TEG-Rec: Scene-driven Thought-Evolution Generative Recommender

1st Chun Xu
Donghua University
School of Information and Intelligent Science
Shanghai, China
qiancijun@gmail.com

2nd Zhaohui Zhang*
Donghua University
School of Information and Intelligent Science
Shanghai, China
zhzhang@dhu.edu.cn

***Abstract*—Recommender systems are evolving towards generative methods. However, they often overlook the fundamental decision-making factors behind user interaction and treat the complex decision-making process as a series of fixed identifiers. To overcome this limitation, we propose TEG-Rec (Thought-Evolution Generative Recommender), an innovative framework that integrates large language models (LLMs) and hypergraph reasoning into the generation-based recommendation process. Specifically, TEG-Rec first uses LLMs to extract the potential "thought scenarios" related to the items, thereby bridging the semantic gap between item attributes and user intentions. Next, a thought evolution hypergraph is constructed to represent the dynamic transformation of these user intentions, and hypergraph neural networks (HGNNs) are used to generate structured thought embeddings. These embeddings as continuous prompts in the Transformer decoder, guiding the autoregressive generation of item's semantic IDs. Comprehensive experiments conducted on real datasets show that TEG-Rec can effectively capture high-order collaborative information and outperform the existing best benchmark models in recommendation performance.**

***Index Terms*—Generative recommendation, Thought enhancement, Large language model, Graph neural network**

I. Introduction

Generative retrieval has redefined recommendation as a sequence-to-sequence generation task [1], utilizing semantic IDs(e.g., via RQ-VAE [2]) to instead original item IDs. However, despite the efficacy of Transformer-based architectures [3]. Most current methods are unable to clearly capture the constantly changing intentions behind user behaviors, thereby limiting the effectiveness of recommendations. From this, we have identified the key flaws of the two existing methods. Capturing subtle temporal dynamics is a fundamental challenge across various domains [4], [5].

First, intent ambiguity cannot be resolved by current methods. Conventional methods, such as BPR-MF [6] or SASRec [7], rely on static snapshots or ID-based embeddings, which record a user's purchases but fail to capture their motivations. Despite the introduction of semantic indexing, even recent generative baselines (e.g., TIGER [8]) treat user intents as flat, implicit vectors. The fact that a single item may originate from motivationally divergent paths is ignored by this cursory modeling. A "hoodie" could suggest a "camping scenario" (leading to tents) or a "fishing scenario" (leading to rods), as shown in Fig. 1. Models find it difficult to distinguish between these contexts without explicit reasoning, which leads to erroneous correlations. Some other works [9] directly incorporate the entire knowledge base of the LLM to assist in the recommendation process.

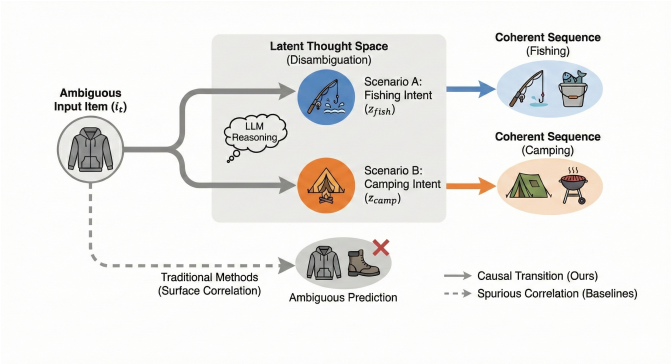

Fig. 1. A single item (e.g., a hoodie) is associated with multifaceted motives. While conventional methods (dashed lines) rely on superficial correlations, TEG-Rec (solid lines) disambiguates context via a latent thought space (e.g., distinguishing "fishing" from "camping"), generating logically coherent sequences.

Second, the high-order structure between items is disregarded by the linear sequence model. While hypergraph models (e.g., DHCN [10]) and graph neural networks (e.g., LightGCN [11], NGCF [12]) try to capture the collaborative signals, they frequently fall short of deep semantic reasoning. Sequence models, on the other hand, ignore the global topological framework that links various reasoning scenarios and instead treat history as an isolated chain-like structure [13]. The model is unable to use collaborative intelligence to inform individual predictions because of this fragmentation.

To bridge these gaps, we proposed the TEG-Rec. Unlike those algorithms based on the original item sequence, TEG-Rec explicitly models the evolution of users' thoughts. We use large language models (LLMs) [14] to extract potential conceptual scenarios from the items (for example, rephrasing "laptop" as "workstation"), and construct a thought evolution hypergraph to depict the global process of these intentions. By employing hypergraph neural networks, through the mes-

*Corresponding author.

sage propagation mechanism, obtaining structured thought embeddings. These embeddings can simultaneously encode semantic content and evolution trajectories. The structured thought embeddings act as continuous prompts, guiding the decoder to generate logically coherent sequences of items based on the guidance of the transformer. The code repository of our experiment https://github.com/qiancijun/TEG-Rec

The principal contributions of this work are delineated as follows:

- We have proposed an innovative approach, leveraging large language models (LLMs) to clearly express the underlying intentions of users and constructing thought evolution diagrams to learn the patterns of users' thought evolution.
- We combine the large language model with the structural representation of the hypergraph neural network (HGNN) to generate recommendations for the next item.
- Extensive tests on three real-world datasets show that TEG-Rec significantly outperforms current state-of-the-art graph-based and generative baseline methods.

II. Methods

This section introduces TEG-Rec, which is a comprehensive framework designed to enhance generative recommendation systems by integrating the development process of user intentions. As shown in Figure 2, this framework consists of three interrelated components:

- **Thought Evolution Modeling**: Use LLMs as reasoning tools to extract specific thinking scenarios (such as "seeking a smooth gaming experience") from item metadata, and use hypergraph neural networks to generate high-order node embeddings for semantic thought nodes.
- **Item Sequence Modeling**: Using the RQ-VAE model, the textual features of the items are encoded into semantic IDs. These semantic IDs are generated as semantic embeddings through the Transformer Encoder.
- **Thought-Guided Generative Retrieval**: The recommendation task is formulated as a sequence-to-sequence generation problem. The learned thought embeddings, within a Transformer-based architecture, act as consecutive prompts, guiding the autoregressive generation process of the hierarchical item index.

Formally, given a user interaction history S_u, the goal is to predict the subsequent item i_{t+1} by maximizing the conditional probability $P(i_{t+1} \mid S_u, \mathcal{G}_{thought})$, where $\mathcal{G}_{thought}$ is the node embedding after the process of propagation learning.

A. Semantic Initialization

We have designed two parallel-executable semantic processes to initialize the manifestation of thoughts and items: Latent Thought Induction and Item Hierarchical Quantization.

1) Latent Thought Induction: The traditional approaches based on user behavior sequences hard to fully capture the user intentions (i.e., "reasons") that trigger purchase behaviors. To gain this semantic discrepancy, we utilize LLMs to extract potential cognitive scenarios from the original itemfeatures. Specifically, for each item i, we concatenate metadata fields (including title, attributes, and category) to build a text representation x_i. Subsequently, we formulate a prompt $\Phi(\cdot)$ to guide the LLM to act as a "user intention analysis expert" [15], responsible for inferring concise, single-sentence explanations for the reasons that drive users' interactions with the item as shown in Fig. 3.

$$T_i = \text{LLM}\left(\Phi(x_i)\right) \tag{1}$$

Here, T_i represents the possible mental intentions of the user when interacting with this items (for example, "finding a high-performance laptop for gaming"). This approach explicitly converts objective attributes into subjective user intents. To integrate T_i into our Thought Evolution Hypergraph Learning, we use the pre-trained Sentence-T5 to encode it into a semantic vector:

$$\mathbf{h}_i^{(0)} = \text{Sentence-T5}(T_i) \tag{2}$$

2) Item Hierarchical Quantization: We adopted the implementation of TIGER and used the residual quantized variational autoencoder (RQ-VAE) to represent the item as a hierarchical tuple of tokens. Let $\mathbf{v}_i$ denote the embedding of the i-th item. The encoder maps $\mathbf{v}_i$ to a latent vector $\mathbf{z}_i$, which is recursively approximated in D iterations using the shared code library $\mathcal{C}$. At depth d, given the residual $\mathbf{r}_{d-1}$ (with $\mathbf{r}_0 = \mathbf{z}_i$), we identify the discrete code $c_{i,d}$ that best approximates the residual:

$$c_{i,d} = \arg\min_k \|\mathbf{r}_{d-1} - \mathbf{e}_k\|_2^2, \quad \mathbf{r}_d = \mathbf{r}_{d-1} - \mathbf{e}_{c_{i,d}} \tag{3}$$

After D steps, $\mathbf{z}_i$ is approximated as $\hat{\mathbf{z}}_i = \sum_{d=1}^{D} \mathbf{e}_{c_{i,d}}$. The decoder then reconstructs the item embedding $\hat{\mathbf{v}}_i$. The training minimizes the reconstruction and commitment loss:

$$\mathcal{L}_{RQ} = \|\mathbf{v}_i - \hat{\mathbf{v}}_i\|_2^2 + \beta \sum_{d=1}^{D} \|\,\text{sg}[\mathbf{r}_{d-1}] - \mathbf{e}_{c_{i,d}}\|_2^2 \tag{4}$$

Post-training, the RQ-VAE is frozen, and each item i is represented by the discrete sequence $C_i = (c_{i,1}, \ldots, c_{i,D})$, serving as the semantic target for generation.

B. Thought Evolution Hypergraph Learning

The standard graph can only depict many-to-one or one-to-many relationships between nodes. However, in certain scenarios, the relationships between thoughts are often many-to-mant high-order relationships. The scene can also be regarded as a collection of multiple sets of thoughts. To capture this higher-order relationship, we model the thinking space as a hypergraph.

Thought Evolution Hypergraph Construction. We define the Thought Evolution Hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, which $\mathcal{V}$ represents unique thought of each item. Unlike pairwise edges, a hyperedge $\epsilon \in \mathcal{E}$ connects a subset of nodes to encapsulate group-level relationships. For each user session S, the associated collection of thought scenarios $\mathcal{T}_S$ is represented as a

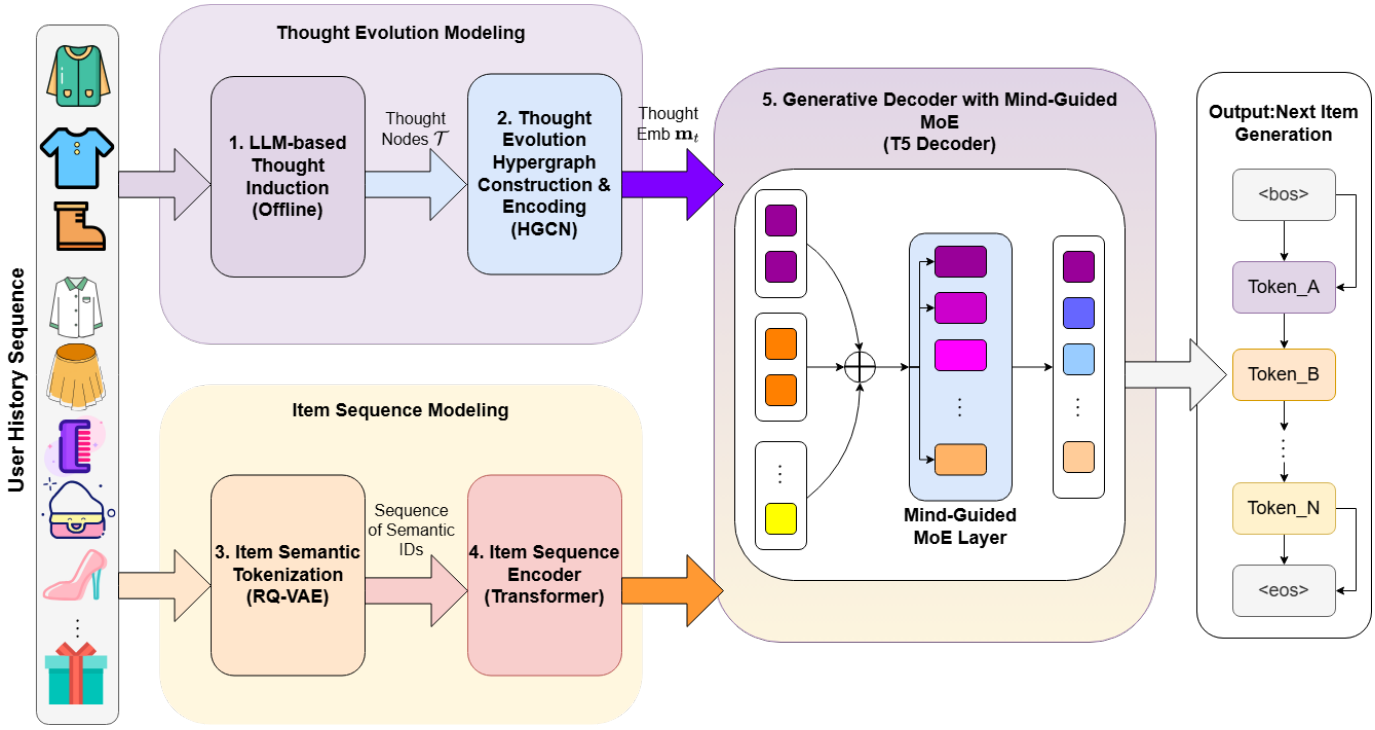

Fig. 2. The TEG-Rec framework. It features a dual-stream architecture: (1) Item Sequence Modeling (bottom) encodes semantic IDs via Transformer; (2) Thought Evolution Modeling (top) utilizes a hypergraph on LLM-induced nodes to capture intent dynamics. A Mind-Guided MoE Decoder finally fuses these representations to generate the next item.

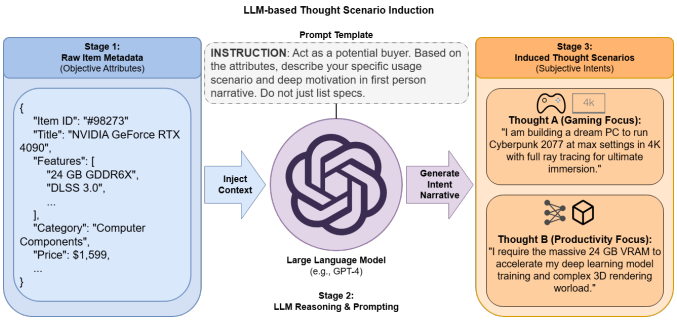

Fig. 3. Latent Thought Generation. Raw item attributes are input into an LLM with specific prompts to synthesize narrative thought scenarios that articulate distinct purchase motives (e.g., gaming vs. productivity).

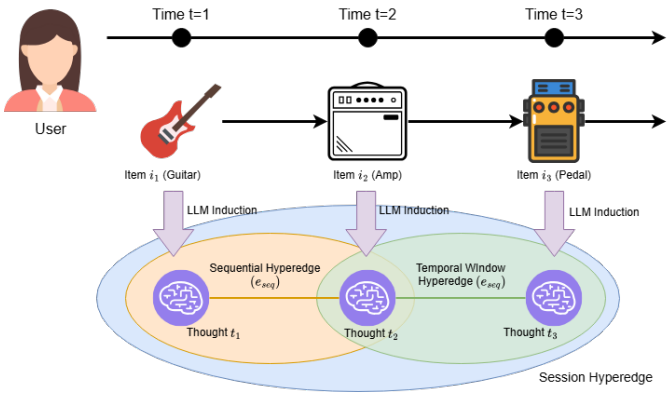

Fig. 4. Construction of the Thought Evolution Hypergraph. Sequences of user interactions are encoded into latent thought nodes through a large language model (LLM). Subsequently, construct hyperedges at three hierarchical levels—Sequential, Temporal Window, and Global Session—to effectively represent the dynamic and evolving dependencies.

single hyperedge, for maintaining the contextual coherence of the data, we sequential hyperedge and temporal hyperedge like Fig. 4). The resulting topology is described by an incidence matrix $\mathbf{U} \in \{0,1\}^{|\mathcal{V}|\times|\mathcal{E}|}$. Utilizing $\mathbf{U}$, we construct the diagonal degree matrices for nodes and hyperedges, denoted as $\mathbf{\Lambda}_\mathcal{V}$ and $\mathbf{\Lambda}_\mathcal{E}$, respectively. These matrices are defined such that $\mathbf{\Lambda}_\mathcal{V}(i,i) = \sum_\epsilon \mathbf{U}_{i,\epsilon}$ and $\mathbf{\Lambda}_\mathcal{E}(\epsilon,\epsilon) = \sum_i \mathbf{U}_{i,\epsilon}$.

Structure-Aware Propagation. To reduce the risk of over-smoothing, we are inspired by LightGCN [11] and utilize a linearized variant of the HGCN [16]. Let $\mathbf{H}^{(0)}$ represent the initial embeddings. The propagation mechanism is defined by the following rule:

$$\mathbf{H}^{(l+1)} = \mathbf{\Lambda}_\mathcal{V}^{-1/2}\mathbf{U}\mathbf{\Lambda}_\mathcal{E}^{-1}\mathbf{U}^T\mathbf{\Lambda}_\mathcal{V}^{-1/2}\mathbf{H}^{(l)} \tag{5}$$

After stacking L layers,We obtain the final embeddings of the thought nodes through average pooling: $\mathbf{m}_t = \frac{1}{L+1}\sum_{l=0}^{L}\mathbf{H}_t^{(l)}$. This resulting embedding $\mathbf{m}_t$ encapsulates both intrinsic thought semantics and global high-order co-occurrence patterns.

C. Thought-Guided Generative Retrieval

We reformulate recommendation as a thought-guided sequence generation task. Unlike standard methods feeding only item IDs, we construct a hybrid input sequence interleaving continuous thought embeddings $\mathbf{m}_t$ (the "cause") with discrete item codes $C_i = (c_{t,1}, \ldots, c_{t,D})$ (the "effect"). The input embedding sequence $\mathbf{E}_{in}$ is constructed as:

$$\mathbf{E}_{in} = [\mathbf{e}_{user}, \underbrace{\mathbf{m}_1, \mathbf{e}(c_{1,1}), ..., \mathbf{e}(c_{1,D})}_{\text{Step 1}}, ..., \underbrace{\mathbf{m}_k, \mathbf{e}(c_{k,1}), ..., \mathbf{e}(c_{k,D})}_{\text{Step k}}] \quad (6)$$

where $\mathbf{m}_t$ acts as a "soft prompt" to condition the autoregressive generation of item tokens.

Fixed-Routing MoE. To reconstruct hierarchical semantics, we introduce a *Fixed-Routing Mixture-of-Experts (MoE)* [17], [18] in the decoder (see Fig. 5). Diverging from shared FFNs, we allocate $D+1$ specialized experts $\{\mathcal{E}_{mind}, \mathcal{E}_1, \ldots, \mathcal{E}_D\}$ to process specific token types. Instead of dynamic routing, we employ a deterministic position-based strategy. Let $p_k = k \pmod{D+1}$ denote the intra-segment position at step k. The routing logic $\mathbf{h}'_k = \mathcal{E}_{p_k}(\mathbf{h}_k)$ ensures that $\mathcal{E}_0$ exclusively processes thought embeddings, while $\mathcal{E}_j$ handles item tokens at level j. The model is trained end-to-end by minimizing the Cross-Entropy Loss:

$$\mathcal{L} = -\sum_{(S_u,i)\in\mathcal{D}} \sum_{j=1}^{D} \log P(c_{i,j} \mid c_{i,<j}, \mathbf{m}_t, S_u) \quad (7)$$

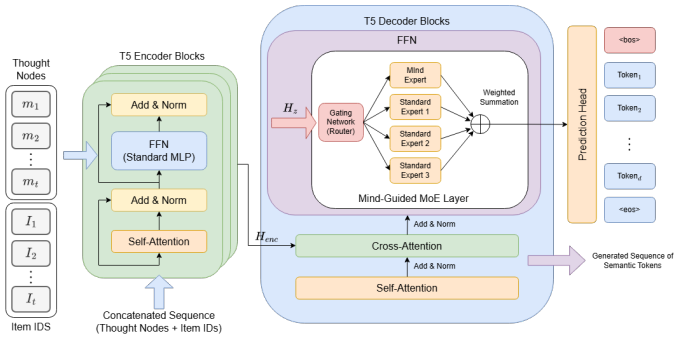

Fig. 5. Generative Decoding Architecture with Mind-Guided MoE. The decoder employs positional activation to select between dedicated Mind Experts and standard experts, synthesizing the semantic token sequence under explicit thought guidance.

III. Experiments

In this section, we conduct comparative experiments with existing methods on three datasets. We also perform ablation experiments on each module to analyze the effectiveness of TEG-Rec.

A. Experimental Settings

We conducted the evaluation on three real-world datasets from the Amazon 2023 benchmark test set (Scientific, Instruments, Video Games) [19]. These datasets show significant differences in terms of the domain and data sparsity. Following standard preprocessing protocols, we apply 5-core filtering to exclude items with fewer than 5 interactions and retain the most recent 20 interactions for each user sequence. The detailed statistics of these datasets are summarized in Table I.

TABLE I
STATUSTICS OF THE DATASET

Dataset	Users	Items	Interactions	Avg Len
Instrument	57,439	24,587	511,836	9
Scientific	50,985	25,848	412,947	8
Video Games	94,762	25,612	814,586	8

Baseline Methods. We evaluated the performance of TEG-Rec and compared it with a series of representative methods. **(1) Sequential Models: GRU4Rec** [20] and **Caser** [21] use RNNs and CNNs, respectively, to capture sequential patterns. **HGN** [22] adds gating mechanisms to select important features. Moving to Transformers, **SASRec** [7] uses unidirectional self-attention, while **BERT4Rec** [23] uses bidirectional self-attention to model long-range dependencies. **FDSA** [24] further considers feature-level attention. **FMLP-Rec** [25] replaces GNNs with an all-MLP architecture using frequency-domain filters.
(2) Graph Models: SR-GNN [26] applies GNNs on session graphs to capture item transitions. **GC-SAN** [27] improves this by combining GNNs with self-attention to learn both local and global contexts. **LightGCN** [11] simplifies GCNs by removing non-linearities and keeping only linear aggregation.
(3) Generative Models: S^3-Rec [28] uses mutual information maximization for self-supervised learning. **HSTU** [29] is a high-efficiency Transformer designed for large-scale streaming. **TIGER** [8] is a generative method that quantizes items into semantic IDs and uses a Transformer to predict the next ID directly.

We use the standard **Leave-One-Out** strategy: the last item in each sequence is used for testing, the second-to-last for validation, and the rest for training.

B. Implementation Details

We implement TEG-Rec and all baselines using PyTorch and the HuggingFace Transformers library [30]. All experiments are conducted on a Linux server equipped with a single Tesla T4 (12G) GPU.

Item Tokenization: Following TIGER, we employ Sentence-T5 [31] to encode item textual metadata into semantic vectors. The RQ-VAE utilizes a hierarchical codebook with 3 levels (size 256 each), plus an additional level for collision resolution. The MLP dimensions in the encoder/decoder are set to $[2048, 1024, 512, 256, 128]$. We train the RQ-VAE using the Adagrad optimizer (learning rate $lr = 0.001$, batch size 2048) for 10,000 iterations.

Thought Graph Model: For initialization, we utilize Qwen3-14b [32] to generate explicit textual thought scenarios, which are encoded by Sentence-T5 into 768-dimensional initial nodes to ensure alignment with the item space. The Thought Evolution Hypergraph is built with three edge types:

TABLE II
PERFORMANCE COMPARISON OF DIFFERENT METHODS ON INSTRUMENT, SCIENTIFIC, AND GAME DATASETS. THE BEST RESULTS ARE **BOLDFACED**, AND THE SECOND-BEST RESULTS ARE UNDERLINED.

Methods	Instrument				Scientific				Game			
	Recall@5	Recall@10	NDCG@5	NDCG@10	Recall@5	Recall@10	NDCG@5	NDCG@10	Recall@5	Recall@10	NDCG@5	NDCG@10
Caser	0.0238	0.0389	0.0154	0.0194	0.0162	0.0255	0.0099	0.0135	0.0332	0.0556	0.0212	0.0278
HGN	0.0318	0.0520	0.0205	0.0262	0.0208	0.0354	0.0133	0.0173	0.0426	0.0684	0.0268	0.0359
SR-GNN	0.0097	0.0149	0.0062	0.0078	0.0151	0.0246	0.0097	0.0127	0.0289	0.0470	0.0185	0.0243
LightGCN	0.0261	0.0416	0.0157	0.0208	0.0206	0.0419	0.0173	0.0192	0.0411	0.0727	0.0306	0.0409
GC-SAN	0.0232	0.0358	0.0136	0.0176	0.0255	0.0330	0.0146	0.0203	0.0526	0.0859	0.0323	0.0430
GRU4Rec	0.0327	0.0498	0.0211	0.0269	0.0205	0.0335	0.0132	0.0176	0.0496	0.0802	0.0323	0.0413
SASRec	0.0333	0.0523	0.0213	0.0274	0.0259	0.0412	0.0150	0.0199	0.0535	0.0847	0.0331	0.0438
BERT4Rec	0.0310	0.0492	0.0192	0.0255	0.0183	0.0299	0.0116	0.0156	0.0463	0.0735	0.0296	0.0460
FMLP-Rec	0.0339	0.0536	0.0218	0.0282	0.0269	0.0422	0.0155	0.0204	0.0528	0.0857	0.0338	0.0444
FDSA	0.0347	0.0545	0.0230	0.0293	0.0262	0.0421	0.0169	0.0213	0.0544	0.0852	0.0361	0.0448
HSTU	0.0343	0.0577	0.0191	0.0271	0.0271	0.0429	0.0147	0.0198	0.0578	0.0903	0.0334	0.0442
S^3-Rec	0.0317	0.0496	0.0199	0.0257	0.0263	0.0418	0.0171	0.0219	0.0485	0.0769	0.0315	0.0406
TIGER	0.0373	0.0567	0.0241	0.0309	0.0267	0.0419	0.0177	0.0223	0.0562	0.0868	0.0369	0.0464
Ours	**0.0405**	**0.0613**	**0.0268**	**0.0341**	**0.0302**	**0.0464**	**0.0206**	**0.0251**	**0.0615**	**0.0945**	**0.0401**	**0.0507**
Improve	8.58%	6.24%	11.20%	10.36%	11.44%	7.91%	16.38%	12.56%	6.40%	4.65%	8.67%	9.27%

sequential, temporal window ($w = 3$), and global session edges. We set the hypergraph convolution layers to $L = 3$.

Generative Recommender: We adopt a T5-based [33] backbone with an output dimension of 128, hidden size of 256, and 4 attention heads. Both the encoder and decoder consist of 7 layers, with the final 4 layers of the decoder replaced by our Fixed-Routing MoE layers. We use a cosine learning rate scheduler. The batch size is set to 256 for training and 64 for testing.

C. Overall Performance

Table II establishes TEG-Rec as the consistent leader across all datasets, validating the necessity of explicit thought modeling.

Better than Discriminative Baselines Conventional approaches encounter semantic and structural bottlenecks. Sequential or pairwise dependencies are captured by transformer-based models (e.g., [7]) and graph-based techniques (e.g., [27]), but they are still limited by ID-based sparsity and limited structural order. By combining LLM-derived semantics with hypergraph topology, TEG-Rec overcomes these constraints. It achieves significant improvements by capturing high-order intent dependencies that pairwise graphs overlook thanks to this synergy.

Advantage over Generative Baselines Although the generative retrieval paradigm is validated by TIGER and HSTU, they usually encode user intents as implicit, static latent vectors. TEG-Rec sets itself apart by explicitly modeling the reasoning process. Our model significantly outperforms the strongest generative baseline TIGER by using evolved thought scenarios to disambiguate user intent, demonstrating the importance of resolving intent ambiguity for accurate next-item generation.

D. Ablation Study

To verify the contribution of each component, we compare TEG-Rec with three variants: **w/o Thought** (removes LLM scenarios), **w/o HGCN** (replaces hypergraph convolution with simple embedding lookup), and **w/o MoE** (replaces the MoE module with a standard Feed-Forward Network). Figure 6 illustrates the training curves. The results on the test set are shown in the tab. III.

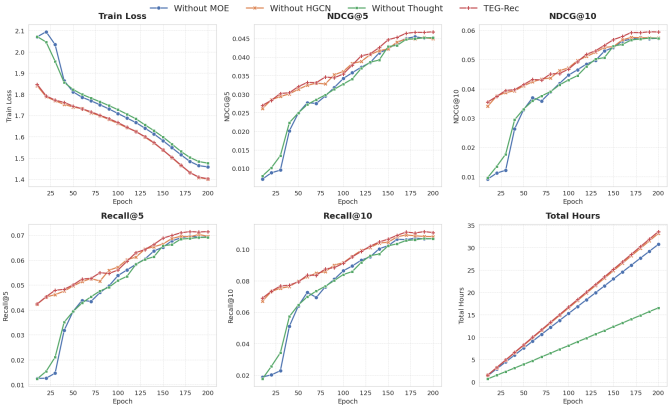

Fig. 6. Ablation study curves during the training process. We compare TEG-Rec with three variants across Train Loss, NDCG@{5,10}, Recall@{5,10}, and Training Time (Total Hours).

TABLE III
PERFORMANCE OF EACH COMPONENT WITH RESPECT TO TEG-REC ON VIDEO GAMES

Methods	Recall@5	Recall@10	NDCG@5	NDCG@10
TEG-Rec	0.0615	0.0945	0.0401	0.0507
w/o Thought	0.0562	0.0868	0.0369	0.0464
w/o HGCN	0.0609	0.0940	0.0393	0.0499
w/o MoE	0.0600	0.0929	0.0393	0.0499

Latent Thoughts' Impact The **w/o Thought** variant (green line) exhibits the simplest convergence and the biggest performance decline. This demonstrates that using sparse ID sequences alone is inadequate. Prediction accuracy is limited by the model's inability to capture the complex motivations underlying user behaviors in the absence of explicit thought modeling.

The necessity of mind-guided MoE The **w/o MoE** variant (blue line) performs poorly, producing outcomes that are

comparable to the baseline without thoughts. This implies that fusing heterogeneous modalities (continuous thought embeddings vs. discrete item tokens) is difficult for a standard shared FFN. To avoid parameter interference and guarantee that ideas successfully direct the generation process, the MoE structure is required.

The Hypergraph Structure's Advantage Though it still lags behind the full model, the **w/o HGCN** variant (orange line) performs better than the other two. This suggests that the Thought Evolution Hypergraph further improves performance by capturing high-order structural dependencies across sessions, even though semantic content offers a solid foundation.

Efficiency Analysis The "Total Hours" plot illustrates how adding thought modeling raises training costs. However, we view this computational overhead as a reasonable trade-off for high-precision recommendation given the significant accuracy gains (e.g., in Recall@10).

IV. Conclusion

In this work, we design TEG-Rec, a framework that explicitly models user thoughts to address intent ambiguity. Our model links item attributes to user motives and captures the evolution of intents over time by integrating LLM-generated scenarios with a Thought Evolution Hypergraph. To make sure the generated items correspond with these evolved thoughts, we also implemented a Mind-Guided MoE decoder. TEG-Rec consistently outperforms top baselines like TIGER in experiments conducted on three real-world datasets, proving that incorporating explicit reasoning greatly increases recommendation accuracy in challenging situations.

V. acknowledgments

This work is Sponsored by the Natural Science Foundation of Shanghai (No. 24ZR1403300) and the National Natural Science Foundation of China (No. 62477006).

References

[1] L. Wu, Z. Zheng, Z. Qiu, H. Wang, H. Gu, T. Shen, C. Qin, C. Zhu, H. Zhu, Q. Liu, H. Xiong, and E. Chen, "A survey on large language models for recommendation," 2024.

[2] D. Lee, C. Kim, S. Kim, M. Cho, and W.-S. Han, "Autoregressive image generation using residual quantization," 2022.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[4] K. Yu, Z. Zhang, C. Hu, and J. Luo, "Sofp: Capturing subtle facial dynamics with symmetric optical flow perception for micro-expression recognition," *Pattern Recognition*, p. 113199, 2026.

[5] K. Yu, W. Ma, S. Chen, X. Wu, Y. Wang, and T. Zhou, "Cpcr: Crop pests cross-modal retrieval with prior instruction via multimodal large language model," *Computers and Electronics in Agriculture*, vol. 245, p. 111546, 2026.

[6] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme, "Bpr: Bayesian personalized ranking from implicit feedback," 2012.

[7] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," 2018.

[8] S. Rajput, N. Mehta, A. Singh, R. H. Keshavan, T. Vu, L. Heldt, L. Hong, Y. Tay, V. Q. Tran, J. Samost, M. Kula, E. H. Chi, and M. Sathiamoorthy, "Recommender systems with generative retrieval," 2023.

[9] Y. Hou, J. Zhang, Z. Lin, H. Lu, R. Xie, J. McAuley, and W. X. Zhao, "Large language models are zero-shot rankers for recommender systems," 2024.

[10] X. Xia, H. Yin, J. Yu, Q. Wang, L. Cui, and X. Zhang, "Self-supervised hypergraph convolutional networks for session-based recommendation," 2022.

[11] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," 2020.

[12] X. Wang, X. He, M. Wang, F. Feng, and T.-S. Chua, "Neural graph collaborative filtering," in *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '19. ACM, Jul. 2019, p. 165–174.

[13] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," 2016.

[14] W. X. Zhao *et al.*, "A survey of large language models," 2025.

[15] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," 2023.

[16] S. Bai, F. Zhang, and P. H. S. Torr, "Hypergraph convolution and hypergraph attention," 2020.

[17] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," 2017.

[18] D. Dai, C. Deng, C. Zhao, R. X. Xu, H. Gao, D. Chen, J. Li, W. Zeng, X. Yu, Y. Wu, Z. Xie, Y. K. Li, P. Huang, F. Luo, C. Ruan, Z. Sui, and W. Liang, "Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models," 2024.

[19] Y. Hou, J. Li, Z. He, A. Yan, X. Chen, and J. McAuley, "Bridging language and items for retrieval and recommendation," *arXiv preprint arXiv:2403.03952*, 2024.

[20] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," 2016.

[21] J. Tang and K. Wang, "Personalized top-n sequential recommendation via convolutional sequence embedding," 2018.

[22] C. Ma, P. Kang, and X. Liu, "Hierarchical gating networks for sequential recommendation," 2019.

[23] F. Sun, J. Liu, J. Wu, C. Pei, X. Lin, W. Ou, and P. Jiang, "Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer," 2019.

[24] T. Zhang, P. Zhao, Y. Liu, V. S. Sheng, J. Xu, D. Wang, G. Liu, and X. Zhou, "Feature-level deeper self-attention network for sequential recommendation," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, ser. IJCAI'19. AAAI Press, 2019, p. 4320–4326.

[25] K. Zhou, H. Yu, W. X. Zhao, and J.-R. Wen, "Filter-enhanced mlp is all you need for sequential recommendation," in *Proceedings of the ACM Web Conference 2022*. ACM, Apr. 2022, p. 2388–2399.

[26] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, "Session-based recommendation with graph neural networks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, p. 346–353, Jul. 2019.

[27] C. Xu, P. Zhao, Y. Liu, V. S. Sheng, J. Xu, F. Zhuang, J. Fang, and X. Zhou, "Graph contextualized self-attention network for session-based recommendation," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, ser. IJCAI'19. AAAI Press, 2019, p. 3940–3946.

[28] K. Zhou, H. Wang, W. X. Zhao, Y. Zhu, S. Wang, F. Zhang, Z. Wang, and J.-R. Wen, "S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization," in *Proceedings of the 29th ACM International Conference on Information; Knowledge Management*, ser. CIKM '20. ACM, Oct. 2020, p. 1893–1902.

[29] J. Zhai, L. Liao, X. Liu, Y. Wang, R. Li, X. Cao, L. Gao, Z. Gong, F. Gu, M. He, Y. Lu, and Y. Shi, "Actions speak louder than words: Trillion-parameter sequential transducers for generative recommendations," 2024.

[30] T. Wolf *et al.*, "Transformers: State-of-the-art natural language processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45.

[31] J. Ni, G. H. Ábrego, N. Constant, J. Ma, K. B. Hall, D. Cer, and Y. Yang, "Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models," 2021.

[32] A. Yang *et al.*, "Qwen3 technical report," 2025.

[33] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," 2023.