Candidate Reranking: A Reproducible Two-Stage Study

Ruizhi Xu [1,*]

[1] School of Mathematics, University of Edinburgh, Edinburgh, EH8 9YL, United Kingdom
* Correspondence author: rzxuedu@163.com

Abstract: This study investigates whether introducing dynamic user information and simple knowledge signals can improve the re-ranking effect of the fixed candidate list in the recommendation system when computing resources are limited. To this end, we have constructed a reproducible two-stage recommendation framework: the first stage uses an ID-based collaborative retrieval model to generate the candidate item list, and the second stage utilizes a lightweight model to re-sort these candidates, combined with dynamic user modeling as well as semantic and knowledge-related features. We conducted experiments on the MovieLens-1M dataset and adopted a data setting based on user time sequence, evaluating the sorting ability of the second-stage model on the fixed candidate set. The experimental results show that, compared with the lightweight static ID re-ranking baseline, the dynamic re-ranking model has improved in terms of Recall@10, NDCG@10, and MRR@10 indicators. The ablation results indicate that adding more model components does not necessarily lead to better generalization ability. Some simplified configurations perform better on the test set, and the contribution of dynamic user modeling is positive but relatively limited. Further analysis also reveals that this method still faces significant difficulties in recommending rare items, but the efficiency evaluation shows that the entire process is feasible under moderate hardware conditions. Overall, this paper proposes and validates a two-stage candidate re-ranking framework, and through experimental analysis, demonstrates the role of different components in the fixed candidate recommendation scenario.

Keywords: Recommender Systems, Candidate Reranking, Dynamic User Modeling, Knowledge-Enhanced Learning, Two-Stage Retrieval, Computational Efficiency.

1. Introduction

Recommendation systems have gradually expanded from simple interaction modeling to methods that incorporate semantic information, structured knowledge, and time-based modeling of user behavior [1]. This change essentially stems from a problem: relying solely on collaborative signals is difficult to fully describe user preferences, and information beyond historical IDs often also affects the relevance of the items [2]. Especially when interactions are sparse or unevenly distributed, textual descriptions and structured attributes usually play a supplementary role [3]. Moreover, user interests themselves do change, and models often need to handle both short-term behaviors and long-term preferences simultaneously [4].

The existing methods generally advance in different directions: multimodal methods enhance the representational ability, knowledge enhancement methods introduce graph structures or semantic relationships, and sequence models model temporal changes [5-6]. Each of these directions is valid on its own, but when placed in the same system, if too many modules are stacked, two problems often arise: one is that it is difficult to determine exactly which part contributes to the performance

Copyright: © 2026 by the authors

improvement, and the other is that reproducing it becomes more challenging under limited computing power [7-8]. From the results, a more complex model does not necessarily mean a clearer conclusion.

This article addresses a more specific issue: under conditions of limited computing resources, what roles do dynamic user modeling and lightweight knowledge signals play in fixed candidate reranking, and do different components truly bring about stable generalization improvements. We employ a two-stage framework to conduct this analysis. The first stage generates the candidate set using a purely ID-based collaborative retrieval model; the second stage performs reranking based on the fixed candidates, and the model integrates temporal user modeling, frozen semantic representations, and attribute-level knowledge signals. Once the candidates are fixed, the changes in the second stage will no longer be interfered with by the retrieval stage.

In terms of implementation, we deliberately did not introduce complex structures such as large-scale graph propagation or online multi-hop computations. Instead, we integrated knowledge information into the model in a relatively simple way, such as based on attribute matching features; semantic information was also directly used with the pre-computed embeddings obtained offline, rather than training a large end-to-end model. The main purpose of this approach is to control complexity and make the experiments easier to interpret.

Based on this framework, this research mainly focuses on three issues: whether dynamic user modeling is truly more effective than the static method under a given time setting; which types of features (text, knowledge, gating) are stable and effective, and which are not; and whether these analyses can be completed in a repeatable two-stage process without relying on strong computing power.

We carried out four things: proposed a reproducible two-stage reranking framework, compared dynamic user modeling, semantic representation and lightweight knowledge signals under the same protocol; clarified the experimental setup, including time segmentation, fixed candidate evaluation and ablation experiments; provided a set of analytical evaluation methods, including bucket analysis, ablation, efficiency and multiple random seed results; finally, it was found that under this controlled setting, more complex combinations did not consistently outperform simple models, and this instability itself actually helped to understand the roles of each component.

2. Related Work

2.1. Multimodal Recommendation

Multimodal recommendation typically enhances the recommendation modeling capability by introducing text, images, audio, or other item description information to complement the collaborative signals. Existing studies generally believe that semantic features can lead to performance improvement when interaction data is sparse or when it is difficult to distinguish semantically similar items based solely on collaborative signals. Different methods vary greatly in the combination of multimodal encoding and graph structure propagation, as well as between the downstream recommendation models. Some methods adopt a relatively tight end-to-end design, such as the graph-based multimodal propagation model MGAT [9], while others model based on multimodal knowledge graphs, like MKGCN [10], and further extended M3KGR [11]. These methods usually rely on more complex joint training or propagation mechanisms. In contrast, our method is simpler, with semantic information mainly obtained through offline feature caching and fused lightly in the re-ranking stage, rather than through end-to-end joint optimization.

2.2. Knowledge-Enhanced Recommendation

Knowledge-enhanced recommendation introduces structured information beyond user-item interactions, often in the form of knowledge graphs, item attributes, and semantic relationships, etc. [12]. Existing studies have shown that such information can complement the shortcomings of collaborative embedding in sparse scenarios, and is particularly evident in the cold-start problem [13]. A large number of works in this direction focus on methods such as graph propagation, attention message passing, or high-order relationship modeling to handle information transmission issues in heterogeneous structures. Our work is related to this direction, but does not adopt online multi-hop reasoning or complex graph propagation. Instead, we directly use item-level attribute information and introduce knowledge information into a lightweight ranking model through explicit user-item matching features.

2.3. Sequential and Dynamic Recommendation

Sequential recommendation typically assumes that user preferences change over time, thus requiring explicit modeling of the behavior sequence rather than using static user representations. Early methods were mostly based on recurrent networks, such as GRU4Rec [14], and later developed models based on self-attention, such as SASRec [15] and BERT4Rec [16], as well as graph-structured session modeling methods like SR-GNN [17]. These methods vary in structure, but the basic assumption is the same: the recent behavior sequence often plays a crucial role in the next recommendation [18]. Our approach is also based on this, but instead of using complex sequential models, it splits short-term behaviors and long-term preferences to a certain extent within a lightweight re-ranking framework to maintain the simplicity of user modeling.

2.4. Two-Stage Ranking and Resource-Aware Design

The two-stage recommendation is widely applied in practical systems, and its core idea is to decouple the candidate generation and ranking processes to enhance efficiency and scalability [18]. This design reduces computational costs and brings an additional advantage: when the candidate set is fixed, the differences between different re-ranking models might be attributed to the models themselves rather than the retrieval changes [19]. In recent years, some studies have also begun to focus on lightweight graph models and simplified information dissemination methods to improve efficiency and deployability, such as LightGCN and SocialLGN [20]. Our work follows this approach, based solely on the use of lightweight ID candidate generation, and further studies a richer but still controlled re-ranking design to achieve a balance between expressiveness, computational cost, and reproducibility.

3. Method

3.1. Problem Definition

Let U and I denote the user and item sets. Each interaction is represented as (u,i,t), where $u \in U$, $i \in I$, and t is the timestamp. For each user, we observe an ordered interaction history:

$$H_u = \{(i_1,t_1),\ldots,(i_n,t_n)\} \tag{1}$$

where $t_1 < t_2 < \cdots < t_n$. The goal is to re-arrange the projects that the user might be interested in based on their relevance, so that the more suitable results appear at the top of the list. The entire process is divided into two steps: first, select a batch of candidates from the large-scale candidate space, and then conduct a refined ranking on this fixed number of candidates.

3.2. Overall Framework

Figure 1 presents the overall structure. Stage A is an ID-based collaborative filtering retrieval model that learns the interaction relationships between users and items on the training set, and generates fixed-length candidate lists in the validation and testing stages. Stage B reorders this fixed candidate set and only scores these existing candidates, so the two steps are separated in terms of training and evaluation. The advantage of this design is that it is more controllable in terms of computation and it is also easier to analyze exactly how much improvement the refined model brings to the screening stage.

The refined model does not need to process the entire project library; instead, it only needs to make more detailed preference judgments within the candidate range. In Stage B, the model mainly consists of three parts: the project representation module, the dynamic user modeling module, and a lightweight knowledge path. Finally, the results of these are combined through a simple fusion scoring layer.

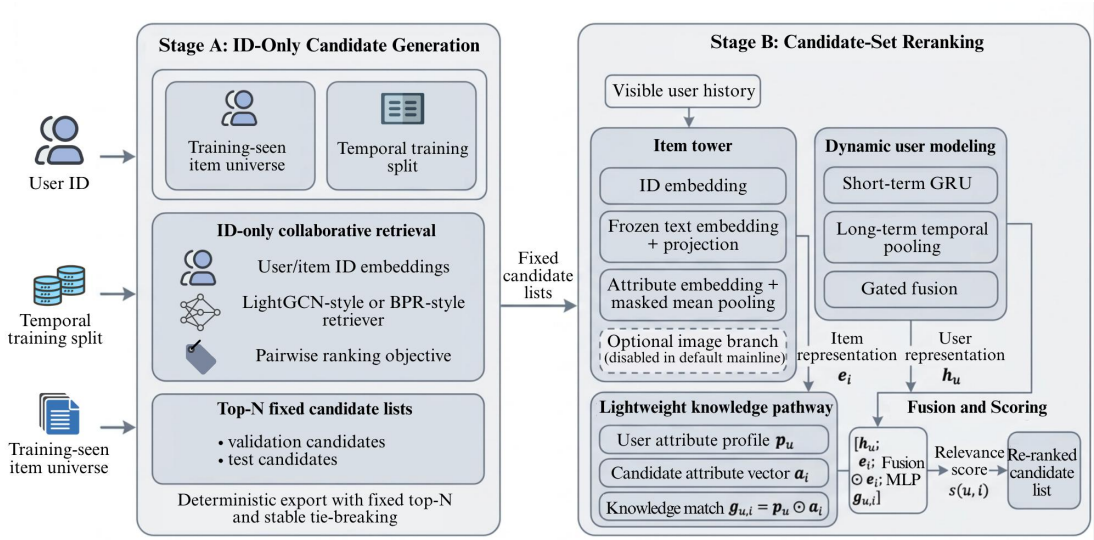

Figure 1. Two-stage framework for fixed-candidate reranking.

Stage A: ID-Only Collaborative Retrieval

Stage A is a retrieval model based on ID-based collaborative filtering. Its main task is to first select a batch of candidates from the entire set of items. The common approach used here is, for example, LightGCN or BPR. Essentially, they both learn the ID vectors of users and items, and then score and rank based on the interaction records.

After the model is trained, the best checkpoint with the best performance will be selected from the validation set for use. Then, a full score is given to each user: first, the items that have already been interacted with in the training set are removed, and then the top- N items with the highest scores are selected from the remaining items and saved as the candidate list. This candidate list will be directly used by Stage B and will not be updated in the middle.

The entire candidate generation process is fixed. Even if there are cases with the same scores, a unified processing rule will be used to fix the order, avoiding different results due to sorting details when running repeatedly.

3.3. Stage B: Candidate-Set Reranking

Stage B reorders the fixed candidate list exported from Stage A. For each sample, the model input includes the user index, the candidate item list, and the visible user history determined by the evaluation protocol. The reorderinger only calculates scores for these candidate items, so this stage deals with the internal sorting of the candidate set rather than recommendations on the complete item set.

In the experiment, all the reordering models used the same Stage A candidate file. The recall method, the number of candidates, and the candidate export process remained consistent. This prevented changes in the candidate generation stage from affecting the comparison results, allowing the differences between the different models to mainly be reflected in the sorting process of Stage B.

3.4. Item Representation

In Stage B, each candidate project is represented by a lightweight item tower. This structure allows for the integration of multiple feature branches. The default configuration includes three sources: a trainable item ID embedding, a frozen text embedding loaded from an offline cache and projected into the reordering space, and an attribute branch based on the index of project attributes. The attribute branch obtains its representation through embedding look-up and is aggregated using masked mean pooling. The code also includes an optional image branch based on offline visual embedding, but it is not enabled in the main experimental configuration. The default model mainly uses ID features, freezes the text features and attribute information. Formally, the candidate item representation is defined as:

$$e_i = e_i^{id} + P_t(e_i^{txt}) + e_i^{attr} \tag{2}$$

where e_i^{id} is the trainable ID embedding, e_i^{txt} is a frozen text embedding obtained offline, $P(\cdot)$ is a trainable projection layer, and e_i^{attr} is the pooled attribute representation. This structure maintains the

lightweight nature of the item tower, while allowing the integration of collaborative signals, semantic information, and attribute features within the same reordering space. If expansion is needed, visual features can also be added, but this part does not belong to the default experimental configuration.

3.5. Dynamic User Modeling

The user representation is constructed from the visible history of item representations produced by the same item tower. Rather than collapsing the entire history into a single average, we explicitly distinguish short-term and long-term preference signals.

The short-term component is formed from the most recent interactions and encoded by a lightweight sequence encoder, implemented as a GRU in the default setting:

$$h_u^{short} = GRU(e_{i_{n-L+1}}, \ldots, e_{i_n}) \tag{3}$$

where $e_{i_{n-L+1}}, \ldots, e_{i_n}$ denote the item representations of the most recent L interactions in the visible history.

The long-term component summarizes older history through temporally weighted pooling:

$$h_u^{long} = \frac{\sum_{k=1}^{n} \exp(-\beta \Delta t_k) e_{i_k}}{\sum_{k=1}^{n} \exp(-\beta \Delta t_k)} \tag{4}$$

where Δt_k is the elapsed time between interaction k and the current prediction point, and β controls the decay rate.

The final user representation is gained through gated fusion:

$$\alpha_u = \sigma(W_g [h_u^{short}; h_u^{long}] + b_g), h_u = \alpha_u \odot h_u^{short} + (1 - \alpha_u) \odot h_u^{long} \tag{5}$$

where $\sigma(\cdot)$ is the sigmoid function, $[;]$ denotes vector concatenation, and $\odot$ denotes element-wise multiplication. If the gate is disabled in an ablation setting, the model falls back to a fixed 0.5/0.5 mixture. This short-term/long-term structure gives the user model a clear interpretation: one branch captures recent preference drift, while the other preserves more persistent taste.

3.6. Lightweight Knowledge Injection

The design of the knowledge components remains in a lightweight structure. During the reordering stage, online graph traversal, neighbor propagation, or multi-hop graph reasoning are not performed. Instead, knowledge information is incorporated into the model through two simple methods. First, attribute features are embedded and directly incorporated into the item representation. Second, based on the attribute statistics in the user history and the attribute distribution of candidate items, a compact user-item matching vector is calculated and used as an additional rating signal.

Let $a_i \in 0,1^m$ denote the attribute indicator vector of candidate item i, where m is the number of attribute types. Let:

$$p_u = \frac{1}{|H_u|} \sum_{j \in H_u} a_j \tag{6}$$

denote the user-side historical attribute profile estimated from the visible history. The lightweight knowledge match feature for user u and candidate item i is then defined as:

$$g_{u,i} = p_u \odot a_i \tag{7}$$

which captures the overlap between the candidate's attribute profile and the user's historical attribute distribution. In the current mainline implementation, these attributes mainly correspond to the information at the type level. This design provides a structured preference alignment signal, which can serve as a supplement to the collaborative features and semantic representations. The specific effects still depend on the experimental evaluation results, rather than being pre-determined in the model design stage.

3.7. Fusion and Scoring

The final relevance score is produced by a compact fusion MLP. Its input consists of the user representation, the candidate item representation, their element-wise interaction, and the optional knowledge match vector:

$$s(u,i) = MLP([h_u; e_i; h_u \odot e_i; g_{u,i}]) \tag{8}$$

The scoring structure also maintains a relatively simple form. The model does not employ cross-modal Transformers or large generative scoring modules. The relatively compact structure not only ensures controllability of the training process but also facilitates the observation of the impact of different signal paths in ablation experiments independently.

3.8. Training Objective

The reranker is trained with a pairwise BPR objective over training interactions, where negative items are sampled from the training item universe. For a user u, a positive item i^+, and a sampled negative item i^-, the ranking loss is defined as:

$$L_{BPR} = -\sum_{(u,i^+,i^-)} \log \sigma\left(s\left(u,i^+\right)\right) - \left(s\left(u,i^-\right)\right) \tag{9}$$

Besides, the training loop may include an alignment regularizer that encourages projected user representations and projected positive-item representations to remain close in cosine space:

$$L_{align} = \sum_{(u,i^+)} \left(1 - \cos\left(p_u\left(h_u\right), p_i\left(e_{i^+}\right)\right)\right) \tag{10}$$

where $p_u(\cdot)$ and $p_i(\cdot)$ are trainable projection layers. The overall objective is:

$$L = L_{BPR} + \lambda L_{align} \tag{11}$$

where λ controls the strength of the alignment term. Here, L_{align} acts as a regularization term for stabilizing representation learning rather than as a teacher–student distillation loss.

4. Experimental Setup

4.1. Dataset and Preprocessing

The experiment was conducted on the MovieLens-1M dataset. In the preprocessing stage, the original interactions were mapped to continuous user and item indices to ensure the stability of subsequent modeling. To better align with the sequential recommendation setting rather than random preference modeling, we sorted each user's interaction records by time and divided them into training, validation, and test sets in an 80/10/10 ratio. This division method preserves the chronological order of behavior occurrence and avoids information leakage that may occur when randomly shuffling.

In terms of project-side features, we completed all offline preparations before the rerankinging stage. The text features were constructed based on the original directory metadata and encoded into frozen vectors, which were then cached for direct invocation during the reranking process. Attribute information (such as word-level attributes and item statistical features) was also completed in the offline stage. If image information is introduced in the auxiliary experiments, the image features will also be extracted and cached in advance, but they are not included as part of the main input in the default settings, thus not considered as the core experimental configuration.

For the convenience of subsequent analysis, we also constructed a set of diagnostic buckets based on the statistical information of the training set. These buckets are generated solely using the training data and do not incorporate information from the validation set or the test set, thus maintaining the consistency of the evaluation.

4.2. Evaluation Protocol

The assessment process is divided into two stages to distinguish the contributions of candidate generation from those of reranking.

The Stage A is handled by the recall model, which assigns scores to the entire item set. The model only uses the interaction information visible during the training phase. Under this setting, only when the target item in the validation or test sample appears in the assessable set, will this sample be included in the calculation. For each valid sample, we perform a full-library ranking while excluding the positive

examples from the user's training set, and record the position of the target item in the ranking list. Therefore, the indicators of this stage essentially reflect the full-library retrieval performance under the transfer setting.

The Stage B only conducts sorting evaluation on the candidate set generated in the Stage A. In the experiment, this candidate set was fixed as the Top-100 results. Only when the samples simultaneously meet the "evaluable" condition and the target items exist in the candidate set, will they participate in the final indicator statistics; otherwise, the remaining samples will be excluded. Therefore, the indicators of this stage reflect the re-ranking ability on the candidate set rather than the overall library sorting ability. The results of the two stages are always presented separately in the report to avoid confusing the contributions of different modules.

In terms of evaluation metrics, the standard Top-K indicators are used for unified measurement, including Recall@K, NDCG@K, and MRR@K, which are employed to reflect the ranking quality under single-target related settings. Since the two stages correspond to different ranking spaces respectively, this split evaluation not only ensures fairness but also helps to explain the actual gains brought by different modules.

4.3. Baselines

The overall comparison setting mainly focuses on the current two-stage process.

In the Stage A, we employed a pure ID collaborative filtering model based on LightGCN as the recaller. The main function of this module is to generate a stable and fully covered candidate set, rather than pursuing more complex semantic modeling capabilities.

In the Stage B, we compared two types of reranking methods. The first type is the static ID reranking model, which uses trainable user and item embeddings and scores through dot product. The training process adopts a BPR-style objective function. This method serves as a lightweight baseline, relying solely on ID information for sorting optimization. The second type is the dynamic reranking model proposed in this paper. This model combines dynamic modeling of user behaviors, item tower structure, and a fusion scoring mechanism, and can optionally include an alignment regularization term. Under the default configuration, the model uses item ID representation, frozen text features, and attribute-level knowledge signals, while disabling the image branch to ensure consistency and controllability of the experimental setup.

4.4. Implementation Details

Model selection is entirely accomplished based on the performance of the validation set. During the reranking stage, we use the candidate results on the validation set for early stopping judgment, and only retain the checkpoints that perform the best on the validation set. A one-time evaluation is conducted on the fixed test candidate set. The test set results do not participate in any form of model selection or parameter tuning process. The model selection in the recall stage follows the same processing method.

In the default dynamic reranking settings, the model training employs the AdamW optimizer with a learning rate of 0.001, weight decay of 1e-4, batch size of 256, and a training limit of 30 epochs. We use the validation set's NDCG@10 as the early stopping metric and set the patience to 5. All experiments default to setting the random seed to 42. The training samples are constructed based on the chronological order of user behavior prefixes, and the optimization objective is the pairwise BPR loss. In some settings, an alignment regularization term is also added. The static ID reranking is trained using its own independent optimization configuration, rather than sharing the same set of training budgets or hyperparameter settings with the dynamic model.

Throughout the entire experimental process, a crucial approach is to ensure that all comparisons in the Stage B are based on the same set of candidate results from the Stage A, including the size of the candidate set and the exported files being exactly the same. This is done primarily to avoid additional fluctuations caused by candidate differences, thereby making the comparison in the reranking stage clearer. Unless otherwise specified, the results reported in this paper are all from the experimental run with a single random seed of 42.

4.5. Reporting

In addition to the top-*K* indicators in the main table, we also presented several additional types of auxiliary analysis results. The first type is bucket-based evaluation, which is used to observe whether the model improvement is concentrated on specific interaction patterns or is relatively consistent across different user groups. The second type is efficiency-related indicators extracted from training logs and run statistics, which are used to reflect the performance of the entire system in terms of resource consumption, which is also consistent with the lightweight setting emphasized in this paper. The third

type is the aggregation of results through multiple sub-statistical summaries when there are multiple repeated experiments, including mean, standard deviation, and corresponding significance tests. However, such statistics are only provided when multiple runs have actually been conducted.

5. Results and Analysis

5.1. Overall Ranking Performance

Table 1 reports the reranking results of Stage-B on the test set. It should be noted that since Stage-B is evaluated based on the fixed candidate set generated by Stage A, the values of Recall@10, NDCG@10, and MRR@10 actually represent the sorting performance within the candidate set, rather than the overall directory sorting results. Under this evaluation setting, the dynamic reranking configuration is overall superior to the static ID baseline. In terms of specific values, Recall@10 has increased from 0.1507 to 0.1694, NDCG@10 has risen from 0.0727 to 0.0794, and MRR@10 has improved from 0.0494 to 0.0527.

This result indicates that under fixed candidate conditions, introducing more comprehensive Stage-B representations can to some extent improve the sorting quality, and is more effective than relying solely on ID-based scoring methods. Meanwhile, Table 1 also shows that there are certain fluctuations among different configurations. The complete model does not achieve the best results in all indicators. Some ablation settings perform comparably or even slightly better than it. This phenomenon indicates that the improvement in performance does not solely come from the accumulation of structural complexity, but rather there is a certain non-uniform contribution among different components.

Therefore, the results of this section are more appropriately interpreted as a comparative observation of the behaviors of different modeling components within the two-stage framework: Dynamic reranking has generally improved over the static baseline, but the introduction of additional components does not necessarily lead to a stable increase in performance.

Table 1. Main Stage-B test results

Method	Recall@10	NDCG@10	MRR@10	ΔNDCG@10 vs. Static
Static ID baseline	0.1507	0.0727	0.0494	0.0000
Main dynamic MMKG	0.1694	0.0794	0.0527	+0.0068
w/o dynamic	0.1633	0.0784	0.0530	+0.0057
w/o gate	0.1765	0.0835	0.0558	+0.0109
w/o KG	0.1817	0.0855	0.0568	+0.0128
w/o text	0.1765	0.0831	0.0553	+0.0105

5.2. Ablation Results and Component Behavior

The results of the ablation experiments show that there is no monotonous improvement relationship of "the more components, the better" for the overall structure. As shown in Table 2, after removing the dynamic user module, Recall@10 and NDCG@10 slightly decreased, but MRR@10 showed a slight increase, indicating that this module has a more stable effect on the overall sorting consistency, but its contribution to the top position hit of some samples is limited. When removing the gating mechanism, knowledge branch, or text branch, the model's performance on the test set was better than the complete configuration. Among them, the result of removing the KG was the most prominent: Recall@10 = 0.1817, NDCG@10 = 0.0855, MRR@10 = 0.0568, and NDCG@10 improved by 0.0061 compared to the complete model. Figure 2 shows a comparison of the results of each variant. It can be seen that the impact of different modules is not consistent. Some components are more likely to introduce noise rather than bring stable gains under the current data scale and training settings. The contribution of each module shows conditional dependence rather than a fixed positive superposition effect.

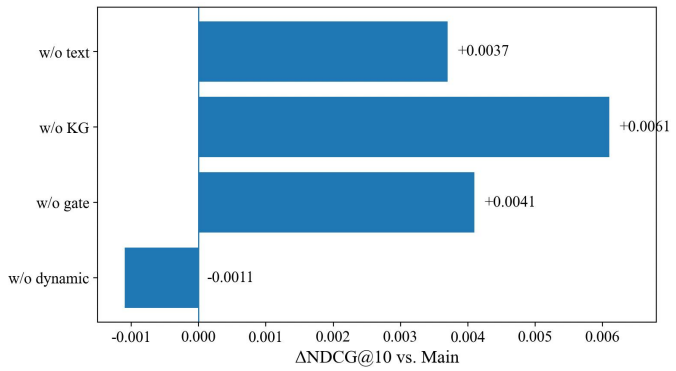

Figure 2. Ablation Results Relative to the Main Model.

These results do not support the strong assumption that "each branch in the complete architecture consistently improves performance". They expose the unstable combination between dynamic modeling, text semantic features, lightweight knowledge information, and gating mechanisms. Some interactions may introduce additional fluctuations in the fixed candidate ranking setting. That is to say, the increase in model complexity does not necessarily lead to better temporal generalization performance. A more reasonable interpretation is that the significance of this experimental setup lies in allowing us to observe the true effects of different components under unified candidate constraints, thereby distinguishing between effective contributions and redundant designs.

Overall, in the ablation experiments, dynamic user modeling still brings a small but consistent improvement compared to the more basic representation methods. However, in the complete configuration, the knowledge branches, text branches, and gating mechanisms did not show a stable test gain. Some settings even experienced a performance decline. These phenomena suggest that the effectiveness of the reranking module cannot be inferred solely based on structural complexity or design intuition, but rather needs to be verified item by item through ablation experiments to determine its actual contribution.

Table 2. Ablation Results Relative to the Full Model.

Variant	Recall@10	NDCG@10	MRR@10	ΔNDCG@10 vs. Main
Main dynamic MMKG	0.1694	0.0794	0.0527	0.0000
w/o dynamic	0.1633	0.0784	0.0530	-0.0011
w/o gate	0.1765	0.0835	0.0558	+0.0041
w/o KG	0.1817	0.0855	0.0568	+0.0061
w/o text	0.1765	0.0831	0.0553	+0.0037

5.3. Bucket-Based Analysis

To better understand the differences in the model's performance, we further conducted a group analysis of the main configurations based on two dimensions: user history length and project frequency. Table 3 presents the group results on the sample subsets where the sorting results are available and the grouping is valid. Figure 3 shows the trend comparisons for different groupings.

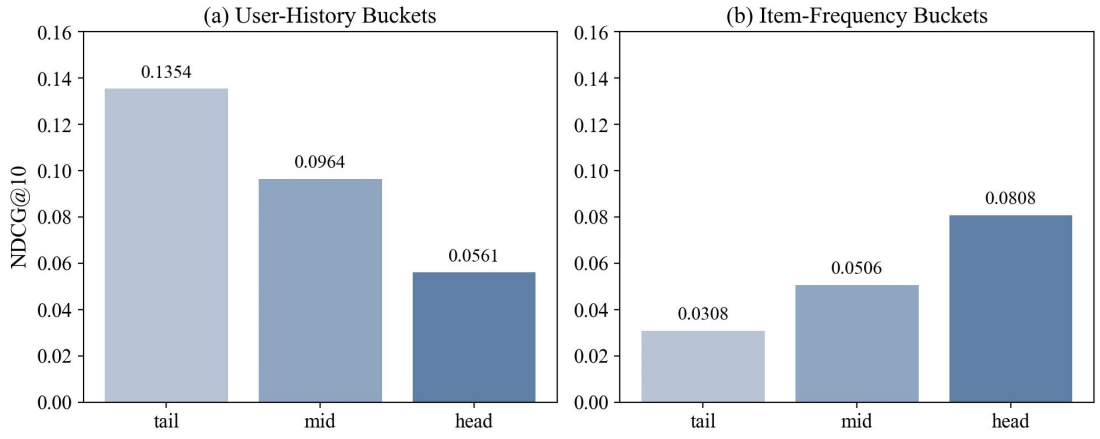

Figure 3. Bucket-level NDCG@10 of the main dynamic configuration across (a) user-history buckets and (b) item-frequency buckets.

In terms of the length of user history, there are significant differences in performance among different groups. It is noteworthy that the user groups with shorter or medium-length histories achieved better results, while the user group with the longest history performed the worst, with a Recall@10 of 0.1241 and an NDCG@10 of 0.0561; in contrast, the best-performing user group achieved a Recall@10 of 0.2775 and an NDCG@10 of 0.1354. This phenomenon proves that under the current time modeling and ranking settings, an increase in user history does not necessarily bring ranking advantages. A longer history may introduce more behavioral noise or preference drift.

In terms of the project frequency dimension, the model's performance on long-tail projects is significantly weaker. The Recall@10 of the tail frequency bucket is only 0.0741, and the NDCG@10 is 0.0308, which is significantly lower than that of the head frequency bucket. This indicates that the ranking of long-tail projects remains one of the main difficulties in the current task. However, it should be noted that the sample size of the tail bucket is relatively small, so this result is more suitable as a trend reference rather than a strict statistical conclusion.

The group results show that the average indicators cannot fully reflect the true differences of the model across different data subsets. The model demonstrates significantly higher stability in high-frequency items, and the relationship between the user's historical length and the sorting difficulty is not a simple linear one; instead, it is jointly influenced by the data distribution and the complexity of behaviors.

Table 3. Bucket-level test results for the main dynamic configuration.

Bucket axis	Bucket	n	Recall@10	NDCG@10	MRR@10
overall	all	23914	0.1694	0.0794	0.0527
user history	tail	3770	0.2775	0.1354	0.0930
user history	mid	6405	0.2028	0.0964	0.0648
user history	head	13739	0.1241	0.0561	0.0360
item frequency	tail	54	0.0741	0.0308	0.0179
item frequency	mid	1016	0.1230	0.0506	0.0293
item frequency	head	22844	0.1716	0.0808	0.0538

5.4. Efficiency and Practical Feasibility

The original design intention of this framework was to operate within limited computing resources. However, the current efficiency statistics still show that more complex reranking structures do incur certain training and computational costs. Table 4 summarizes the relevant training information: The LightGCN retriever in Stage-A was trained for a total of 34 epochs. The best result occurred at the 24th epoch, with a total training time of 3097.8 seconds, averaging approximately 93.9 seconds per epoch. In contrast, the main reranking configuration was trained for 30 epochs, reaching the optimal result at the 26th epoch, with a parameter size of approximately 607K. However, there are no finer-grained records of time delay and memory usage.

These results indicate that this process can complete end-to-end training under the current resource conditions. However, the introduction of the reranking stage is not "zero-cost", and its actual cost still exists and has not been fully quantified. By combining the aforementioned ablation results, it can be seen that when a simplified model performs no worse or even better in terms of performance, the additional structure not only needs to prove the benefits brought by its representational ability, but also needs to simultaneously prove that its computational and training costs are reasonable and necessary.

Table 4. Recorded training summary for the reported runs.

Run	Total epochs	Best epoch	Param count	Total training time (s)	Avg. time per epoch (s)
Recall (LightGCN)	34	24	90,021,120	3097.8	93.9
Rerank (main dynamic MMKG)	30	26	607,009	-	-

5.5. Discussion and Final Interpretation

The experimental results show a relatively consistent but not completely one-sided trend. On one hand, the main dynamic reranking configuration is overall superior to the lightweight static ID baseline, showing that in the two-stage framework with fixed candidates, introducing a more complex B-stage

reranking does indeed cause a certain performance improvement. However, the complete model did not achieve the best results in all settings, and some simplified versions (such as removing the knowledge branch, text branch, or gating mechanism) performed better instead.

Therefore, in this paper, it is not appropriate to simply interpret each module as "all bringing positive gains". A more appropriate statement would be that this framework is more like an experimental platform for observing the differences in the behavior of components in candidate reranking. Dynamic user modeling is somewhat helpful overall, but the improvement is limited; knowledge, text, and gated-related branches have not shown stable marginal benefits under the current training and evaluation settings; meanwhile, the sorting ability of long-tail projects remains one of the main bottlenecks.

From the overall results, some more direct patterns can be observed: Within the two-stage framework with fixed candidates, introducing a stronger reranking model is indeed generally superior to the static ID method, but this improvement is not achieved by the cumulative effect of all modules. Some components do not exhibit stable performance under different data distributions, and sometimes even lower the overall effect. At the same time, the model's performance on long-tail projects is consistently weak, indicating that the current method is still mainly constrained by data sparsity and distribution imbalance. Overall, these phenomena suggest that performance changes may be closely related to data characteristics and the stability during the training process.

Therefore, the ablation analysis in this paper is more inclined towards empirical observations at the component behavior level, rather than a direct confirmation of the effectiveness of each module. Its main significance is to distinguish which designs truly bring generalization benefits under the shared evaluation protocol and which designs increase structural complexity without establishing a sustained advantage.

5.6. Dynamic User Modeling Ablation

The dynamic user ablation experiment mainly compares the complete model with the version without time modeling to see if dynamic user modeling is useful. In the complete model, user representation is composed of short-term sequences, long-term aggregation, and gated fusion; while in the ablation version, these time-related structures are directly removed and replaced by a static approach based on historical behavior.

From the results, after removing the dynamic user module, Recall@10 and NDCG@10 decreased slightly, but MRR@10 was slightly higher instead. Overall, this module is helpful, but the improvement is not significant and it does not consistently outperform in all metrics.

5.7. Text Signal Ablation

Text fusion involves revoking the text branches on the project side, removing the part of the representation obtained from the pre-trained text encoding in the candidate projects, while keeping the rest of the structure unchanged.

This was mainly done to see if the semantic information of the text provided any additional assistance. From the results, it can be seen that after removing the text branches, the actual performance improved slightly. It proves that this fixed text path never consistently bring gains t under the current training. More precisely, it's not that the text information is useless, but rather it has not been well transformed into signals that are helpful for the ranking in this model.

5.8. Knowledge Signal Ablation

Knowledge dissipation involves removing the attribute information part of the project, as well as the user-project knowledge matching features used during scoring. The two-stage reranking framework is still retained. From the results, it can be seen that after removing the knowledge branch, the test performance was actually better. This indicates that this relatively lightweight knowledge module does not consistently bring additional benefits in the complete model. However, this result cannot be directly interpreted as "knowledge is useless". A more reasonable interpretation is that this part of the attribute information may have some interfering effects under the current implementation method, or it does not cooperate well with other branches. A more appropriate design or training method is needed to truly bring its effect.

5.9. Gate Ablation

Gate-based ablation means turning off the adaptive fusion between short-term and long-term user representations. It simply uses a fixed 0.5/0.5 for a simple average.

From the results, it can be seen that this fixed average performs slightly better than the version with learning gating. Perhaps it is because the gating mechanism adds an extra layer of learnable parameters, making the training more unstable, or introducing some fluctuations, but it does not result in better generalization performance. Therefore, the gating mechanism is more like an "optional feature".

5.10. Limitations and Robustness Considerations

The previous analysis has already explained some quite clear phenomena: there are indeed significant differences between the historical lengths and project frequencies of different users, especially in the case of long-tail projects where the performance is relatively weak. However, it should be noted that the sample size of the tail buckets is very small, so these results are more suitable for observing trends rather than being regarded as strict statistical conclusions.

From an efficiency perspective, the overall process can be executed under the current hardware conditions. However, the more complex reranking stage does indeed incur additional costs, and the existing logs do not cover complete latency and memory information. Therefore, these efficiency results are more indicative of "feasibility" and are not sufficient to support a comprehensive deployment cost assessment.

Another limitation is that the current results are mostly derived from a single experimental run. Although the process itself supports multiple repeated experiments, no systematic comparison of various sub-components has been conducted yet. Therefore, it is uncertain whether the component ordering remains stable under different random seeds. Thus, these conclusions are more inclined to be a preliminary empirical observation and cannot be regarded as a final conclusion at present.

6. Conclusion

This paper proposes and implements a two-stage candidate reranking framework under the condition of limited computing resources. The Stage A employs a lightweight collaborative retrieval model based solely on IDs to generate a fixed candidate set. The Stage B performs reranking on this candidate set by combining dynamic user modeling, frozen semantic features, and attribute-level knowledge signals.

Under the fixed candidate setting, the dynamic reranking configuration is overall superior to the static ID baseline. The results show that introducing a more comprehensive representation at this stage can improve the quality of the candidate set ranking. The ablation experiment results show that after removing the knowledge branches, text branches, or adaptive gating structure, the model's performance did not decline in some settings, and even improved slightly; the benefits brought by the dynamic user module are relatively limited but relatively stable. These results are closer to reflecting the differences in the effects between components rather than a unified cumulative benefit model. In the further analysis, the bucket results revealed significant differences among various user behavior patterns. Particularly, the performance on long-tail items remained relatively weak. The efficiency statistics indicated that, under the current resource constraints, the two-stage process could operate as a whole, but the more complex reranking design would incur additional computational costs.

The experiments in this paper were mainly conducted under a single-subject setting, and the stability of multiple subjects has not been systematically verified. The evaluation in stage B relies on a fixed candidate set, so the results mainly reflect the sorting ability within the candidate set rather than the performance across the entire search space.

References

1. Li, Y., Liu, K., Satapathy, R., Wang, S., & Cambria, E. (2024). Recent developments in recommender systems: A survey. IEEE Computational Intelligence Magazine, 19(2), 78-95.
2. Pan, L., Pan, Z., Cai, F., & Chen, H. (2025). Multimodal Recommender Systems: A Survey of Representation, Modeling, and Optimization. Information Fusion, 103991.
3. Ganhör, C., Moscati, M., Hausberger, A., Nawaz, S., & Schedl, M. (2024, October). A multimodal single-branch embedding network for recommendation in cold-start and missing modality scenarios. In Proceedings of the 18th ACM conference on recommender systems (pp. 380-390).
4. Liu, B., Li, D., Wang, J., Wang, Z., Li, B., & Zeng, C. (2024). Integrating user short-term intentions and long-term preferences in heterogeneous hypergraph networks for sequential recommendation. Information Processing & Management, 61(3), 103680.

5. Bellini, V., Di Sciascio, E., Donini, F. M., Pomo, C., Ragone, A., & Schiavone, A. (2024). A qualitative analysis of knowledge graphs in recommendation scenarios through semantics-aware autoencoders. Journal of Intelligent Information Systems, 62(3), 787-807.
6. Spillo, G., Musto, C., De Gemmis, M., Lops, P., & Semeraro, G. (2024). Recommender systems based on neuro-symbolic knowledge graph embeddings encoding first-order logic rules. User Modeling and User-Adapted Interaction, 34(5), 2039-2083.
7. Hu, H., Guo, W., Liu, X., Liu, Y., Tang, R., Zhang, R., & Kan, M. Y. (2024, March). User behavior enriched temporal knowledge graphs for sequential recommendation. In Proceedings of the 17th ACM International Conference on Web Search and Data Mining (pp. 266-275).
8. Tahmasebi, S., Nikzad, N., Payberah, A. H., Asgari-Chenaghlu, M., & Matskin, M. (2025, April). Fact vs. Fiction: Are the Reportedly "Magical" LLM-Based Recommenders Reproducible?. In European Conference on Information Retrieval (pp. 79-94). Cham: Springer Nature Switzerland.
9. Tao, Z., Wei, Y., Wang, X., He, X., Huang, X., & Chua, T. S. (2020). MGAT: Multimodal graph attention network for recommendation. Information Processing & Management, 57(5), 102277.
10. Cui, X., Qu, X., Li, D., Yang, Y., Li, Y., & Zhang, X. (2023). MKGCN: MultiModal knowledge graph convolutional network for music recommender systems. Electronics, 12(12), 2688.
11. Wei, Z., Wang, K., Li, F., & Ma, Y. (2024). M3KGR: A momentum contrastive multi-modal knowledge graph learning framework for recommendation. Information Sciences, 676, 120812.
12. Wang, H., Zhao, M., Xie, X., Li, W., & Guo, M. (2019). Knowledge graph convolutional networks for recommender systems. In Proceedings of the 2019 World Wide Web Conference (pp. 3307–3313).
13. Wang, X., He, X., Cao, Y., Liu, M., & Chua, T. S. (2019). KGAT: Knowledge graph attention network for recommendation. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (pp. 950–958).
14. Hidasi, B., Karatzoglou, A., Baltrunas, L., & Tikk, D. (2016). Session-based recommendations with recurrent neural networks. In International Conference on Learning Representations (ICLR).
15. Kang, W. C., & McAuley, J. (2018). Self-attentive sequential recommendation. In 2018 IEEE International Conference on Data Mining (ICDM) (pp. 197–206).
16. Sun, F., Liu, J., Wu, J., Pei, C., Lin, X., Ou, W., & Jiang, P. (2019). BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management (pp. 1441–1450).
17. Wu, S., Tang, Y., Zhu, Y., Wang, L., Xie, X., & Tan, T. (2019). Session-based recommendation with graph neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence, 33, 346–353.
18. Covington, P., Adams, J., & Sargin, E. (2016). Deep neural networks for YouTube recommendations. In Proceedings of the 10th ACM Conference on Recommender Systems (pp. 191–198).
19. He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., & Wang, M. (2020). LightGCN: Simplifying and powering graph convolution network for recommendation. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (pp. 639–648).
20. Liao, J., Zhou, W., Luo, F., Wen, J., Gao, M., Li, X., & Zeng, J. (2022). SocialLGN: Light graph convolution network for social recommendation. Information Sciences, 589, 595–607.