Enhancing Time Awareness in Generative Recommendation

Anonymous ACL submission

Abstract

Generative recommendation has emerged as a promising paradigm that formulates the recommendations into a text-to-text generation task, harnessing the vast knowledge of large language models. However, existing studies focus on considering the sequential order of items and neglect to handle the *temporal dynamics* across items, which can imply evolving user preferences. To address this limitation, we propose a novel model, ***G**enerative **R**ecommender **U**sing **T**ime awareness (**GRUT**)*, effectively capturing hidden user preferences via various temporal signals. We first introduce *Time-aware Prompting*, consisting of two key contexts. The user-level temporal context models personalized temporal patterns across timestamps and time intervals, while the item-level transition context provides transition patterns across users. We also devise *Trend-aware Inference*, a training-free method that enhances rankings by incorporating trend information about items with generation likelihood. Extensive experiments demonstrate that GRUT outperforms state-of-the-art models, with gains of up to 30.0% and 24.8% in Recall@5 and NDCG@5 across four benchmark datasets. The code will be available upon acceptance.

1 Introduction

Generative recommendation (GR) is an emerging paradigm that redefines the traditional recommendation task as a text-to-text generation problem (Rajput et al., 2023; Geng et al., 2022). While the conventional discriminative approach ranks items individually (Kang and McAuley, 2018), GR directly generates the identifier (ID) of the target item given a user history. Notably, it benefits from directly leveraging the extensive capabilities of large language models (LLMs) for recommendations (Raffel et al., 2020; Touvron et al., 2023).

Despite its success, existing GR models overlook a crucial dimension: *temporal dynamics*. As illustrated in Figure 1, the temporal information of items significantly affects user preferences (Li et al., 2020; Zhang et al., 2024). Without temporal information, the model might recommend another '*stuffed animal*' based on frequent occurrences in the user history, even after preference has shifted toward the '*LEGO product*' over one year (Figure 1a). In contrast, a time-aware GR model can accurately discern preference shifts and recommend products that match the user's current interests by considering temporal dynamics (Figure 1b). Moreover, timestamps may imply seasonal preferences that the mere item order cannot capture, *e.g.*, Christmas or holidays (Wang et al., 2020a).

Figure 1: Illustration of our motivation. While (a) existing generative recommenders only consider sequential order, (b) our method utilizes temporal information.

Incorporating temporal information into recommendations yields several challenges. (i) Temporal signals exist in distinct forms: *absolute timestamps* and *relative time intervals* across user interactions. Each signal provides different signals, making it challenging to preserve their information while effectively combining them (Cho et al., 2020; Zhang et al., 2025). (ii) Temporal item patterns vary in scope from individual user behavior to collective item-level trends and transition patterns. The collective patterns further require analyzing the interaction of all users. While previous work has adopted graph-based methods (Zhang et al., 2024; Wang et al., 2020b), representing temporal knowledge in natural language form for GR remains unex-

plored. (iii) Integrating temporal signals into GR requires unique modeling. Unlike traditional sequential models that rely on explicit temporal embeddings (Li et al., 2020; Zhang et al., 2025; Hu et al., 2025) or contrastive learning (Tian et al., 2022; Dang et al., 2023; Zhang et al., 2024), it is crucial to translate complex temporal patterns into natural language. Concurrently, GR models are required to maintain the ability to generate precise item IDs from the vast item candidate pool.

To address these challenges, we propose a novel model, ***G**enerative **R**ecommender **U**sing **T**ime awareness (**GRUT**)*, enhancing GR through temporal signals of items. To model distinct temporal signals, we first introduce *Time-aware Prompting*, which consists of two contexts. At the user level, we integrate absolute timestamps and time intervals between interactions in the prompt to model individual user patterns. At the item level, item transition patterns are represented in natural language forms, incorporating broader temporal patterns that individual user history alone cannot provide. Besides, we devise *Trend-aware Inference*, a flexible method that refines beam search ranking with the temporal trend of items. It adaptively combines item generation likelihoods with trend scores, assigning higher scores to recently trending items. Despite its simplicity, it enables the model to reflect diverse and timely recommendation scenarios.

Our key contributions are summarized as follows: (i) To the best of our knowledge, this is the first work to integrate temporal dynamics into GR, demonstrating its importance beyond the mere sequential order of items. (ii) We propose *Time-aware Prompting*, which effectively incorporates multi-dimensional temporal patterns at both user and item levels. (iii) We design *Trend-aware Inference*, which adaptively leverages trends to refine recommendation rankings without model retraining. (iv) Extensive experiments demonstrate that GRUT outperforms state-of-the-art models, with improvements of up to 30.0% in Recall@5 and 24.8% in NDCG@5 across four benchmark datasets.

2 Related Work

We categorize sequential recommendation[1] into two dimensions, as shown in Table 1: temporal information utilization and whether they employ generative approaches (Li et al., 2024).

[1]For more details on existing sequential recommendation models, see Appendix A.1.

	Discriminative	Generative
Sequential info.	GRU4Rec, HGN, SASRec, BERT4Rec, FDSA, S^3Rec	P5, TIGER, LC-Rec, LETTER, IDGenRec, TransRec, ELMRec
Temporal info.	TiSASRec, TGSRec, TCPSRec, TiCoSeRec, TGCL4SR, HM4SR, HORAE	**GRUT (Ours)**

Table 1: Category of existing sequential recommendation models. GRUT introduces a time-aware generative recommendation model.

2.1 Generative Recommendation

It directly generates the target item identifier from user history as a text-to-text generation task[2]. P5 (Geng et al., 2022; Hua et al., 2023) first pioneered this paradigm with multi-task learning. Recent works have largely focused on item identifiers. TIGER (Rajput et al., 2023), LC-Rec (Zheng et al., 2024), and LETTER (Wang et al., 2024a) use vector quantization (Zeghidour et al., 2022) for codebook-based identifiers. LC-Rec further aligns language and collaborative semantics with codebook IDs, and LETTER integrates collaborative signals into identifiers. Meanwhile, IDGenRec (Tan et al., 2024) generates keyword IDs from textual metadata, and TransRec (Lin et al., 2024) combines multiple identifier types. More recently, ELMRec (Wang et al., 2024b) injects graph-based high-order interaction knowledge. However, the temporal dynamics of items remain unexplored in GR, which struggles to grasp shifting user preferences over time.

2.2 Temporal Recommendation

Temporal information in recommendations implies how user preferences evolve, providing richer information than the sequential order of items in the sequence. TiSASRec (Li et al., 2020) initiated the use of time intervals with self-attention (Kang and McAuley, 2018), while TGSRec (Fan et al., 2021) incorporates timestamp embeddings. Several models leverage contrastive learning with temporal information: TCPSRec (Tian et al., 2022) employs temporal contrastive pre-training, TiCoSeRec (Dang et al., 2023) develops time-aware sequence augmentation methods, and TGCL4SR (Zhang et al., 2024) constructs temporal item transition graphs for graph-based

[2]We mainly focus on improving the GR model, aiming to generate target item identifiers. See Appendix A.2 for further LLM-based recommendation models.

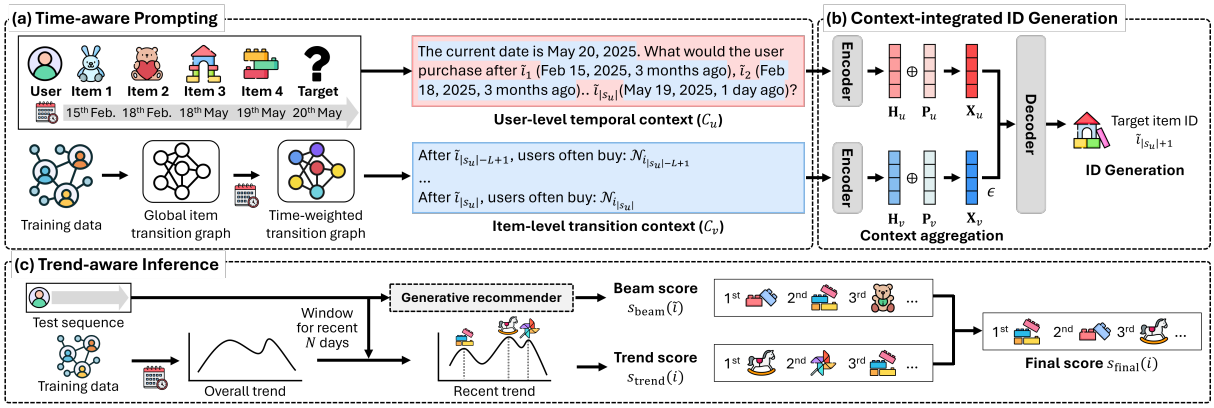

Figure 2: Overall architecture of GRUT. The core innovation of the model is (a) *Time-aware Prompting* that captures evolving user preferences. This is followed by (b) *Context-integrated ID Generation* that aggregates contexts and complemented by (c) *Trend-aware Inference* that adaptively incorporates the trend of items.

contrastive learning. Recent work extends temporal dynamics to non-neural models (Park et al., 2025).

Importantly, for multi-modal sequential recommendation, HM4SR (Zhang et al., 2025) encodes timestamps into embeddings and combines them with item ID, text, and image representations through a mixture of experts. HORAE (Hu et al., 2025) enhances a multi-interest pre-training model by incorporating temporal context with texts. However, both models only extract representations from LLMs without fine-tuning, limiting their ability to fully harness the capabilities of LLMs. Recent work has also explored temporal awareness for LLMs in sequential recommendation (Chu et al., 2024). However, it only evaluates on sampled candidates rather than the entire set, limiting its scalability.

3 Background

Let $\mathcal{U}$ and $\mathcal{I}$ denote the sets of users and items, respectively. Each user $u \in \mathcal{U}$ has an interaction history represented as a sequence $s_u = (i_1, \ldots, i_{|s_u|})$, where each interaction corresponds to actions such as purchasing or clicking. Each element i_j represents the item the user interacted with at the j-th position, and $|s_u|$ indicates the number of items in s_u. The timestamp sequence is denoted by $T_u = (t_1, \ldots, t_{|s_u|})$, indicating the temporal information corresponding to s_u. Sequential recommendation aims to predict the next item $i_{|s_u|+1} \in \mathcal{I}$ that the user is most likely to interact with.

For generative recommendation, each item $i \in \mathcal{I}$ is assigned a unique ID $\tilde{i}$. Generally, item IDs can be represented as a sequence of codebook tokens (Rajput et al., 2023; Zheng et al., 2024) or short text (Tan et al., 2024). With the item ID, the user sequence is converted to the sequence of item IDs $\tilde{s}_u = (\tilde{i}_1, \tilde{i}_2, \ldots, \tilde{i}_{|s_u|})$. Inspired by (Tan et al., 2024), we extract keywords from item descriptions using term frequency (Jones, 2004) to create descriptive item IDs.[3] In existing studies (Geng et al., 2022; Tan et al., 2024), the user sequence is represented without temporal information:

> What would the user purchase after $\tilde{i}_1$, $\tilde{i}_2$, $\ldots$, $\tilde{i}_{|s_u|}$?

Here, the goal is to generate the target item ID $\tilde{i}_{|s_u|+1}$, which the user is most likely to prefer.

4 Proposed Model

We present a ***G**enerative **R**ecommender **U**sing **T**ime awareness (**GRUT**)*, which enhances GR via explicit modeling of temporal dynamics. The overall architecture is depicted in Figure 2. Our primary contribution is *Time-aware Prompting* that effectively captures temporal patterns from both individual user behavior and collective user transitions (Section 4.1). These patterns are then utilized in *Context-integrated ID Generation* (Section 4.2). After training, we design *Trend-aware Inference*, which refines rankings by incorporating generation likelihood with temporal trends (Section 4.3).

4.1 Time-aware Prompting

We introduce time-aware prompting that harnesses temporal dynamics by incorporating user-level temporal context and item-level transition patterns. It models individual temporal patterns based on absolute timestamps and relative intervals while leveraging collective transition patterns across users.

[3] See Appendix B.3.1 for details of keyword extraction.

4.1.1 User-level Temporal Context

The user-specific temporal patterns are encoded into natural language form, leveraging the capabilities of LLMs to process temporal information through prompting (Xiong et al., 2024). We enhance the basic input of GR by injecting the temporal information of interactions.

Specifically, we utilize two distinct forms of temporal signals: *target-relative intervals* and *absolute timestamps*. (i) The target-relative interval Δt_i (*i.e.*, the interval between each timestamp t_i and the inference timestamp $t_{|s_u|+1}$) effectively reflects how user preferences may have shifted over time. For instance, recent interests can be highlighted when recommending shortly after an interaction, while stable long-term preferences are emphasized for longer intervals. (ii) Absolute timestamps t_i enable the model to recognize seasonal patterns or cyclical behaviors, *e.g.*, holiday shopping, that intervals alone cannot capture.

The user-level temporal context C_u is as follows:

> The current date is $t_{|s_u|+1}$.
> What would the user purchase after
> $\tilde{i}_1$ (t_1, Δt_1 ago), $\tilde{i}_2$ (t_2, Δt_2 ago), $\cdots$,
> $\tilde{i}_{|s_u|}$ ($t_{|s_u|}$, $\Delta t_{|s_u|}$ ago) ?

where $\tilde{i}$ represents item IDs that can take various forms, *e.g.*, keywords or titles. Note that our method can be generally applied regardless of the item ID representations, as shown in Appendix C.2.

This context enables learning complex patterns beyond the sequential orders of items, grasping preference shifts according to time intervals. Notably, it has three key advantages: (i) The target-relative intervals and absolute timestamps provide complementary signals that consistently outperform either when used alone, as shown in Table 3. (ii) Owing to the target-relative interval, it is particularly effective for long time intervals, where user preferences have likely evolved, as demonstrated in Figure 3. (iii) By considering current dates, recommendations can be dynamically adapted based on inference timestamps, unlike existing GR models that make identical predictions regardless of inference timestamps. It is reported in Appendix D.1.

4.1.2 Item-level Transition Context

We leverage item-level transition patterns to capture common consumption behaviors across all users, identifying what items users typically consume next after specific items. While user-level temporal context focuses on individual preferences over time, it cannot model collective patterns across users. The item transition pattern has been widely recognized as crucial information in recommendations (Zhang et al., 2024; Wang et al., 2020b). Apart from previous studies, we convert these structural patterns into natural language formats for GR.

Global Item Transition Graph. We first construct a global item transition graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ from all training sequences. Here, the node set $\mathcal{V}$ represents all items, and the edge set $\mathcal{E}$ represents transitions between items. For each user sequence s_u, we extract all item pairs $(i_t, i_{t'})$ where $t < t'$ and record the time interval $\Delta t_{i,j} = t_j - t_i$. We add all pairs as directed edges to the graph, where each edge $e_{i,j} \in \mathcal{E}$ denotes a transition from item i to item j, along with the corresponding time interval $\Delta t_{i,j}$.

Time-weighted Transition Graph. For a given item $i \in \mathcal{I}$, we calculate transition probabilities for all outgoing edges $\{e_{i,j} | j \in \mathcal{I}\}$ from the graph, considering time intervals (Park et al., 2025). We assign a time-decaying weight that gives higher importance to shorter time intervals:

$$w(\Delta t_{i,j}) = \max\left(\exp\left(-\frac{|\Delta t_{i,j}|}{\tau}\right), c\right), \quad (1)$$

where τ controls decay speed and c ensures minimum weight for long-term transitions. Using time-aware weights, the transition probability $p_{i,j}$ is formulated as:

$$p_{i,j} = \frac{\sum_{(i,j)\in\mathcal{A}_{i,j}} w(\Delta t_{i,j})}{\sum_{j'\in\mathcal{I}} \sum_{(i,j')\in\mathcal{A}_{i,j'}} w(\Delta t_{i,j'})}, \quad (2)$$

where $\mathcal{A}_{i,j}$ denotes the set of all pairs from item i to item j in the training data.

Based on the transition probability, we extract meaningful patterns by selecting the top-k neighboring items for each of the last L items in the sequence:

$$\mathcal{N}_i = \{\tilde{i}^1, ..., \tilde{i}^k\} = \underset{j\in\mathcal{I}}{\text{Top-}k}\, p_{i,j}, \quad (3)$$

where $\mathcal{N}_i$ represents the set of top-k neighboring item IDs for item i. Here, k and L are hyperparameters. We focus on the last L items in the sequence, considering the recency and the maximum input sequence length of language models. These top-k items refer to the items that users most frequently purchased after the given item, based on the collective behavior patterns across all users.

Prompt Transformation. The extracted transition patterns are then transformed into natural language using item IDs. The item-level transition context C_v is expressed as:

> After $\tilde{i}_{|s_u|-L+1}$, users often buy: $\mathcal{N}_{i_{|s_u|-L+1}}$.
> ...
> After $\tilde{i}_{|s_u|}$, users often buy: $\mathcal{N}_{i_{|s_u|}}$.

where the item $\tilde{i}_{|s_u|-L+n}$ is the n-th item among the last L items, and $\mathcal{N}_{i_{|s_u|-L+n}}$ is represented by concatenating all item IDs within the set. This context can integrate the item transition knowledge with the recommendation process, in addition to the user-specific temporal context.

4.2 Context-integrated ID Generation

After extracting user-level temporal patterns and item-level transition knowledge, we aggregate the two contexts to generate accurate target item IDs that reflect evolving user preferences.

Context Aggregation. We employ a well-established parallel encoding approach (Izacard and Grave, 2021; Yen et al., 2024). It consists of two key steps: (i) encoding each context independently and (ii) aggregating contexts in the decoder through cross-attention. First, the user-level temporal context C_u and the item-level transition context C_v are separately encoded with a shared encoder:

$$\mathbf{H}_c = \text{Encoder}\,(C_c) \in \mathbb{R}^{M\times d}, \quad c \in \{u, v\} \quad (4)$$

where M represents the number of text tokens, and d is the hidden dimension size. To further distinguish context types, learnable context-type embeddings are added to encoder outputs:

$$\mathbf{X}_c = \mathbf{H}_c + \mathbf{P}_c \in \mathbb{R}^{M\times d}, \quad c \in \{u, v\} \quad (5)$$

where $\mathbf{P}_u$ and $\mathbf{P}_v$ are unique embeddings for user-level and item-level contexts, respectively. We then combine all representations into a unified embedding matrix $\mathbf{X}$:

$$\mathbf{X} = [\mathbf{X}_u; \epsilon \cdot \mathbf{X}_v] \in \mathbb{R}^{(2\times M)\times d}, \quad (6)$$

where ϵ is a hyperparameter that controls the effect of transition patterns without overwhelming user-specific signals. Finally, the decoder processes the unified information via cross-attention, where $\mathbf{X}$ serves as the key-value matrix.

Training Objective. Once contexts are aggregated, the decoder autoregressively generates the target item ID $\tilde{i}$. The model is trained by minimizing the sequence-to-sequence cross-entropy loss with teacher forcing:

$$\mathcal{L} = -\sum_{t=1}^{|\tilde{i}|} \log P(\tilde{i}^t | C_u, C_v, \tilde{i}^{<t}), \quad (7)$$

where $\tilde{i}^t$ is a t-th token of $\tilde{i}$, and $\tilde{i}^{<t}$ denotes the sequence of tokens generated before the t-th token.

4.3 Trend-aware Inference

We design trend-aware inference to incorporate real-time item trends at recommendation time $t_{|s_u|+1}$ into the final ranking. This training-free method adjusts predictions to reflect dynamic patterns, such as rapidly trending items that emerged after training. This ensures timely recommendations without requiring model retraining, as further demonstrated in Appendix D.2.

Beam Score. The beam score for an item ID $\tilde{i}$ is defined as:

$$s_{\text{beam}}(\tilde{i}) = \sum_{t=1}^{|\tilde{i}|} \log P(\tilde{i}^t | C_u, C_v, \tilde{i}^{<t}). \quad (8)$$

Based on this score, it yields the B most probable item IDs, where B is the beam size. To generate valid IDs, we use a prefix tree Trie (Cormen et al., 2022), following existing works (Tay et al., 2022; Wang et al., 2022).

Trend Score. We calculate the trend score for an item i as follows:

$$s_{\text{trend}}(i) = \log(\frac{r_i}{\max_j r_j} + 1), \quad (9)$$

where r_i is the number of appearances of item i. The logarithmic scale prevents high values from dominating the score, while normalization by the maximum frequency maintains the relative importance across items. To consider the recent trend, r_i is counted only during the N most recent days before the recommendation time $t_{|s_u|+1}$. The window size N is a hyperparameter that can be adjusted according to the characteristics of the recommendation domain or the trend changes.

Score Aggregation. We aggregate both scores for the final ranking. For B items obtained after beam search, the final score is calculated as:

$$s_{\text{final}}(i) = s_{\text{beam}}(\tilde{i}) + \lambda \cdot s_{\text{trend}}(i), \quad (10)$$

where λ is a hyperparameter to control the trend influence. Since trend scores can be pre-computed,

Model	Beauty				Toys				Sports				Yelp			
	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10	R@5	N@5	R@10	N@10
Traditional recommendation models																
GRU4Rec	0.0429	0.0288	0.0643	0.0357	0.0371	0.0254	0.0549	0.0311	0.0237	0.0154	0.0373	0.0197	0.0240	0.0157	0.0398	0.0207
HGN	0.0350	0.0217	0.0589	0.0294	0.0345	0.0212	0.0553	0.0279	0.0203	0.0127	0.0340	0.0171	0.0366	0.0250	0.0532	0.0304
SASRec	0.0323	0.0200	0.0475	0.0249	0.0339	0.0208	0.0442	0.0241	0.0147	0.0089	0.0220	0.0113	0.0284	0.0214	0.0353	0.0245
BERT4Rec	0.0267	0.0165	0.0450	0.0224	0.0210	0.0131	0.0355	0.0178	0.0136	0.0085	0.0233	0.0116	0.0244	0.0159	0.0401	0.0210
FDSA	0.0570	0.0412	0.0777	0.0478	0.0619	0.0455	0.0805	0.0514	0.0283	0.0201	0.0399	0.0238	0.0331	0.0218	0.0534	0.0284
S^3Rec	0.0377	0.0235	0.0627	0.0315	0.0365	0.0231	0.0592	0.0304	0.0229	0.0145	0.0370	0.0190	0.0190	0.0117	0.0321	0.0159
Temporal recommendation models																
TiSASRec	0.0564	0.0359	0.0842	0.0449	0.0665	0.0410	0.0944	0.0499	0.0312	0.0178	0.0474	0.0231	0.0427	0.0323	0.0610	0.0382
TiCoSeRec	0.0377	0.0186	0.0622	0.0260	0.0408	0.0212	0.0663	0.0292	0.0265	0.0147	0.0455	0.0219	0.0433	0.0301	0.0618	0.0354
HM4SR	0.0566	0.0409	0.0773	0.0476	0.0647	0.0480	0.0847	0.0545	0.0288	0.0204	0.0402	0.0241	0.0273	0.0185	0.0447	0.0241
HORAE	0.0508	0.0310	0.0834	0.0415	0.0555	0.0331	0.0902	0.0442	0.0379	0.0235	**0.0620**	0.0313	0.0419	0.0279	0.0663	0.0357
Generative recommendation models																
P5-SID	0.0465	0.0329	0.0638	0.0384	0.0216	0.0151	0.0325	0.0186	0.0295	0.0212	0.0403	0.0247	0.0299	0.0211	0.0432	0.0253
P5-CID	0.0465	0.0325	0.0668	0.0391	0.0223	0.0143	0.0357	0.0186	0.0295	0.0214	0.0420	0.0254	0.0226	0.0155	0.0363	0.0199
P5-SemID	0.0459	0.0327	0.0667	0.0394	0.0264	0.0178	0.0416	0.0227	0.0336	0.0243	0.0481	0.0290	0.0212	0.0143	0.0329	0.0181
TIGER	0.0352	0.0236	0.0533	0.0294	0.0274	0.0174	0.0438	0.0227	0.0176	0.0111	0.0311	0.0146	0.0164	0.0103	0.0262	0.0135
IDGenRec†	0.0463	0.0328	0.0665	0.0393	0.0462	0.0323	0.0651	0.0383	0.0273	0.0186	0.0403	0.0228	0.0310	0.0219	0.0448	0.0263
ELMRec†	0.0372	0.0267	0.0506	0.0310	0.0148	0.0119	0.0193	0.0131	0.0241	0.0181	0.0307	0.0203	0.0424	0.0301	0.0501	0.0324
LETTER	0.0364	0.0243	0.0560	0.0306	0.0309	0.0296	0.0493	0.0262	0.0209	0.0136	0.0331	0.0176	0.0298	0.0218	0.0403	0.0252
LC-Rec	0.0503	0.0352	0.0715	0.0420	0.0543	0.0385	0.0753	0.0453	0.0259	0.0175	0.0384	0.0216	0.0341	0.0235	0.0501	0.0286
GRUT	**0.0741**	**0.0514**	**0.1092**	**0.0627**	**0.0772**	**0.0534**	**0.1113**	**0.0643**	**0.0419**	**0.0285**	0.0615	**0.0348**	**0.0489**	**0.0344**	**0.0699**	**0.0412**
Gain (%)	30.0*	24.8*	29.7*	31.1*	16.1*	11.1*	17.9*	18.1*	10.6*	16.9*	-0.8	11.2*	12.8*	6.5*	5.5*	7.9*

Table 2: Overall performance comparison. The best model is marked in **bold**, and the second-best model is underlined. Gain measures the improvement of the proposed method over the best competitive baseline. '$*$' indicates statistical significance ($p < 0.05$) by a two-tailed t-test. '†' indicates baselines where results differ from the original papers after we addressed preprocessing issues. Please see Appendix B.3.3 for further details.

it adds minimal computational overhead while balancing model predictions with trending items. Notably, trend-aware inference can be applied to various generative recommendation models, as demonstrated in Appendix C.3.

5 Experimental Setup

Datasets. We conduct experiments on four real-world datasets: three subcategories from Amazon review dataset (McAuley et al., 2015; He and McAuley, 2016)[4] ("Sports and Outdoors", "Beauty", and "Toys and Games") and the Yelp dataset[5]. We apply the standard 5-core filtering, removing users and items with fewer than five interactions, following Hua et al. (2023). The data statistics are in Table 6.

Evaluation Protocols and Metrics. We adopt the *leave-one-out* strategy to split train, validation, and test sets following Kang and McAuley (2018); Zheng et al. (2024). For each user sequence, we use the last item for testing, the second last item as validation data, and the remaining items as training data. Rather than sampling items, we perform *full-ranking evaluations* on all items for an accurate assessment. For metrics, we adopt top-k Recall (R@k) and Normalized Discounted Cumulative Gain (N@k) with cutoff $k = \{5, 10\}$.

Baselines. We validate the effectiveness of GRUT against the following eighteen sequential recommenders as baselines. For traditional baselines, we adopt six models: **GRU4Rec** (Hidasi et al., 2016), **HGN** (Ma et al., 2019), **SASRec** (Kang and McAuley, 2018), **BERT4Rec** (Sun et al., 2019), **FDSA** (Zhang et al., 2019), and **S^3Rec** (Zhou et al., 2020). For temporal baselines, we adopt four models: **TiSASRec** (Li et al., 2020), **TiCoSeRec** (Dang et al., 2023), **HM4SR** (Zhang et al., 2025), and **HORAE** (Hu et al., 2025). Lastly, we adopt eight state-of-the-art generative recommenders: **P5-SID**, **P5-CID**, **P5-SemID** (Hua et al., 2023), **TIGER** (Rajput et al., 2023), **IDGenRec** (Tan et al., 2024), **ELMRec** (Wang et al., 2024b), **LETTER** (Wang et al., 2024a), and **LC-Rec** (Zheng et al., 2024). The detailed descriptions are in Appendix B.2.

Implementation Details. The maximum item sequence length was set to 20, following Zheng et al. (2024). We tuned all hyperparameters on the validation set using NDCG@10. We used Adam opti-

[4] https://jmcauley.ucsd.edu/data/amazon/
[5] https://www.yelp.com/dataset

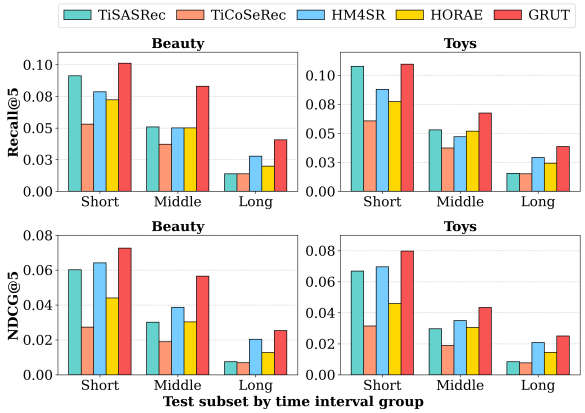

Figure 3: Performance comparison across time interval groups, defined by the number of days between each user's most recent interaction and the target item.

mizer (Kingma and Ba, 2015) with a learning rate of 0.001 and a linear scheduler with a warm-up ratio of 0.05. The maximum text length and the batch size were set to 128. Consistent with the generative baselines (Hua et al., 2023; Tan et al., 2024; Wang et al., 2024b), we initialized with T5-small (Raffel et al., 2020). Due to space limits, we provide further details in Appendix B.3.

6 Experimental Results

6.1 Main Results

Overall Performance. As shown in Table 2, we thoroughly evaluate the effectiveness of GRUT on four real-world datasets, revealing the following key findings: (i) GRUT exhibits the state-of-the-art or comparable performance against existing baselines, achieving up to 30.0% and 24.8% gains in R@5 and N@5, respectively. GRUT outperforms the best temporal baseline by 31.1% in R@5 and exceeds the best generative baseline by 47.3% in R@5. It demonstrates the effectiveness of GRUT in integrating temporal dynamics with generative recommendations. (ii) Temporal models generally surpass both traditional and generative baselines, highlighting the crucial role of temporal information in capturing evolving user preferences.

Performance by Time Interval Group. Figure 3 illustrates the performance of GRUT and temporal models depending on time intervals between each user's most recent interaction and target item. We categorize users into Short, Middle, and Long subsets.[6] Our observations are as follows: (i) Performance decreases across all models as time intervals increase. It reflects user preference drift over long time intervals between interactions, which presents significant challenges for prediction (Li et al., 2020)[7]. (ii) GRUT delivers substantial gains in Long interval groups with gains of 32.6–46.0% in R@5 and 20.2–24.0% in N@5 compared to the best baseline HM4SR. It confirms the effectiveness of GRUT in identifying preference shifts of users. (iii) The temporal models that utilize textual metadata (HM4SR, HORAE, GRUT) relatively perform better with longer temporal gaps, implying that textual metadata provides valuable signals when recent behavioral items are insufficient.

[6]Please refer to Appendix B.1 for detailed statistics.

Type	Beauty R@5	Beauty N@5	Toys R@5	Toys N@5
Target-relative + Abs.	**0.0741**	**0.0514**	**0.0772**	**0.0534**
None	0.0575	0.0400	0.0569	0.0396
Absolute	0.0581	0.0402	0.0603	0.0412
Relative	0.0582	0.0406	0.0586	0.0407
Target-relative	0.0660	0.0468	0.0672	0.0478
Relative + Abs.	0.0595	0.0415	0.0618	0.0428

Table 3: Performance of GRUT over time information types in C_u. 'Abs.' denotes the absolute timestamps.

Model	Beauty R@5	Beauty N@5	Toys R@5	Toys N@5
GRUT	**0.0741**	**0.0514**	**0.0772**	**0.0534**
w/o user-level	0.0575	0.0400	0.0569	0.0396
w/o item-level	0.0713	0.0492	0.0755	0.0518
w/o trend score ($\lambda = 0$)	0.0731	0.0505	0.0754	0.0522
w/o context embedding	0.0711	0.0500	0.0723	0.0529
w/o epsilon ($\epsilon = 1$)	0.0681	0.0486	0.0739	0.0535

Table 4: Ablation study of GRUT. We examine the effect of (i) time-aware prompting, (ii) trend-aware inference, and (iii) additional techniques.

6.2 Ablation Study

Effect of Time Information Types. Table 3 presents the impact of temporal information types in the user-level temporal context C_u. We compare six variants: None, Absolute timestamps (t_i), Relative intervals ($t_{i+1} - t_i$), Target-relative intervals ($t_{|s_u|+1} - t_i$), Relative + Absolute, and Target-relative + Absolute[8]. All time-aware variants outperform the baseline, with up to 35.7% gains in R@5, confirming the benefits of verbalizing temporal dynamics. The target-relative intervals especially achieve the highest performance, suggesting that recency relative to recommendation time effectively captures user preferences. Notably, combining absolute timestamps and interval information consistently yields gains of 2.2%–14.9% in R@5.

[7]This is also shown in our analysis in Appendix C.1.

[8]See Appendix B.4 for detailed prompts of each variant.

User sequence (ASIN:A1M2CZP3XOVZO5)					
Image					
Name	Edward Doll	Bella Doll	SpongeBob Game	InnoTab Storage (Pink)	InnoTab Storage (Blue)
Category	Dolls	Dolls	Learning Game	System Acc.	System Acc.
Time	2010-01-11	2010-01-11	2010-02-16	2012-12-11	2012-12-11
GRUT Top-5 predictions at 2012-12-11 (Without temporal information)					
Ranking	1	2	3	4	5
Image					
Name	SpongeBob Beanie	Eclipse Victoria Doll	2012 Holiday Doll	Photo Fashion Doll	Carlisle Doll
Category	Plush	Dolls	Dolls	Dolls	Dolls
GRUT Top-5 predictions at 2012-12-11 (With temporal information)					
Ranking	1	2	3	4	5
Image					
Name	InnoTab 2S Tablet	InnoTab 2 White Tablet	InnoTab 2 Pink Tablet	Winx Bloom Doll	InnoTab Thomas
Category	Learning Tablet	Learning Tablet	Learning Tablet	Dolls	Learning Software

Table 5: GRUT's top-5 predictions on the Toys dataset with and without temporal information. The five most recent items in the sequence are shown for simplicity. The target item is marked with a red dotted line.

It demonstrates that two distinct forms of temporal signals successfully complement each other.

Effect of Various Components. Table 4 shows the effectiveness of various components in GRUT. (i) Both user-level temporal context C_u and item-level transition context C_v contribute to performance. Specifically, temporal information in C_u enhances R@5 by up to 35.7%. It highlights the importance of user-specific temporal patterns, while transition patterns also convey valuable additional guidance. (ii) Trend-aware inference not only provides flexibility in controlling trend influence but also improves recommendation accuracy by up to 2.4% in R@5. This improvement comes from incorporating real-time trend signals unavailable during training. (iii) The context-type embeddings $\mathbf{P}$ in Eq. (5) and ϵ in Eq. (6) boost R@5 by up to 6.8% and 8.9%, respectively. It indicates that distinguishing context types while ensuring transitions as supplementary information enhances recommendation accuracy.

6.3 In-depth Analysis

Case Study. Table 5 illustrates the impact of temporal information on the recommendation results of GRUT. Without temporal information, the model recommends '*Plush*' and '*Dolls*', missing that the user's purchasing pattern has shifted over the past two years from '*Dolls*' to '*InnoTab*'. Conversely, GRUT with temporal information successfully identifies the preference shift and recommends an '*InnoTab 2S Tablet*', while also suggesting a '*Winx Bloom Doll*'. It depicts that temporal dynamics are crucial in capturing user preferences that evolve over time, leading to more accurate recommendations. Please see Appendix D for additional cases.

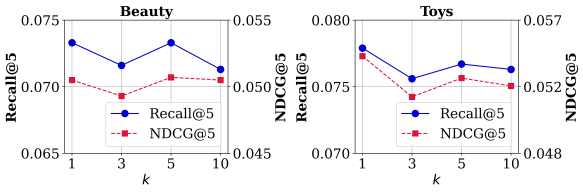

Figure 4: Performance of GRUT over varying the number of neighboring items k in C_v.

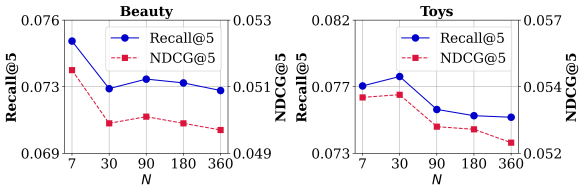

Figure 5: Performance of GRUT over varying the window size N in the trend score.

Hyperparameter Sensitivity. Figures 4 and 5 show the performance of GRUT when varying neighboring items k and trend window size N. We observe optimal performance at $k = 1$ for both Beauty and Toys datasets, suggesting that more neighbors may introduce noise. For N, the optimal values for Beauty and Toys are 7 and 30, respectively. It highlights the importance of adjusting the trend window size according to how rapidly preferences change in each domain. An additional analysis of ϵ and L are in Appendix C.4.

7 Conclusion

We propose GRUT, a novel model that effectively incorporates temporal dynamics into GR. Our time-aware prompting captures both user-specific temporal patterns and item-level transition knowledge. Additionally, trend-aware inference enhances rankings by injecting trend information. Extensive experiments on four benchmark datasets demonstrate improvements of GRUT compared to state-of-the-art recommendation models, up to 30.0% in R@5 and 24.8% in N@5, particularly in scenarios with long time intervals between interactions. Our work highlights the importance of time awareness in GR, opening new directions for future models that better reflect evolving user preferences.

8 Limitations

The limitations of our work are as follows. (i) To construct the item-level transition context C_v, we include all transition pairs from the training data in a global item transition graph. This approach has a limitation as it may incorporate noise or spurious patterns, *e.g.*, accidental clicks. This challenge has also been noted in previous work (Zhang et al., 2024), and future research could apply denoising techniques to extract only meaningful temporal patterns. (ii) Our method currently incorporates temporal information uniformly across all users. However, as pointed out in the existing work (He et al., 2023), users exhibit diverse purchasing patterns which our approach does not explicitly model, presenting another limitation of our work. We believe that modeling user preferences in a user-adaptive manner would be meaningful. For instance, in trend-aware inference, the value of λ could be dynamically adjusted according to individual patterns. We leave further exploration as future work.

Ethics Statement

This work fully complies with the ACL Ethics Policy. We declare that there are no ethical issues in this paper. The scientific artifacts we have utilized are publicly available for research under permissive licenses, and the utilization of these tools is consistent with their intended applications.

References

Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. In *RecSys*, pages 1007–1014.

Yongjun Chen, Zhiwei Liu, Jia Li, Julian J. McAuley, and Caiming Xiong. 2022. Intent contrastive learning for sequential recommendation. In *WWW*, pages 2172–2182.

Sung Min Cho, Eunhyeok Park, and Sungjoo Yoo. 2020. MEANTIME: mixture of attention mechanisms with multi-temporal embeddings for sequential recommendation. In *RecSys*, pages 515–520.

Zhendong Chu, Zichao Wang, Ruiyi Zhang, Yangfeng Ji, Hongning Wang, and Tong Sun. 2024. Improve temporal awareness of llms for sequential recommendation. *CoRR*, abs/2405.02778.

Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. 2022. *Introduction to algorithms*. MIT press.

Yizhou Dang, Enneng Yang, Guibing Guo, Linying Jiang, Xingwei Wang, Xiaoxiao Xu, Qinghui Sun, and Hong Liu. 2023. Uniform sequence better: Time interval aware data augmentation for sequential recommendation. In *AAAI*, pages 4225–4232.

Ziwei Fan, Zhiwei Liu, Jiawei Zhang, Yun Xiong, Lei Zheng, and Philip S. Yu. 2021. Continuous-time sequential recommendation with temporal graph collaborative transformer. In *CIKM*, pages 433–442.

Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2022. Recommendation as language processing (RLP): A unified pretrain, personalized prompt & predict paradigm (P5). In *RecSys*, pages 299–315.

Ruining He and Julian J. McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *WWW*, pages 507–517.

Zhicheng He, Weiwen Liu, Wei Guo, Jiarui Qin, Yingxue Zhang, Yaochen Hu, and Ruiming Tang. 2023. A survey on user behavior modeling in recommender systems. In *IJCAI*, pages 6656–6664.

Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2016. Session-based recommendations with recurrent neural networks. In *ICLR*.

Shirui Hu, Weichang Wu, Zuoli Tang, Zhaoxin Huan, Lin Wang, Xiaolu Zhang, Jun Zhou, Lixin Zou, and Chenliang Li. 2025. Horae: Temporal multi-interest pre-training for sequential recommendation. *ACM Trans. Inf. Syst.*

Wenyue Hua, Shuyuan Xu, Yingqiang Ge, and Yongfeng Zhang. 2023. How to index item ids for recommendation foundation models. In *SIGIR-AP*, pages 195–204.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *EACL*, pages 874–880.

Karen Sparck Jones. 2004. A statistical interpretation of term specificity and its application in retrieval. *J. Documentation*, 60(5):493–502.

Wang-Cheng Kang and Julian J. McAuley. 2018. Self-attentive sequential recommendation. In *ICDM*, pages 197–206.

Sein Kim, Hongseok Kang, Seungyoon Choi, Donghyun Kim, Min-Chul Yang, and Chanyoung Park. 2024. Large language models meet collaborative filtering: An efficient all-round llm-based recommender system. In *KDD*, pages 1395–1406.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024. Nv-embed: Improved techniques for training llms as generalist embedding models. *CoRR*, abs/2405.17428.

Jiacheng Li, Ming Wang, Jin Li, Jinmiao Fu, Xin Shen, Jingbo Shang, and Julian J. McAuley. 2023a. Text is all you need: Learning language representations for sequential recommendation. In *KDD*, pages 1258–1267.

Jiacheng Li, Yujie Wang, and Julian J. McAuley. 2020. Time interval aware self-attention for sequential recommendation. In *WSDM*, pages 322–330.

Xinhang Li, Chong Chen, Xiangyu Zhao, Yong Zhang, and Chunxiao Xing. 2023b. E4srec: An elegant effective efficient extensible solution of large language models for sequential recommendation. *CoRR*, abs/2312.02443.

Yongqi Li, Xinyu Lin, Wenjie Wang, Fuli Feng, Liang Pang, Wenjie Li, Liqiang Nie, Xiangnan He, and Tat-Seng Chua. 2024. A survey of generative search and recommendation in the era of large language models. *CoRR*.

Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, Xiang Wang, and Xiangnan He. 2024. Llara: Large language-recommendation assistant. In *SIGIR*, pages 1785–1795.

Xinyu Lin, Wenjie Wang, Yongqi Li, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. 2024. Bridging items and language: A transition paradigm for large language model-based recommendation. In *KDD*, pages 1816–1826.

Qidong Liu, Xian Wu, Wanyu Wang, Yejing Wang, Yuanshao Zhu, Xiangyu Zhao, Feng Tian, and Yefeng Zheng. 2025. Llmemb: Large language model can be a good embedding generator for sequential recommendation. In *AAAI*, pages 12183–12191.

Qidong Liu, Xian Wu, Yejing Wang, Zijian Zhang, Feng Tian, Yefeng Zheng, and Xiangyu Zhao. 2024. LLM-ESR: large language models enhancement for long-tailed sequential recommendation. In *NeurIPS*.

Chen Ma, Peng Kang, and Xue Liu. 2019. Hierarchical gating networks for sequential recommendation. In *KDD*, pages 825–833.

Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based recommendations on styles and substitutes. In *SIGIR*, pages 43–52.

Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B. Hall, Daniel Cer, and Yinfei Yang. 2022. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. In *Findings of the ACL*, pages 1864–1874.

Seongmin Park, Mincheol Yoon, Minjin Choi, and Jongwuk Lee. 2025. Temporal linear item-item model for sequential recommendation. In *WSDM*, pages 354–362.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q. Tran, Jonah Samost, Maciej Kula, Ed H. Chi, and Mahesh Sathiamoorthy. 2023. Recommender systems with generative retrieval. In *NeurIPS*, pages 10299–10315.

Xubin Ren, Wei Wei, Lianghao Xia, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. 2024. Representation learning with large language models for recommendation. In *WWW*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

Leheng Sheng, An Zhang, Yi Zhang, Yuxin Chen, Xiang Wang, and Tat-Seng Chua. 2025. Language representations can be what recommenders need: Findings and potentials. In *ICLR*.

Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *CIKM*, pages 1441–1450.

Juntao Tan, Shuyuan Xu, Wenyue Hua, Yingqiang Ge, Zelong Li, and Yongfeng Zhang. 2024. Idgenrec: Llm-recsys alignment with textual id learning. In *SIGIR*, page 355–364.

Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *WSDM*, pages 565–573.

Yi Tay, Vinh Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Prakash Gupta, Tal Schuster, William W. Cohen, and Donald Metzler. 2022. Transformer memory as a differentiable search index. In *NeurIPS*, pages 21831–21843.

Changxin Tian, Zihan Lin, Shuqing Bian, Jinpeng Wang, and Wayne Xin Zhao. 2022. Temporal contrastive pre-training for sequential recommendation. In *CIKM*, pages 1925–1934.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Jianling Wang, Raphael Louca, Diane Hu, Caitlin Cellier, James Caverlee, and Liangjie Hong. 2020a. Time to shop for valentine's day: Shopping occasions and sequential recommendation in e-commerce. In *WSDM*, pages 645–653.

Wenjie Wang, Honghui Bao, Xinyu Lin, Jizhi Zhang, Yongqi Li, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. 2024a. Learnable item tokenization for generative recommendation. In *CIKM*, pages 2400–2409.

Xinfeng Wang, Jin Cui, Fumiyo Fukumoto, and Yoshimi Suzuki. 2024b. Enhancing high-order interaction awareness in llm-based recommender model. In *EMNLP*, pages 11696–11711.

Yujing Wang, Yingyan Hou, Haonan Wang, Ziming Miao, Shibin Wu, Qi Chen, Yuqing Xia, Chengmin Chi, Guoshuai Zhao, Zheng Liu, Xing Xie, Hao Sun, Weiwei Deng, Qi Zhang, and Mao Yang. 2022. A neural corpus indexer for document retrieval. In *NeurIPS*.

Ziyang Wang, Wei Wei, Gao Cong, Xiao-Li Li, Xianling Mao, and Minghui Qiu. 2020b. Global context enhanced graph neural networks for session-based recommendation. In *SIGIR*, pages 169–178.

Siheng Xiong, Ali Payani, Ramana Kompella, and Faramarz Fekri. 2024. Large language models can learn temporal reasoning. In *ACL*, pages 10452–10470.

Lanling Xu, Zhen Tian, Gaowei Zhang, Junjie Zhang, Lei Wang, Bowen Zheng, Yifan Li, Jiakai Tang, Zeyu Zhang, Yupeng Hou, Xingyu Pan, Wayne Xin Zhao, Xu Chen, and Ji-Rong Wen. 2023. Towards a more user-friendly and easy-to-use benchmark library for recommender systems. In *SIGIR*, pages 2837–2847.

Shuyuan Xu, Wenyue Hua, and Yongfeng Zhang. 2024. Openp5: An open-source platform for developing, training, and evaluating llm-based recommender systems. In *SIGIR*, pages 386–394.

Howard Yen, Tianyu Gao, and Danqi Chen. 2024. Long-context language modeling with parallel context encoding. In *ACL*, pages 2588–2610.

Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. 2022. Soundstream: An end-to-end neural audio codec. *IEEE ACM Trans. Audio Speech Lang. Process.*, 30:495–507.

Shengzhe Zhang, Liyi Chen, Dazhong Shen, Chao Wang, and Hui Xiong. 2025. Hierarchical time-aware mixture of experts for multi-modal sequential recommendation. In *WWW*, pages 3672–3682.

Shengzhe Zhang, Liyi Chen, Chao Wang, Shuangli Li, and Hui Xiong. 2024. Temporal graph contrastive learning for sequential recommendation. In *AAAI*, pages 9359–9367.

Tingting Zhang, Pengpeng Zhao, Yanchi Liu, Victor S. Sheng, Jiajie Xu, Deqing Wang, Guanfeng Liu, and Xiaofang Zhou. 2019. Feature-level deeper self-attention network for sequential recommendation. In *IJCAI*, pages 4320–4326.

Bowen Zheng, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, Ming Chen, and Ji-Rong Wen. 2024. Adapting large language models by integrating collaborative semantics for recommendation. In *ICDE*, pages 1435–1448.

Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. 2020. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *CIKM*, pages 1893–1902.

Dataset	Beauty	Toys	Sports	Yelp
#Users	22,363	19,412	35,598	30,431
#Items	12,101	11,924	18,357	20,033
#Inter.	198,502	167,597	296,337	316,354
Density	0.0734%	0.0724%	0.0453%	0.0519%
Avg. Length	8.9	8.6	8.3	10.4
Avg. Interval	69.6d	86.0d	74.1d	18.6d

Table 6: Statistics of four benchmark datasets.

Dataset	Short	Middle	Long
Beauty	9,719	5,052	7,592
Toys	9,518	3,323	6,571

Table 7: The number of users of each test subset in Figure 3, categorized by time interval between the most recent interaction and the target item.

A Additional Related Work

A.1 Sequential Recommendation

The goal of sequential recommendation is to predict the following items that users may be interested in based on their behavior sequences. Early works focus on various neural-based encoders, such as convolutional neural networks (Tang and Wang, 2018), gated recurrent units (Hidasi et al., 2016), and Transformers (Kang and McAuley, 2018; Sun et al., 2019; Ma et al., 2019). Recent approaches incorporate item textual attributes, employing separate self-attention mechanisms for item and feature information (Zhang et al., 2019), while leveraging self-supervised objectives to learn item-attribute correlations (Zhou et al., 2020). However, these models are limited in fully utilizing the reasoning power of LLMs and textual semantics, unlike generative recommendation approaches.

A.2 LLM-based Recommendation

Recent studies (Bao et al., 2023; Li et al., 2023b; Liao et al., 2024; Kim et al., 2024) employ LLMs directly as re-rankers, where the model is prompted with a subset of item candidates (typically 20 items, including one ground-truth item) to recommend items likely to be preferred by users. These approaches utilize the rich knowledge and reasoning capabilities of LLMs to enhance recommendation quality. Meanwhile, some works (Ren et al., 2024; Liu et al., 2024, 2025; Sheng et al., 2025) only extract LLM knowledge to initialize or enhance traditional recommendation models, avoiding the costly LLM fine-tuning. Unlike these approaches, our work focuses on direct item ID generation, performing full ranking across the entire item space rather than re-ranking from sampled candidates.

B Additional Experimental Setup

B.1 Datasets

Following the previous works (Tan et al., 2024; Wang et al., 2024b; Geng et al., 2022; Hua et al., 2023), we use the Amazon Review dataset containing product reviews and item metadata from 1996 to 2014. We also use the Yelp dataset with business reviews from 2019 to 2020. Table 6 presents statistics of preprocessed datasets. We further provide the number of users for each subset in Figure 3.

B.2 Baselines

We adopt six traditional models, four temporal models, and eight generative models for baselines.

- **GRU4Rec** (Hidasi et al., 2016) encodes sequential user behavior using Gated Recurrent Units.
- **HGN** (Ma et al., 2019) models long- and short-term interests with a hierarchical gating network.
- **SASRec** (Kang and McAuley, 2018) leverages uni-directional Transformers to represent users based on their most recent interaction.
- **BERT4Rec** (Sun et al., 2019) employs bi-directional self-attention for masked item prediction tasks.
- **FDSA** (Zhang et al., 2019) separately models feature-level and item-level self-attention.
- **S^3Rec** (Zhou et al., 2020) enhances representation learning with self-supervised auxiliary tasks.
- **TiSASRec** (Li et al., 2020) introduces relative time interval embeddings as keys and values in self-attention mechanisms.
- **TiCoSeRec** (Dang et al., 2023) improves contrastive learning by augmenting sequences with controlled time interval distributions.
- **HM4SR** (Zhang et al., 2025) employs a mixture of experts architecture to integrate temporal patterns with multi-modal (ID, text) representations.
- **HORAE** (Hu et al., 2025) enhances multi-interest learning with temporal dynamics.
- **P5-SID** (Hua et al., 2023) assigns numeric IDs sequentially based on the item appearance.
- **P5-CID** (Hua et al., 2023) clusters items based on co-occurrences to generate numeric IDs.
- **P5-SemID** (Hua et al., 2023) assigns numeric IDs using item metadata like categories.
- **TIGER** (Rajput et al., 2023) introduces codebook IDs generated through RQ-VAE.
- **IDGenRec** (Tan et al., 2024) generates textual

Hyperparameters	Beauty	Toys	Sports	Yelp
ϵ	0.01	0.01	0.001	0.01
k	1	1	1	1
L	5	2	3	3
λ	0.3	0.4	0.1	0.2
N	7	30	30	30

Table 8: Final hyperparameters for GRUT.

IDs with a generator based on item metadata.

- **ELMRec** (Chen et al., 2022) adopts high-order relationships using soft prompts and re-ranking strategies with numeric IDs.
- **LC-Rec** (Zheng et al., 2024) combines RQ-VAE IDs with multi-task learning to integrate language and collaborative semantics.

B.3 Additional Implementation Details

We conducted all experiments with 2 NVIDIA RTX A6000, 512 GB memory, and 2 AMD EPYC 74F3.

B.3.1 Details for GRUT

We implemented GRUT on OpenP5 (Xu et al., 2024). We tuned ϵ in $\{10^{-3}, 10^{-2}, 10^{-1}, 10^{0}\}$, k in $\{1, 3, 5, 10\}$, L in $\{1, 2, 3, 4, 5, 7\}$, λ in the range of $[0, 1]$ with step size 0.1, N in $\{7, 30, 180, 360\}$, τ in the range of $[2^7, 2^{10}]$ with exponentially increasing steps in powers of 2, c in $[0.5, 1]$. Due to computational constraints, hyperparameters were tuned sequentially. We first optimize ϵ, followed by k, L, λ, and finally N. The final hyperparameters are in Table 8. We sort the user history in C_u and C_v in reverse order to prevent recent items from being truncated following the existing work (Li et al., 2023a). The checkpoints achieving the highest NDCG@10 on the validation set were selected for evaluation on the test set. For hyperparameter sensitivity analysis (Figure 4, 5, and 7), we measured performance without trend-aware inference to ensure a more accurate analysis, *i.e.*, $\lambda = 0$. For calculating the trend score in Eq. (9), the recommendation day itself was excluded, *i.e.*, from day $t_{|s_u|+1} - N - 1$ to day $t_{|s_u|+1} - 1$.

The keywords are extracted from each item's textual metadata and assigned as textual IDs. Rather than learning an ID generator during training (Tan et al., 2024), we precompute TF-IDF scores (Jones, 2004) over the metadata before training. We then select the highest-scoring terms and assign them as IDs. To maintain consistency with a backbone LLM, T5 tokenizer (Raffel et al., 2020) is adopted. For the Amazon Beauty, Toys, and Sports dataset, we concatenate each item's title, brand, category, and description. The name, city, and category fields are used for the Yelp dataset. For the Toys dataset, examples of item IDs include '*musical-piano-concert-keyboard-displays*', '*dinosaur-safari-dragon-knight-headed*', and '*doll-loving-bedroom-mirrored-comfy*'.

B.3.2 Details for Baselines

For traditional and temporal recommendation models, we conducted all experiments on the open-source RecBole library (Xu et al., 2023)[9]. We thoroughly tuned each hyperparameter following guidance from the original papers. The models were optimized using the Adam optimizer with a learning rate of 0.001, a batch size of 256, and an embedding dimension of 64. The training was stopped when the validation NDCG@10 showed no improvement for 10 consecutive epochs. For HM4SR (Zhang et al., 2025), we utilized only ID and text embeddings without image embeddings to ensure fair comparison. While HORAE (Hu et al., 2025) used Amazon 2018 datasets, we pre-trained the model with the corresponding Amazon 2014 datasets (Food, CDs, Kindle, Movies, and Home) for consistency with our experimental setup, then fine-tuned the pre-trained model on Beauty, Toys, Sports, and Yelp datasets, respectively.

For all generative baselines, we follow the official code if publicly available, *e.g.*, P5-variants (Hua et al., 2023)[10], IDGenRec (Tan et al., 2024)[11], ELMRec (Wang et al., 2024b)[12], LC-Rec (Zheng et al., 2024)[13], and LETTER (Wang et al., 2024a)[14]. For TIGER (Rajput et al., 2023), we implemented the model based on the details in the paper since the official code was not publicly available. We used the Sentence-T5 (Ni et al., 2022) for semantic embeddings with a hidden dimension size of 768. The vocabulary size was set to 1024 (256×4). We used T5-small (Raffel et al., 2020) for P5, IDGenRec, and ELMRec, following the official codebase. We instantiate LETTER on TIGER.

For ELMRec, when applying to the Yelp dataset, which is not included in the original paper, we excluded the explanation generation task due to insufficient textual metadata. Additionally, we did not apply the 'reranking approach' proposed in ELMRec for the Yelp dataset since items within a user sequence can reappear as target items. For all other

[9] https://recbole.io/
[10] https://github.com/Wenyueh/LLM-RecSys-ID
[11] https://github.com/agiresearch/IDGenRec
[12] https://github.com/WangXFng/ELMRec
[13] https://github.com/RUCAIBox/LC-Rec
[14] https://github.com/HonghuiBao2000/LETTER

implementation details, including hyperparameter search ranges, we thoroughly followed the specifications described in the ELMRec manuscript.

For LC-Rec, we fully fine-tuned LLaMA-7B (Touvron et al., 2023), adhering to the authors' guidelines with some modifications for the Amazon 2014 dataset. In the asymmetric item prediction task, we set the number of training samples based on the interactions for each dataset, *e.g.*, 20K, 15K, 25K, and 25K for the Beauty, Toys, Sports, and Yelp datasets, respectively. For the personalized preference inference task, we used `gpt-4o-mini-2024-07-18` to infer user preferences on Amazon datasets and omitted this task on Yelp due to insufficient textual metadata.

B.3.3 Modifications to Preprocessing of ELMRec and IDGenRec

For ELMRec, we followed the P5-SID approach used in the official code, but with important modifications to address data leakage issues in the original P5 implementation (Geng et al., 2022). Following recent works (Hua et al., 2023; Xu et al., 2024; Rajput et al., 2023), we excluded validation and test items while assigning numeric IDs. As a result, the results in Table 2 differ from those reported in the original paper. The original P5 methodology assigned consecutive numeric IDs to items based on their appearance order within each user sequence, including validation and test items. For instance, a user sequence is represented as `[8921, 8922, ..., 8927]`, where `8927` becomes the test item in the leave-one-out evaluation. Since P5 uses the SentencePiece tokenizer (Sennrich et al., 2016), test items potentially share subwords with training times in the sequence. It creates unintended correlations that implicitly lead to information leakage during inference. To prevent this issue, we conducted our experiments following works (Hua et al., 2023; Xu et al., 2024), applying sequential indexing only to training items while explicitly excluding validation and test items. This issue has already been identified in previous works (Rajput et al., 2023; Lin et al., 2024)[15].

For IDGenRec, we excluded user IDs from input prompts, following guidance from the original authors[16]. This explains the differences in performance in Table 2 compared to the original paper. Initially, IDGenRec uses both item IDs from the user history and a user ID. The user ID is created by concatenating all sequence items and processing them through the ID generator. For example, with an item sequence $i_1 \rightarrow i_2 \rightarrow i_3 \rightarrow i_4$, information from all items is used. However, this approach creates a potential data leakage issue in leave-one-out evaluation settings, as the user ID contains information about the test item i_4. To address this concern, we removed user IDs from our implementation.

B.4 Examples of Input Prompts for Table 3

We present six types of user-level temporal context C_u with their corresponding input prompt formats shown in Table 3.

None:
What would the user purchase after $\tilde{i}_1, \tilde{i}_2, \cdots, \tilde{i}_{|s_u|}$?

Absolute:
What would the user purchase after $\tilde{i}_1\ (t_1), \tilde{i}_2\ (t_2), \cdots, \tilde{i}_{|s_u|}\ (t_{|s_u|})$?

Relative:
What would the user purchase after $\tilde{i}_1$ (after $t_2 - t_1$), $\tilde{i}_2$ (after $t_3 - t_2$), $\cdots, \tilde{i}_{|s_u|}$?

Target-relative:
What would the user purchase after $\tilde{i}_1\ (t_{|s_u|+1} - t_1$ ago), $\tilde{i}_2\ (t_{|s_u|+1} - t_2$ ago), $\cdots, \tilde{i}_{|s_u|}\ (t_{|s_u|+1} - t_{|s_u|}$ ago) ?

Relative + Absolute:
What would the user purchase after $\tilde{i}_1\ (t_1$, after $t_2 - t_1$), $\tilde{i}_2\ (t_2$, after $t_2 - t_2$), $\cdots, \tilde{i}_{|s_u|}\ (t_{|s_u|})$?

Target-relative + Absolute:
The current date is $t_{|s_u|+1}$.
What would the user purchase after $\tilde{i}_1\ (t_1, \Delta t_1$ ago), $\tilde{i}_2\ (t_2, \Delta t_2$ ago), $\cdots$, $\tilde{i}_{|s_u|}\ (t_{|s_u|}, \Delta t_{|s_u|}$ ago) ?

[15] Please refer to Appendix D of Rajput et al. (2023) and Appendix A.6 of Lin et al. (2024) for details.

[16] https://github.com/agiresearch/IDGenRec/issues/1

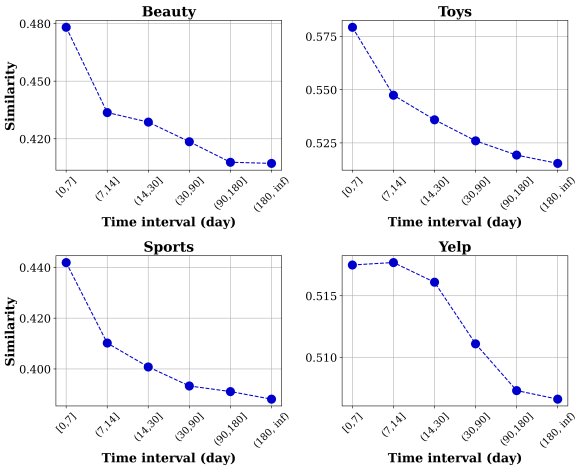

Figure 6: Similarity of item pairs by time interval groups. The x-axis is the time interval between two consecutive items, and the y-axis is the semantic similarity of items.

ID	Temporal	Beauty R@5	Beauty N@5	Toys R@5	Toys N@5
Ours	✓	**0.0741**	**0.0514**	**0.0772**	**0.0534**
	✗	0.0582	0.0404	0.0558	0.0392
IDGenRec	✓	**0.0711**	**0.0489**	**0.0687**	**0.0463**
	✗	0.0533	0.0374	0.0487	0.0329
Title ID	✓	**0.0575**	**0.0396**	**0.0588**	**0.0416**
	✗	0.0411	0.0293	0.0444	0.0314

Table 9: Performance of GRUT over various IDs.

C Additional Experimental Results

C.1 Preference Shifts over Time Interval

We examined whether user preferences evolve over time by analyzing item similarity across different time intervals. Figure 6 shows text similarity between consecutive items grouped by time intervals. For calculating similarity, we generated text embeddings using NVEmbed (Lee et al., 2024) from item metadata[17]. Each consecutive item pair from user sequences is grouped by time intervals of interaction, *e.g.*, intervals of 8 days fall into $(7, 14]$, and interactions of the same day belong to $[0, 7]$. The results clearly show decreasing similarity between consecutive items as time intervals increase across all datasets. It suggests that user preferences shift more significantly over longer time intervals. Despite these challenges, our model demonstrates superior performance, especially in scenarios with long time gaps, as demonstrated in Figure 3.

[17]For Amazon datasets, we used title, brand, and categories. We used name, city, and categories for the Yelp dataset.

Model	Trend	Beauty R@5	Beauty N@5	Toys R@5	Toys N@5
IDGenRec	✓	**0.0480**	**0.0332**	**0.0480**	**0.0328**
	✗	0.0463	0.0328	0.0462	0.0323
LETTER	✓	**0.0373**	**0.0250**	**0.0324**	**0.0211**
	✗	0.0364	0.0243	0.0309	0.0202
LC-Rec	✓	**0.0521**	**0.0365**	**0.0574**	**0.0399**
	✗	0.0503	0.0352	0.0543	0.0385

Table 10: Effectiveness of trend-aware inference when applying to existing generative recommenders.

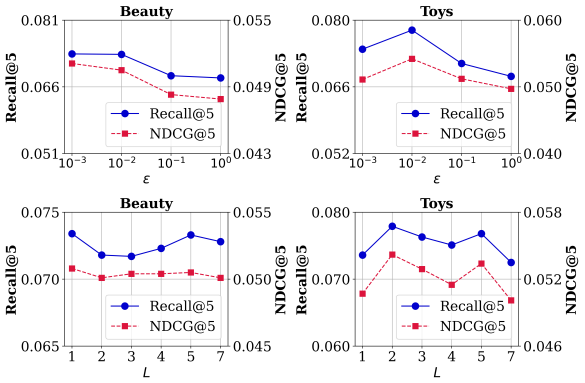

Figure 7: Performance of GRUT over varying (i) ϵ that controls the influence of item-level transition patterns and (ii) the number of the most recent items L in C_v.

C.2 Effect of ID Variants

Table 9 demonstrates the effectiveness of GRUT across different ID variants. When replacing our IDs with those from prior work (Tan et al., 2024) or titles[18], GRUT consistently improves performance, enhancing R@5 and N@5 by 28.7%–41.1% and 27.2%–40.7%, respectively. It implies the robustness of our temporal integration approach regardless of ID schemes.

C.3 Generalizability of Trend-aware Inference

Table 10 illustrates the effect of trend-aware inference when applying to existing generative recommendation baselines, *e.g.*, IDGenRec, LETTER, and LC-Rec. Notably, all baselines consistently show performance improvements, achieving average gains of 4.0% in R@5 and 2.9% in N@5, respectively. This confirms that our trend-aware inference effectively enhances recommendation performance regardless of the underlying architecture.

C.4 Hyperparameter Sensitivity

Figures 7 shows the performance of GRUT depending on ϵ, which controls the influence of C_v, and

[18]We appended additional digits for duplicated titles to ensure uniqueness.

User sequence (ASIN: A2N8D20LSUU85O)					
Image					
Name	WL V911 RC Helicopter	Battery Checker	WL V911 Battery 5-Pack	Helizone Edition	Sofia Amulet
Category	RC Helicopters	Battery Chargers	Vehicle Batteries	RC Propellers	Pretend Play
Time	2013-02-18	2013-02-18	2013-02-18	2013-05-03	2013-12-30
GRUT Top 5 prediction at 2013-12-30					
Ranking	1	2	3	4	5
Image					
Name	Sofia Animals	Sofia Royal Family	Sofia Royal Bed	Magic Castle Friends	Magic Gift Set
Category	Dolls& Playsets	Dolls	Playsets	Action Figs. & Playsets	Playsets
GRUT Top 5 prediction at 2014-06-30					
Ranking	1	2	3	4	5
Image					
Name	Voltage Checker	Double Horse 9053 Gyro	WL V911 Red V2	WL V912 Gyro RTF	Syma Quad Copter
Category	Vehicle Batteries	Vehicle Batteries	RC Helicopters	RC Helicopters	RC Helicopters

Table 11: GRUT's top-5 predictions on the Toys dataset at different inference timestamps. The five most recent items in the sequence are shown for simplicity.

User sequence (ASIN: A2V65NBADV4HY4)					
Image					
Name	Learning Toolbench	Peek-a-Blocks Giraffe	Touch & Tickle Rounds	Garden Hose Sprinkler	LEGO Sorting System
Category	Learning Toys	Baby Toys	Gag Toys	Outdoor Toys	Building Toys
Time	2005-10-31	2006-08-03	2006-08-03	2013-11-19	2014-01-01
GRUT Top 5 prediction at 2014-01-01 ($\lambda = 0.0$)					
Ranking	1	2	3	4	5
Image					
Name	LEGO 6-Case Storage Unit	Star Wars Box	LEGO City Box	Star Wars Battle Bridge	Rainbow Loom
Category	Building Toys	Vehicle Playsets	Vehicle Playsets	Toys & Games	Toys & Games
s_{trend}	0.1361	0.0206	0.0000	0.0206	0.6931
GRUT Top 5 prediction at 2014-01-01 ($\lambda = 0.5$)					
Ranking	1	2	3	4	5
Image					
Name	Rainbow Loom	LEGO 6-Case Storage Unit	Star Wars Box	LEGO City Box	Star Wars Battle Bridge
Category	Toys & Games	Building Toys	Vehicle Playsets	Vehicle Playsets	Toys & Games
s_{trend}	0.6931	0.1361	0.0206	0.0000	0.0206

Table 12: GRUT's top-5 predictions on the Toys dataset with and without trend-aware inference. The target item is marked with a red dotted line.

the number of the most recent items in C_v, denoted as L. For ϵ, the optimal values for Beauty and Toys are 0.001 and 0.01, respectively. This suggests that a large ϵ makes the model excessively focus on item transition patterns, neglecting user-specific signals. Meanwhile, the optimal values of L for Beauty and Toys are 1 and 2, respectively. It highlights how the dataset characteristics directly influence the optimal hyperparameters.

D Additional Case Study

D.1 Effect of Inference Timestamp Shift

Table 11 presents the recommendation results of GRUT for the same user, evaluated at different inference timestamps ($\Delta t_{|s_u|+1}$) in the user-level temporal context C_u. When the inference occurs shortly after the user's last interaction, GRUT emphasizes short-term interests, recommending products related to the most recent purchase, *e.g.*, '*Sofia*'. In contrast, when the inference timestamp is distant from the last interaction, the model recommends items reflecting long-term interests, *e.g.*, '*RC helicopters*', which had been frequently purchased in the past. These results demonstrate GRUT's ability to adapt recommendations based on inference timestamp, unlike existing generative recommendation models that produce identical predictions regardless of when inference occurs.

D.2 Effect of Trend-aware Inference

Table 12 illustrates how GRUT benefits from the trend score s_{trend} to better capture user preference. The user had recently purchased the '*LEGO Sorting Systems*', so various toy-related products appear as top recommendations when $\lambda = 0$ in Eq. (10). Considering temporal trends during inference, the ranking of trending items '*Rainbow Loom*' was elevated, resulting in recommendations that closely aligned with the user preferences. This demonstrates that Trend-aware Inference enables the model to combine time-sensitive trends with the user's intrinsic preference, producing more accurate and timely recommendations. Furthermore, the ability to control the influence of the s_{trend} based on user needs highlights the practical advantage of the proposed method in terms of *controllability*.