AGRec: Adapting Autoregressive Decoders with Graph Reasoning for LLM-based Sequential Recommendation

Xinfeng Wang[†], **Jin Cui**[†], **Fumiyo Fukumoto**[‡], and **Yoshimi Suzuki**[‡]
[†]Graduate School of Engineering
[‡]Interdisciplinary Graduate School
University of Yamanashi, Kofu, Japan
{g22dtsa7, g22dtsa5, fukumoto, ysuzuki}@yamanashi.ac.jp

Abstract

Autoregressive decoders in large language models (LLMs) excel at capturing users' sequential behaviors for generative recommendations. However, they inherently struggle to leverage graph-structured user-item interactions, which are widely recognized as beneficial. This paper presents AGRec, adapting LLMs' decoders with graph reasoning for recommendation. We reveal that LLMs and graph neural networks (GNNs) manifest complementary strengths in distinct user domains. Building on this, we augment the decoding logits of LLMs with an auxiliary GNN model to optimize token generation. Moreover, we introduce a rankable finite state machine to tackle two challenges: (1) adjusting autoregressive generation with discriminative decoders that directly predict user-item similarity, and (2) token homogeneity, where LLMs often generate items with similar prefix tokens, narrowing the scope of beam search. Our AGRec outperforms state-of-the-art models in sequential recommendations. Our source code and data are available online[1].

1 Introduction

Large language models (LLMs) have demonstrated powerful reasoning capabilities in converting recommendation tasks into language generation tasks (Zhang et al., 2024b,a; Wei et al., 2024; Lu et al., 2024; Yu et al., 2024; Kannen et al., 2024). Recent methods exploit enhanced item tokenization (e.g., <a_35><b_11><c_23><d_77> for a basketball and <a_35><b_11><c_23><d_24> for a soccer ball in the top of Fig. 1) (Rajput et al., 2023; Zheng et al., 2024; Wang et al., 2024b), to substitute numerical IDs (e.g., "item_1234") (Li et al., 2023b) and textual IDs (e.g., movie titles) (Tan et al., 2024a). However, an inherent problem with LLMs is their difficulty in interpreting graph-constructed high-order interactions, which are widely acknowledged as valuable for recommendation tasks (Wang et al., 2022, 2024g,f; Zhao et al., 2022; Yu et al., 2022). The reason is that interaction signals and the pre-trained textual semantics of LLMs are distributed in entirely different feature spaces (Wei et al., 2024). This fundamental mismatch explains why, despite numerous efforts to align graph-structured interactions with textual inference (Wang et al., 2024h; Guo et al., 2024; Huang et al., 2024b), LLMs still face challenges in accurately capturing these signals for recommendation. More recently, several attempts (Wang et al., 2024h; Tan et al., 2024b; Zhu et al., 2024; Huang et al., 2024b) generate massive textual prompts based on user-item interaction edges to train their LLMs. This is less effective, as a node with 100 one-hop neighbors could have up to 10,000 two-hop neighbors, growing exponentially with more hops. Another endeavor is to employ graph neural networks (GNNs) (He et al.,

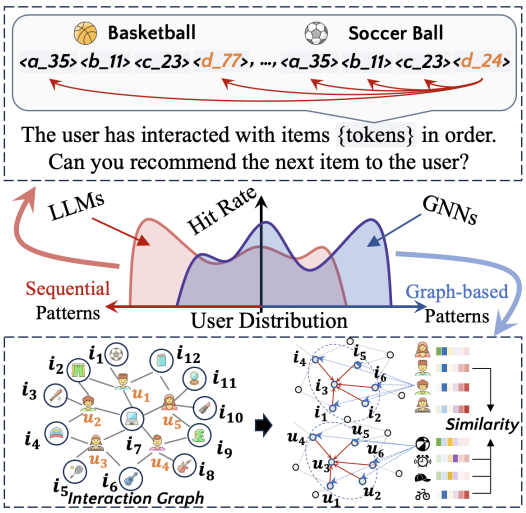


Figure 1: Motivation illustration: LLMs and GNNs both capture collaborative signals (leading to an overlap in terms of hit rate over user distribution), yet each also excels in distinct domains. LLMs excel at capturing user sequential behaviors autoregressively, while GNNs better learn high-order relationships via message-passing.

[1]https://github.com/WangXFng/AGRec

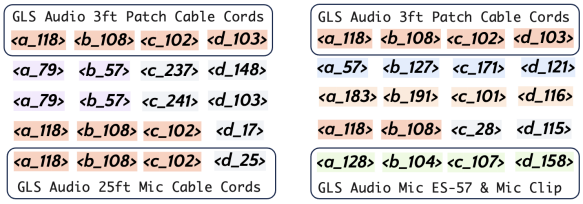
Findings of the Association for Computational Linguistics: ACL 2025, pages 7076–7090
July 27 - August 1, 2025

2020; Yu et al., 2023; Wang et al., 2023) that aggregate node features through interaction edges to assist LLMs in recommendations (Kim et al., 2024; Ma et al., 2024; Wang et al., 2024d). Nevertheless, the autoregressive decoder of LLMs involves token selection from a predefined vocabulary, centered around textual coherence, which inherently differs from GNNs' decoding that relies on user-item representation similarity.

Unlike NLP tasks, training LLMs for recommendation with significantly large vocabularies (where a single token represents each item) is very challenging due to the extremely sparse user-item interactions (less than 0.09% as shown in Table 1). To address the issue, many recent studies have adopted the RQ-VAE module (van den Oord et al., 2017; Zeghidour et al., 2021) for item tokenization. However, LLM decoders still face a challenge: they struggle with the issue of homogeneity (Bao et al., 2024; Lu et al., 2024; Chen et al., 2025). As illustrated in Fig. 2, LLMs often generate items with similar prefixes, narrowing the scope of beam search. This is further exacerbated by the non-uniform distribution of items in the language space (Bao et al., 2024), leading some items to contain tokens with generation probabilities close to 1. Moreover, once a prefix token is incorrect, the model fails to make recommendations, regardless of the subsequent tokens. In contrast, item tokens decoded by GNNs offer greater diversity, enabling LLMs to broaden their perspective in selecting prefix tokens.

In light of the aforementioned discussions, we propose the AGRec model, which adapts LLMs' decoders with graph reasoning for recommendation. Empirically, we found that LLMs and GNNs make effective recommendations for the same user domain, while also demonstrating complementary strengths in specific user domains, as illustrated in Fig. 1. Inspired by recent work (Zheng et al., 2024) that leverages text-free models to generate additional token scores at each decoding step, AGRec employs an auxiliary GNN to directly enhance its decoding logits for token generation, harnessing their complementary strengths. Different from the two common methods: (i) using a graph-based retriever and (ii) incorporating graph-based representations, this approach offers a novel perspective for improving LLM-based recommenders with external graph-structured knowledge.

To address the decoding misalignment and the homogeneity issues, we introduce a novel rankable finite state machine (FSM). The FSM has been widely studied to constructively reformulate text generation for LLMs (Willard and Louf, 2023; Chen et al.; Wang et al., 2024c), by strategically masking undesired tokens during decoding. Specifically, we convert GNN logits into finite states with rating scores, in which only tokens from the top n items are selectable and rankable for LLMs. We reveal that LLMs can intelligently infer the next token, even when the newly generated last token is not the originally highest-ranked.



(a) Tokens Generated By LLMs (b) Tokens Decoded By GNNs

Figure 2: Illustration of token homogeneity. During beam search, LLMs generate item tokens according to contextual coherence, often causing overly similar prefix tokens compared with GNNs.

The main contributions of our work can be summarized as follows: (1) We reveal the complementary strengths of LLMs and GNNs in recommendation tasks and propose AGRec, which augments the token generation of LLMs with an auxiliary GNN; (2) We introduce a novel rankable FSM to address the decoding misalignment between LLMs and GNNs and the homogeneity issue. (3) Extensive experiments demonstrate that AGRec, with LLaMA-1B as its backbone, outperforms state-of-the-art (SOTA) baselines equipped with LLaMA-7B in sequential recommendation tasks.

2 Related work

LLMs have exhibited remarkable reasoning capabilities in recommendations (Huang et al., 2023, 2024a; Li et al., 2023a, 2024; Wang et al., 2024a; Lu et al., 2024; Ji et al., 2024; Du et al., 2024a). Many attempts have leveraged knowledge of LLMs to improve performance (Bao et al., 2023; He et al., 2023; Sanner et al., 2023; Xu et al., 2024), including reranking items using textual data (Yue et al., 2023; Carraro and Bridge, 2024), integrating multimodal data (Geng et al., 2022, 2023; Li et al., 2023b; Wang et al., 2024e) and augmenting textual representations (Harte et al., 2023; Lin et al., 2023; Lei et al., 2024; Viswanathan et al., 2023; Na et al., 2024; Ren et al., 2024). Recently, numerous efforts

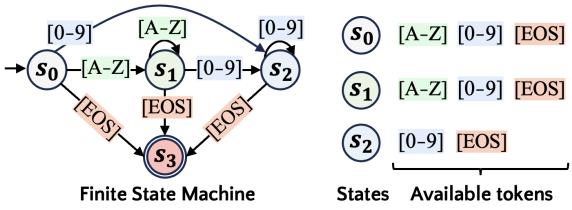


Figure 3: Illustration of the FSM for the regular expression "$([A-Z]*)[0-9]*$". Each state imposes specific constraints on the tokens that LLMs can choose, with all other tokens masked. "[EOS]" represents the early stopping token marking the end of sentences.

have explored tokenization of items for recommendation (Hua et al., 2023; Rajput et al., 2023; Zheng et al., 2024; Wang et al., 2024b).

An increasing number of research has integrated GNNs with LLMs for recommendation, such as aligning interactions with textual inference (Guo et al., 2024; Tan et al., 2024b; Zhu et al., 2024; Huang et al., 2024b), generating text prompts based on interaction edges (Wang et al., 2024h; Tan et al., 2024b; Zhu et al., 2024), encoding node descriptions (Chen et al., 2024; Du et al., 2024b; Damianou et al., 2024), and graph structure augmentation (Wei et al., 2024). Several endeavors employ GNNs (Kim et al., 2024; Wang et al., 2024d) to generate collaborative embeddings for LLMs.

3 Preliminary

3.1 Vector-Quantized Item Tokenization

Tokenizers for LLMs separate numeric IDs into subwords, causing spurious correlations among IDs (Wang et al., 2024d). For example, the IDs "1234" and "3412" are independent but share the same tokens, "12" and "34". Toward this, recent endeavors utilize the residual-quantized variational autoencoder (RQ-VAE) (van den Oord et al., 2017; Zeghidour et al., 2021) to generate item indices (Rajput et al., 2023; Hou et al., 2023; Zheng et al., 2024; Wang et al., 2024b). RQ-VAE represents items using code sequences from learnable shared multi-level codebooks (e.g., four-level codebooks consist of around 1,000 codes). Specifically, this approach first uses pre-trained models to extract item textual embeddings from texts (e.g., descriptions) and then assigns a sequence of the closest codes as item indices, by iteratively minimizing the differences (i.e., residuals) between the textual embeddings and the assigned code embeddings. The RQ-VAE training objectives are (i) reconstructing textual data from the assigned codes and (ii) minimizing residual distances across multiple levels.

3.2 Reformulated Text Generation

The FSM (Sipser, 1996) and trie (Fredkin, 1960) are widely used to guide text generation by masking invalid tokens in task-specific LLMs (Willard and Louf, 2023; Chen et al.; Hua et al., 2023; Zheng et al., 2024; Wang et al., 2024b). For example, the regular expression "$([A-Z]*)[0-9]*$" defines a string pattern (e.g., "$AB12$," "XY," or "789"). As shown in Fig. 3, an FSM constructed from this pattern restricts token selection at each state, masking all invalid options. During decoding, the next state depends on the previous state and the newly generated token, which limits the next valid tokens for LLMs to follow the specified pattern.

4 Approach

Fig. 4 illustrates the overall framework of AGRec, consisting of (a) decoding discriminative GNN logits and (b) recommendation with graph reasoning.

4.1 Decoding Discriminative GNN Logits

The AGRec adopts a pre-trained discriminative model, LightGCN (He et al., 2020) as the auxiliary model due to its simplicity and effectiveness, which estimates the rating score $\mathcal{R}_{u,v}$ of item $v \in \mathcal{V}$ for user $u \in \mathcal{U}$ by an inner-product operation:

$$\mathcal{R}_{u,v} = < \phi(u), \psi(v) >, \tag{1}$$

where $\phi(u)$ and $\psi(v)$ are the embeddings of u and i, respectively. $\phi : \mathcal{U} \rightarrow \mathbb{R}^d$ and $\psi : \mathcal{V} \rightarrow \mathbb{R}^d$ denote the feature aggregations by LightGCN:

$$\begin{aligned} \psi(v)^{(l+1)} &= \sum_{u \in \mathcal{N}_v} \frac{1}{\sqrt{|\mathcal{N}_v|}\sqrt{|\mathcal{N}_u|}} \phi(u)^{(l)}, \\ \phi(u)^{(l+1)} &= \sum_{v \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_v|}} \psi(v)^{(l)}, \end{aligned} \tag{2}$$

where l refers to the LightGCN layer, and $\mathcal{N}_v$ and $\mathcal{N}_u$ denote the neighbors of v and u, respectively.

Building on the logits of LightGCN (i.e., $\mathcal{R}_{u,*}$ of user u), we convert the top n items into code sequences using RQ-VAE (Zheng et al., 2024; Wang et al., 2024b), and construct a rankable FSM based on these code sequences and their scores as shown in Fig. 4 (a). Specifically, we utilize four-level codebooks (i.e., "<*a*>", "<*b*>", "<*c*>", and "<*d*>") with each codebook containing up to 256 tokens, which are assigned as new tokens in the vocabulary (Zheng et al., 2024; Wang et al., 2024b). We define a rankable FSM as a tuple $(Q, \Sigma, \delta, s_0, s_f)$, where:

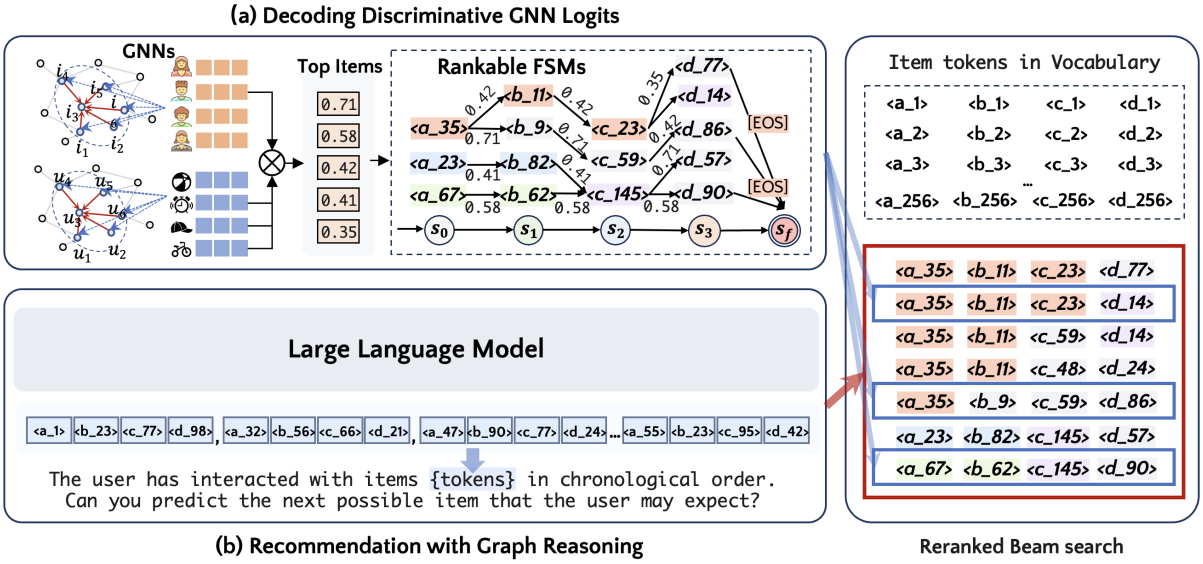


Figure 4: The framework of AGRec, consisting of (a) decoding discriminative GNN logits and (b) recommendation with graph reasoning. In (a), we transforms GNNs' discriminative logits into a probability distribution with rankable FSMs over the vocabulary, aligning them with LLM decoding. In (b), we integrates the FSMs into LLM decoding, allowing AGRec to go beyond relying solely on textual coherence for recommendation.

- Q refers to a finite set of states,
- Σ represents the set of available tokens (e.g., "<a_35>" and "<b_11>"), each associated with a score for every state,
- $\delta : Q \times \Sigma \rightarrow Q$ is the transition function,
- $s_0 \in Q$ represents the initial state, and
- $s_f \in Q$ denotes the accept state, which is reached after encountering the stopping token (e.g., "[EOS]").

At each decoding step, the FSM identifies the current state based on the last generated token and the previous state, thereby providing the next available tokens for LLMs. After masking undesired tokens, the FSM employs LightGCN logits to assign scores to tokens in the vocabulary at state $s(y_{\leq t})$, represented as $p_G(y_t|y_{<t})$, where $y_{\leq t}$ denotes the hypothesis with token y selected at step t, and $p_G(\cdot)$ is the discrete probability distribution of available tokens. In this manner, the FSM transforms the discriminative decoding logits of GNNs for each user into a probability distribution over the vocabulary, aligning them with the LLM's decoding process.

4.2 Recommendation with Graph Reasoning

As illustrated in Fig. 4 (b), after embedding code sequences of items into the prompt template, we tokenize and feed them into LLMs to autoregressively generate the code sequence of the target item. Given the input tokens denoted as $X = [x_1, ..., x_{|X|}]$, the goal of AGRec is to generate the code sequence $Z = [z_1, ..., z_{|Z|}]$. The probability of the target item is defined by:

$$p(z_1, ..., z_{|V|}) = \prod_{i=1}^{|Z|} p(z_i|z_{<i}, X), \tag{3}$$

where $p(z_i|z_{<i}, X)$ indicates the conditional probability of the token z_i at step i, given the previous tokens $z_{<i}$ and input tokens X.

Building on the finding that LLMs and GNNs can complement each other in distinct domains, we integrate graph reasoning with GNNs into LLM decoders. Drawing inspiration from (Zheng et al., 2024), we enhance LLMs' decoders by refining scores with logits from auxiliary GNNs, guiding them to select tokens from the vocabulary:

$$\hat{p}(z_t|z_{<t}, X) = (1-\alpha)p(z_t|z_{<t}, X) + \alpha p_G(z_t|z_{<t}), \tag{4}$$

where $\alpha \in [0, 1]$ refers to the hyperparameter to balance the logits of LLMs and GNNs. A higher value of α indicates the stronger influence of GNN logits. As such, the AGRec no longer relies on merely textual coherence to make recommendations, thereby significantly preventing LLMs from producing overly similar prefixes.

Reranked Beam Search. Beam search methods provide a robust solution to find the optimal code sequence for generative recommendations (Zheng et al., 2024; Tan et al., 2024a; Wang et al., 2024d,b).

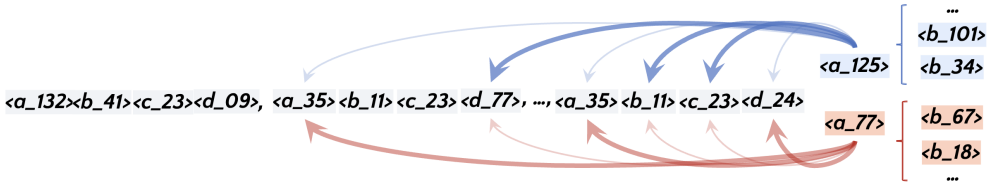


Figure 5: Illustration of autoregressive token generation. Assume that <a_77> is the optimal token for LLMs, while <a_125> ranks highest for GNNs. If an LLM is forced to select <a_125>, its next prediction would dynamically shift to <b_101> and <b_34> instead of <b_67> and <b_18>. This results from the self-attention between the newly generated token (e.g., <a_125>) and previous tokens (e.g., <a_35>), which determines the next token's probability.

We leverage enhanced LLM decoding to rerank tokens during beam search, allowing AGRec to highlight the tokens recommended by the GNNs. The formula to cumulate the beam search score $S(z_{\leq t})$ for the token up to step t is given as follows:

$$S(z_{\leq t}) = S(z_{\leq t-1}) + \log(\hat{p}(z_t|z_{<t}, X)). \quad (5)$$

Remarks. On one hand, LLMs can dynamically adjust logits for the next-token predictions, even if the last generated token was not the highest-ranked for them. This stems from their autoregressive nature, where the probability of the next token is determined by the self-attention between the last token and all previous tokens, as illustrated in Fig. 5. Formally, the hidden representation h_t of the t-th token to predict the (t+1)-th token is a weighted sum of all previous token **Values** (denoted as V): $h_t = \sum_{j=1}^{t} a_{t,j} V_j$, where $a_{t,j}$ depends on the dot product between the **Query** of the last token and the **Keys** of all previous tokens:

$$a_{t,j} = \frac{\exp(q_t k_j^\top / \sqrt{d})}{\sum_{i \leq t} \exp(q_t k_i^\top / \sqrt{d})}. \quad (6)$$

On the other hand, when the last generated token does not match any state defined by the FSMs (i.e., the last token does not align with the top items from the GNNs, indicating that it is purely generated by the LLM logits), the FSMs will not provide available tokens, and AGRec will rely entirely on the LLM logits to continue token generation based on Eq. (4). When both LLMs and GNNs suggest the same tokens, AGRec prioritizes them by combining their likelihoods with Eq. (4). In summary, once the prefix tokens are determined, AGRec automatically selects the subsequent tokens that are recommended by the LLMs, GNNs, or both.

4.3 Model Optimization

To learn the sequential patterns of all codes, AGRec autoregressively generates all tokens for training, including items from historical interactions (Zheng et al., 2024; Wang et al., 2024b). We define the input tokens and the target item tokens as $X = [x_1, \ldots, x_{|X|}]$, and $Z = [z_1, \ldots, z_{|Z|}]$, respectively. The input-label pairs are given as follows:

$$\begin{aligned} X &= [x_1, \ldots, x_{|X|}, z_1, \ldots, z_{|V|-1}], \\ Y &= [x_2, \ldots, x_{|X|}, z_1, \ldots, z_{|V|}]. \end{aligned} \quad (7)$$

We utilize a log-likelihood loss function to optimize the model parameters Θ:

$$\mathcal{L}_\Theta = \frac{1}{|\mathcal{D}|} \sum_{(X,Y)\in\mathcal{D}} \frac{1}{|Y|} \sum_{t=1}^{|Y|} -\log p(y|y_{<t}, X), \quad (8)$$

where $\mathcal{D}$ indicates a set of input-label pairs.

Dataset	#User	#Item	#Interaction	#Avg.	Density
Instruments	24,773	9,923	206,153	8.32	0.084%
Arts	45,142	20,957	390,832	8.66	0.041%
Games	50,547	16,860	452,989	8.96	0.053%
Yelp	30,431	20,033	316,354	10.40	0.052%

Table 1: Statistics of datasets. "#User", "#Item", "#Interaction", and "#Avg" denote the counts of users, items, interactions, and average interactions, respectively.

5 Experiments

5.1 Experimental Setup

We conducted experiments on four benchmark datasets: Instruments, Arts, and Games from the Amazon dataset[2], and the Yelp dataset[3]. Following baselines (Zheng et al., 2024; Wang et al., 2024b,d), we used the last and the second-to-last as test and validation data, and the rest as training data. Table 1 shows the dataset statistics. We utilized two widely used evaluation metrics: hit rate (H@K) and normalized discounted cumulative gain (N@K) with $K \in \{5, 10\}$. We performed parameter-efficient

[2]https://www.amazon.com/

[3]https://business.yelp.com/data/resources/open-dataset/

Dataset	Metric	Discriminative Decoding-based Approaches					Large Language Model-based Approaches							Imprv.
		Caser	HGN	BERT4Rec	SASRec	S^3-Rec	P5-CID	TIGER	ELMRec	RDRec	LR-Rec	LETTER	Ours	
Instruments	H@5	0.0543	0.0813	0.0671	0.0751	0.0857	0.0809	0.0870	0.0836	0.0862	0.0835	0.0921	**0.1053**	+14.3%*
	H@10	0.0710	0.1048	0.0822	0.0947	0.1121	0.0987	0.1058	0.0914	0.0930	0.1056	0.1137	**0.1260**	+10.8%*
	N@5	0.0355	0.0668	0.0560	0.0627	0.0621	0.0695	0.0737	0.0750	0.0783	0.0741	0.0791	**0.0883**	+11.6%*
	N@10	0.0409	0.0774	0.0608	0.0690	0.0705	0.0751	0.0797	0.0775	0.0813	0.0789	0.0848	**0.0950**	+12.0%*
Arts	H@5	0.0543	0.0813	0.0559	0.0757	0.0767	0.0724	0.0788	0.0782	0.0762	0.0931	0.0889	**0.1019**	+9.5%*
	H@10	0.0710	0.1048	0.0713	0.1051	0.1046	0.0902	0.1012	0.0824	0.0811	0.1133	0.1116	**0.1223**	+7.9%*
	N@5	0.0355	0.0668	0.0451	0.0521	0.0541	0.0607	0.0631	0.0720	0.0691	0.0752	0.0740	**0.0820**	+9.0%*
	N@10	0.0409	0.0744	0.0500	0.0612	0.0634	0.0664	0.0703	0.0731	0.0705	0.0812	0.0798	**0.0887**	+9.2%*
Games	H@5	0.0367	0.0517	0.0482	0.0581	0.0606	0.0506	0.0599	0.0390	0.0364	0.0626	0.0642	**0.0815**	+17.9%*
	H@10	0.0617	0.0856	0.0763	0.0940	0.1002	0.0803	0.0939	0.0528	0.0505	0.0930	0.0948	**0.1064**	+ 6.2%*
	N@5	0.0227	0.0333	0.0311	0.0365	0.0364	0.0342	0.0392	0.0300	0.0269	0.0437	0.0443	**0.0572**	+24.6%*
	N@10	0.0307	0.0442	0.0401	0.0481	0.0491	0.0392	0.0501	0.0345	0.0314	0.0535	0.0545	**0.0671**	+22.7%*
Yelp	H@5	0.0150	0.0186	0.0186	0.0183	0.0197	0.0219	0.0253	0.0230	0.0260	0.0250	0.0276	**0.0295**	+6.9%*
	H@10	0.0263	0.0326	0.0291	0.0296	0.0332	0.0347	0.0407	0.0359	0.0320	0.0282	0.0423	**0.0463**	+9.5%*
	N@5	0.0099	0.0115	0.0115	0.0116	0.0123	0.0140	0.0164	0.0158	0.0172	0.0175	0.0181	**0.0187**	+3.3%
	N@10	0.0134	0.0159	0.0159	0.0152	0.0168	0.0181	0.0213	0.0199	0.0212	0.0208	0.0224	**0.0243**	+8.5%*

Table 2: Performance comparison of recommender models. **Bold**: Best, underline: Second best. "*" indicates that the improvement is statistically significant in pairwise t-test across 10 trials (p-value < 0.05).

LoRA (Hu et al.) to fine-tune AGRec with LLaMA-1B (Dubey et al., 2024) as its backbone. More experimental details are provided in Appendix A.

We compared AGRec with eleven methods that are divided into the following two groups:

- **Discriminative Decoding-based Approaches**: Caser (Tang and Wang, 2018), SASRec (Kang and McAuley, 2018), HGN (Ma et al., 2019), BERT4Rec (Sun et al., 2019), and S^3-Rec (Zhou et al., 2020).
- **Large Language Model-based Approaches**: P5-CID (Geng et al., 2022; Hua et al., 2023), TIGER (Rajput et al., 2023), RDRec (Wang et al., 2024e), ELMRec (Wang et al., 2024d), LR-Rec (Zheng et al., 2024) and LETTER (Wang et al., 2024b).

5.2 Main Results

Table 2 presents the performance comparison between AGRec and baseline methods for sequential recommendations. These results demonstrate that AGRec consistently outperforms all competitors in four datasets in terms of H@K and N@K with $K \in \{5, 10\}$. Specifically, the improvements compared with the runner-ups, S^3-Rec, LR-Rec and LETTER, are 10.8% $\sim$ 14.3% in Instruments, 7.9% $\sim$ 9.5% in Arts, 6.2% $\sim$ 24.6% in Games, and 3.3% $\sim$ 9.5% in Yelp. We also obtain the following observations and insights:

(1) Compared with discriminative recommender models, most LLM-based methods that frame sequential recommendation as a language generation task achieve strong performance. This stems from (i) Transformers' exceptional ability to capture user sequential patterns behind prompting tokens and (ii) sufficient pre-trained general knowledge for generative reasoning.

Models	Instruments		Arts		Games	
	H@10	N@10	H@10	N@10	H@10	N@10
w/o Seq.	0.1197	0.0904	0.1171	0.0817	0.0995	0.0623
w/o Graph	0.0967	0.0755	0.0863	0.0635	0.0664	0.0339
AGRec	**0.1260**	**0.0950**	**0.1223**	**0.0887**	**0.1064**	**0.0671**
Impv.	30.3%	25.8%	40.4%	39.7%	60.2%	97.9%

Table 3: Ablation study. "w/o Seq." and "w/o Graph" denote the AGRec without sequential and graph-based features, respectively.

(2) TIGER, LR-Rec, and LETTER outperform other LLM-based recommenders due to their enhanced item tokenization, which leverages learnable RQ-VAE to effectively capture semantic relationships among items.

(3) By leveraging the strengths of both GNNs and LLMs, AGRec surpasses SOTA LLM-based models, LR-Rec and LETTER. This highlights the effectiveness of the rankable finite state machine in aligning GNN and LLM decoding while addressing the homogeneity issue.

(4) It is noteworthy that AGRec takes LLaMA-1B as its backbone while LR-Rec and LETTER are equipped with LLaMA-7B for this task. This indicates both efficacy and efficiency of AGRec for generative recommendations.

Models	Instruments		Arts		Games	
	H@10	N@10	H@10	N@10	H@10	N@10
LR-Rec	0.0888	0.0721	0.0858	0.0633	0.0579	0.0334
LETTER	0.0965	0.0756	0.0897	0.0650	0.0602	0.0343
AGRec	**0.1260**	**0.0950**	**0.1223**	**0.0887**	**0.1064**	**0.0671**
Imprv.	29.5%	24.1%	36.6%	36.5%	76.7%	95.6%

Table 4: Effect of SOTA baselines with LLaMA-1B.

Models	Instruments		Arts		Games	
	H@10	N@10	H@10	N@10	H@10	N@10
w/o Graph	0.0967	0.0755	0.0863	0.0635	0.0664	0.0420
+LightGCN	0.1260	0.0950	0.1223	0.0887	**0.1064**	**0.0671**
Imprv.	30.3%	25.8%	40.4%	39.7%	**60.2%**	**59.8%**
+SGL	**0.1284**	**0.0974**	**0.1304**	**0.0924**	0.1025	0.0638
Imprv.	**32.8%**	**29.0%**	**51.1%**	**45.5%**	54.4%	51.9%
+XSimGCL	0.1070	0.0851	0.1187	0.0878	0.0882	0.0549
Imprv.	10.7%	12.7%	37.5%	38.3%	32.8%	30.7%

Table 5: Effect of AGRec with various auxiliary GNNs.

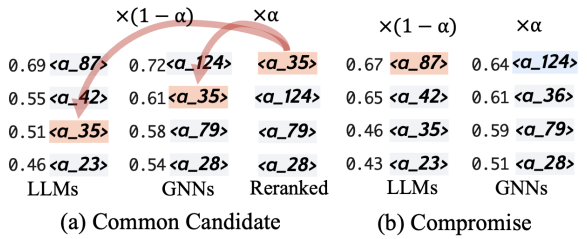


Figure 6: In-depth Analysis. (a) indicates that AGRec can emphasize the common candidates provided by LLMs and GNNs. (b) illustrates that AGRec have to make a compromise when either of LLMs or GNNs correctly predict the target token, relying on α.

5.3 Ablation Study

To investigate the effectiveness of sequential and graph-based features, we conducted an ablation study in Tables 3. We have the following findings:

(1) We can see that relying solely on graph-based features (i.e., w/o Seq.), AGRec fails to achieve optimal recommendation performance, suggesting the essentiality of sequential features by LLMs for recommendation.

(2) By augmenting LLM decoding logits with an auxiliary GNN, AGRec achieves a substantial improvement of 25.8% to 60.2% across three datasets. This underscores the effectiveness of integrating graph-based signals with the proposed rankable FSM to guide LLMs in item token generation.

5.4 Comparison with Lightweight Backbone

We compared AGRec with two SOTA LLM-based baselines, LR-Rec and LETTER, using the lightweight LLaMA-1B as backbone (Dubey et al., 2024). From Table. 4, we make two conclusions:

(1) Our AGRec, fine-tuned with parameter-efficient LoRA and utilizing only 1B parameters, consistently outperforms competing approaches by 24.1% to 95.6% across three datasets, showcasing its effectiveness and scalability when dealing with resource-limited scenarios.

(2) Although LETTER's enhanced item tokenization also leverages collaborative filtering embeddings from a pre-trained LightGCN for promising performance, AGRec achieves even better performance by directly augmenting LLM decoding with LightGCN logits through rankable FSMs.

5.5 Effect of Various Auxiliary GNNs

We investigated AGRec with three GNNs: LightGCN (He et al., 2020), SGL (Wu et al., 2021), and XSimGCL (Yu et al., 2023), which have achieved great success in recommendation tasks. The results in Table. 5 provide three observations:

(1) We selected LightGCN as the optimal auxiliary model due to its simplicity and effectiveness on the Instruments and Arts datasets.

(2) Overall, AGRec improves recommendation performance with the support of any of these GNN models, which demonstrates its robustness and generalization to GNNs;

(3) Although SGL yields the best performance on the Instruments and Arts datasets, it also increases the computational cost via graph structure enhancements, i.e., node dropout, edge dropout, and random walk.

5.6 In-depth Analysis

We conducted an in-depth analysis and obtained the following observations:

(1) As illustrated in Fig. 6 (a), AGRec highlights a candidate token when the token is commonly suggested by LLMs and GNNs, leveraging user sequential and high-order interactive patterns.

(2) In (b), when only either of LLMs and GNNs predicts the target token correctly during inference, AGRec should balance their influence by selecting an appropriate α.

(3) Compared to GNN-based discriminative methods that directly estimate the relevance between users and candidates, LLM-based methods generate tokens of the target item from the vocabulary, making recommendation more challenging.

(4) We analyze the key hyperparameter α, which balances the logits of LLMs and GNNs. As illus-

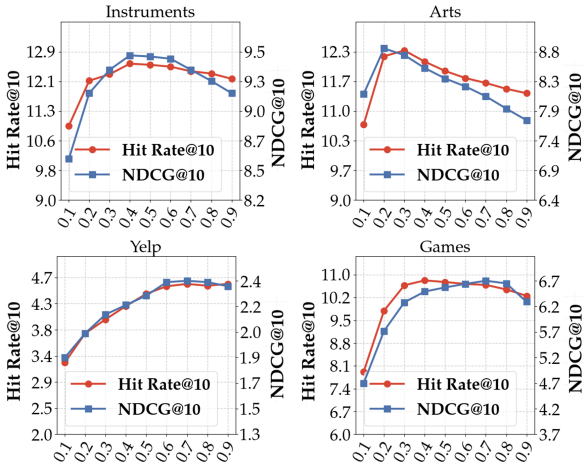


Figure 7: Performance by AGRec over different α on different datasets. The x-axis indicates the values of α.

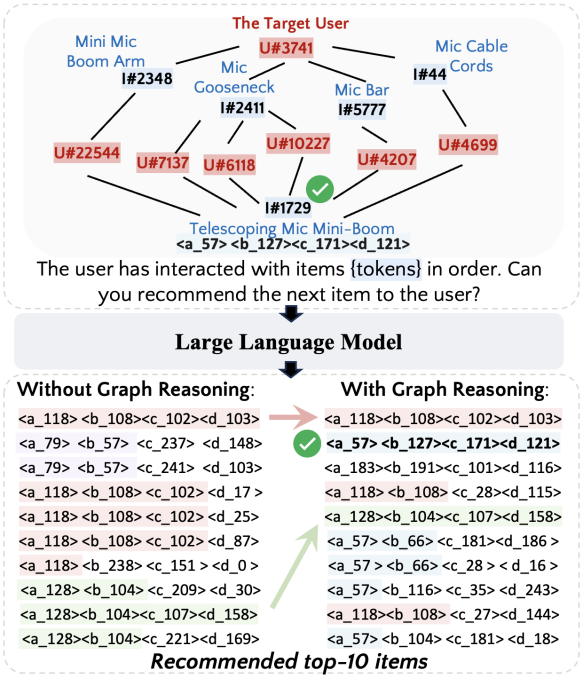


Figure 8: A practical case from the Instruments dataset. "I#A" and "U#B" represent the item and user with IDs of "A" and "B", respectively. The code sequences in the same color demonstrates the homogeneity issue.

trated in Fig. 7, the optimal α varies across datasets. Specifically, AGRec relies less on GNN logits for the Instruments dataset ($\alpha = 0.4$) and the Arts dataset ($\alpha = 0.2$), while for the Games and Yelp datasets, it requires more, with an optimal α of 0.7 for both.

5.7 Case Study

We conducted case study to better understand our AGRec. Fig. 8 presents a typical case from the Instruments dataset. We observed that:

(1) AGRec, utilizing graph reasoning, correctly recommends "Item#1729" to "User#3741", where many similar users who purchased the same products (e.g., Mini Mic Boom Arm, Mic Gooseneck, and Mic Bar) also bought the item.

(2) We can see that the recommended results start with various prefix tokens, lead to the correct prediction (i.e., Telescoping Mic Mini-Boom) for user U#3741. This indicates that the incorporation of GNN logits effectively mitigates the token homogeneity issue.

(3) As shown by pink and light green arrows, AGRec can highlight the tokens for items that are jointly recommended by LLMs and GNNs. Although item <a_118><b_108><c_102><a_103> (i.e., GLS Audio 3ft Patch Cable Cords) is not the target item this time, we believe that it remains a reliable recommendation in practical because it is suggested by both LLMs and GNNs.

5.8 Computational Complexity

Although AGRec utilizes the computationally expensive LightGCN model with a complexity of $\mathcal{O}((|\mathcal{U}| + |\mathcal{V}|)^2)$ during preprocessing, where $|\mathcal{U}|$ and $|\mathcal{V}|$ denote the number of users and items in the recommender system, this preprocessing computation is performed **only once**, making it feasible for real-world deployment.

Unlike prior works (Kim et al., 2024; Ma et al., 2024; Wang et al., 2024d) that employ GNNs to iteratively aggregate node features for LLM to learn, AGRec directly integrates the logits from an auxiliary pretrained GNN into LLM decoders. This design significantly reduces the reasoning burden during both training and inference, lowering the computational complexity from $\mathcal{O}((|\mathcal{U}| + |\mathcal{V}|)^2)$ to $\mathcal{O}(L^2)$, where L represents the average interaction sequence length per user and satisfies $L \ll (|\mathcal{U}| + |\mathcal{V}|)$. As a result, compared to these GNN-based approaches, our AGRec mitigates memory overhead in real-time recommendation scenarios with graph reasoning.

Besides, we use a prefix trie (Hua et al., 2023; Tan et al., 2024a; Wang et al., 2024b) constructed by the code sequences of all candidate items to prevent our AGRec from generating ghost items that are non-existent in the recommender system. Once the generated token does not match the trie, we force it to choose the early stoping token, "[EOS]". This setting could also reduce the inference time without scarifying the performance.

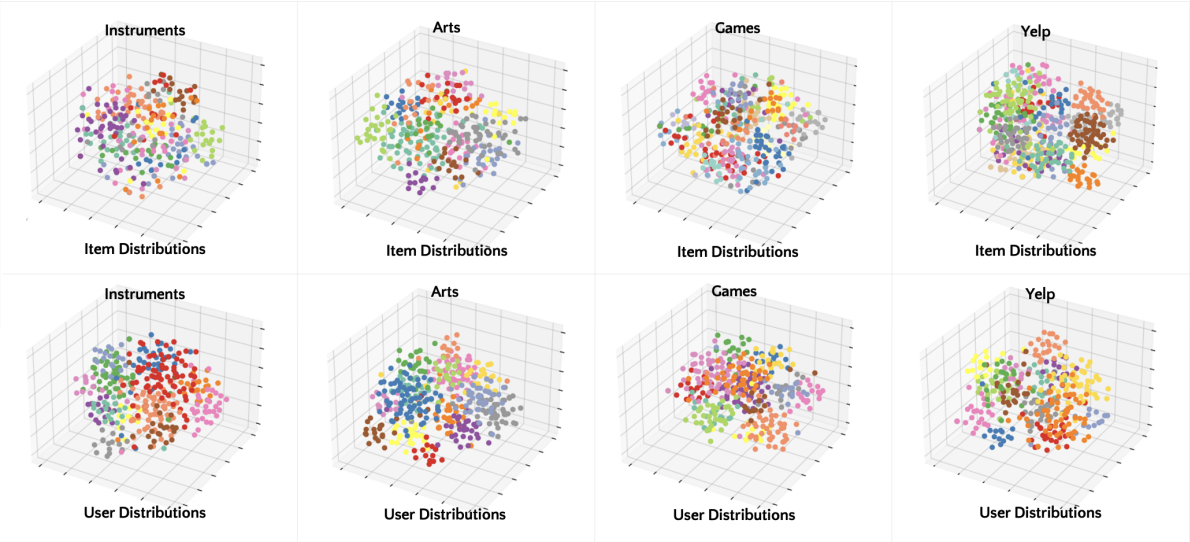


Figure 9: Visualization of user and item distributions based on LightGCN embeddings. The same color represents users (or items) who have purchased (or been purchased by) the same items (or users).

5.9 Visualization

We also visualized the distributions of users and items based on LightGCN embeddings using t-SNE (Van der Maaten and Hinton, 2008). The same color represents users who have purchased the same items or items that have been purchased by the same users. The results in Fig. 9 show that even when graph embeddings are compressed into three dimensions, dots of the same color still cluster closely together. We observed that LightGCN effectively transforms high-order collaborative signals into graph embeddings. This supports the notion that the inner products of these embeddings can serve as valuable guidance logits for LLMs in generative recommendation tasks.

6 Conclusion

This paper introduces AGRec that manifests complementary strengths of LLMs and GNNs in distinct user domains for sequential recommendations. We introduce a rankable finite state machine to adjust autoregressive generation with discriminative decoders, and mitigate token homogeneity to broaden the scope of beam search. The experimental results demonstrate the effectiveness and efficiency of AGRec. Our future work involves (i) incorporating multi-modal features, such as images (Geng et al., 2023; Bian et al., 2023) and (ii) exploring more effective item tokenization (Hua et al., 2023; Rajput et al., 2023; Wang et al., 2024b) for generative recommendations.

Limitations

The limitations of AGRec include: (i) The preprocessing stage employs the computationally expensive LightGCN model, with a costly complexity. Although this is ***only required once***, it partially limits the employment of AGRec in super large-scale recommendation scenarios. (ii) The AGRec model requires re-training the entire model from scratch for cold-start scenarios involving new users or items. This is a challenging, yet intriguing future work to further improve AGRec for zero / few-shot and cold-start recommendations (He et al., 2023; Sanner et al., 2023). (iii) The model focuses solely on interactive signals between users and items for recommendation, neglecting other important features such as textual reviews (Li et al., 2023a; Wang et al., 2024e; Wei et al., 2024) and images (Geng et al., 2023; Ji et al., 2023).

Ethics Statement

This paper does not involve the presentation of a new dataset, an NLP application, and the utilization of demographic or identity characteristics information.

Acknowledgements

We would like to thank anonymous reviewers for their thorough comments and suggestions. This work is supported by JKA (No.2024M-557), China Scholarship Council (No.202208330093), JSPS KAKENHI (No.22K12146), and SCAT.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631.

Keqin Bao, Jizhi Zhang, Yang Zhang, Xinyue Huo, Chong Chen, and Fuli Feng. 2024. Decoding matters: Addressing amplification bias and homogeneity issue in recommendations for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 10540–10552.

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 *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 1007–1014.

Shuqing Bian, Xingyu Pan, Wayne Xin Zhao, Jinpeng Wang, Chuyuan Wang, and Ji-Rong Wen. 2023. Multi-modal mixture of experts represetation learning for sequential recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 110–119.

Diego Carraro and Derek Bridge. 2024. Enhancing recommendation diversity by re-ranking with large language models. *ACM Transactions on Recommender Systems*.

Hongqiao Chen, Kexun Zhang, Lei Li, and William Yang Wang. Tooldec: Syntax error-free and generalizable tool use for llms via finite-state decoding. In *The 3rd Workshop on Mathematical Reasoning and AI at NeurIPS'23*.

Jiaju Chen, Chongming Gao, Shuai Yuan, Shuchang Liu, Qingpeng Cai, and Peng Jiang. 2025. Dlcrec: A novel approach for managing diversity in llm-based recommender systems. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, pages 857–865.

Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, et al. 2024. Exploring the potential of large language models (llms) in learning on graphs. *ACM SIGKDD Explorations Newsletter*, 25(2):42–61.

Andreas Damianou, Francesco Fabbri, Paul Gigioli, Marco De Nadai, Alice Wang, Enrico Palumbo, and Mounia Lalmas. 2024. Towards graph foundation models for personalization. In *Companion Proceedings of the ACM on Web Conference 2024*, pages 1798–1802.

Yingpeng Du, Di Luo, Rui Yan, Xiaopei Wang, Hongzhi Liu, Hengshu Zhu, Yang Song, and Jie Zhang. 2024a. Enhancing job recommendation through llm-based generative adversarial networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 8363–8371.

Yingpeng Du, Ziyan Wang, Zhu Sun, Haoyan Chua, Hongzhi Liu, Zhonghai Wu, Yining Ma, Jie Zhang, and Youchen Sun. 2024b. Large language model with graph convolution for recommendation. *arXiv preprint arXiv:2402.08859*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Edward Fredkin. 1960. Trie memory. *Communications of the ACM*, 3(9):490–499.

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 *Proceedings of the 16th ACM Conference on Recommender Systems*, pages 299–315.

Shijie Geng, Juntao Tan, Shuchang Liu, Zuohui Fu, and Yongfeng Zhang. 2023. Vip5: Towards multimodal foundation models for recommendation. In *2023 Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9606–9620.

Naicheng Guo, Hongwei Cheng, Qianqiao Liang, Linxun Chen, and Bing Han. 2024. Integrating large language models with graphical session-based recommendation. *arXiv preprint arXiv:2402.16539*.

Jesse Harte, Wouter Zorgdrager, Panos Louridas, Asterios Katsifodimos, Dietmar Jannach, and Marios Fragkoulis. 2023. Leveraging large language models for sequential recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 1096–1102.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 639–648.

Zhankui He, Zhouhang Xie, Rahul Jha, Harald Steck, Dawen Liang, Yesu Feng, Bodhisattwa Prasad Majumder, Nathan Kallus, and Julian McAuley. 2023. Large language models as zero-shot conversational recommenders. In *Proceedings of the 32nd ACM international conference on information and knowledge management*, pages 720–730.

Yupeng Hou, Zhankui He, Julian McAuley, and Wayne Xin Zhao. 2023. Learning vector-quantized item representation for transferable sequential recommenders. In *Proceedings of the ACM Web Conference 2023*, pages 1162–1171.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Wenyue Hua, Shuyuan Xu, Yingqiang Ge, and Yongfeng Zhang. 2023. How to index item ids for recommendation foundation models. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, pages 195–204.

Ting-Ji Huang, Jia-Qi Yang, Chunxu Shen, Kai-Qi Liu, De-Chuan Zhan, and Han-Jia Ye. 2024a. Improving llms for recommendation with out-of-vocabulary tokens. *arXiv preprint arXiv:2406.08477*.

Xu Huang, Jianxun Lian, Yuxuan Lei, Jing Yao, Defu Lian, and Xing Xie. 2023. Recommender ai agent: Integrating large language models for interactive recommendations. *arXiv preprint arXiv:2308.16505*.

Xuanwen Huang, Kaiqiao Han, Yang Yang, Dezheng Bao, Quanjin Tao, Ziwei Chai, and Qi Zhu. 2024b. Can gnn be good adapter for llms? In *Proceedings of the ACM on Web Conference 2024*, pages 893–904.

Jianchao Ji, Zelong Li, Shuyuan Xu, Wenyue Hua, Yingqiang Ge, Juntao Tan, and Yongfeng Zhang. 2024. Genrec: Large language model for generative recommendation. In *European Conference on Information Retrieval*, pages 494–502. Springer.

Wei Ji, Xiangyan Liu, An Zhang, Yinwei Wei, Yongxin Ni, and Xiang Wang. 2023. Online distillation-enhanced multi-modal transformer for sequential recommendation. In *Proceedings of the 31st ACM International Conference on Multimedia*, pages 955–965.

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

Nithish Kannen, Yao Ma, Gerrit J.j. Van Den Burg, and Jean Baptiste Faddoul. 2024. Efficient pointwise-pairwise learning-to-rank for news recommendation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 12403–12418.

Sein Kim, Hongseok Kang, Seungyoon Choi, Donghyun Kim, Minchul Yang, and Chanyoung Park. 2024. Large language models meet collaborative filtering: An efficient all-round llm-based recommender system. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1395–1406.

Yuxuan Lei, Jianxun Lian, Jing Yao, Xu Huang, Defu Lian, and Xing Xie. 2024. Recexplainer: Aligning large language models for explaining recommendation models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1530–1541.

Lei Li, Yongfeng Zhang, and Li Chen. 2023a. Personalized prompt learning for explainable recommendation. *ACM Transactions on Information Systems*, 41(4):1–26.

Lei Li, Yongfeng Zhang, and Li Chen. 2023b. Prompt distillation for efficient llm-based recommendation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pages 1348–1357.

Yaoyiran Li, Xiang Zhai, Moustafa Alzantot, Keyi Yu, Ivan Vulić, Anna Korhonen, and Mohamed Hammad. 2024. Calrec: Contrastive alignment of generative llms for sequential recommendation. In *Proceedings of the 18th ACM Conference on Recommender Systems*, pages 422–432.

Xinyu Lin, Wenjie Wang, Yongqi Li, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. 2023. A multi-facet paradigm to bridge large language model and recommendation. *arXiv preprint arXiv:2310.06491*.

Wensheng Lu, Jianxun Lian, Wei Zhang, Guanghua Li, Mingyang Zhou, Hao Liao, and Xing Xie. 2024. Aligning large language models for controllable recommendations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8159–8172.

Chen Ma, Peng Kang, and Xue Liu. 2019. Hierarchical gating networks for sequential recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 825–833.

Qiyao Ma, Xubin Ren, and Chao Huang. 2024. XRec: Large language models for explainable recommendation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 391–402.

Hyunsoo Na, Minseok Gang, Youngrok Ko, Jinseok Seol, and Sang-goo Lee. 2024. Enhancing large language model based sequential recommender systems with pseudo labels reconstruction. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7213–7222.

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. *The Journal of Machine Learning Research*, 21(1):5485–5551.

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, Jonah Samost, et al. 2023. Recommender systems with generative retrieval. *Advances in Neural Information Processing Systems*, 36:10299–10315.

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 *Proceedings of the ACM on Web Conference 2024*, pages 3464–3475.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2012. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*.

Scott Sanner, Krisztian Balog, Filip Radlinski, Ben Wedin, and Lucas Dixon. 2023. Large language models are competitive near cold-start recommenders for language-and item-based preferences. In *Proceedings of the 17th ACM conference on recommender systems*, pages 890–896.

Michael Sipser. 1996. Introduction to the theory of computation. *ACM Sigact News*, 27(1):27–29.

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 *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1441–1450.

Juntao Tan, Shuyuan Xu, Wenyue Hua, Yingqiang Ge, Zelong Li, and Yongfeng Zhang. 2024a. Idgenrec: Llm-recsys alignment with textual id learning. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 355–364.

Yanchao Tan, Hang Lv, Xinyi Huang, Jiawei Zhang, Shiping Wang, and Carl Yang. 2024b. Musegraph: Graph-oriented instruction tuning of large language models for generic graph mining. *arXiv preprint arXiv:2403.04780*.

Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 565–573.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Aaron van den Oord, Oriol Vinyals, and koray kavukcuoglu. 2017. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Vijay Viswanathan, Luyu Gao, Tongshuang Wu, Pengfei Liu, and Graham Neubig. 2023. Datafinder: Scientific dataset recommendation from natural language descriptions. In *the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10288–10303.

Chenyang Wang, Yuanqing Yu, Weizhi Ma, Min Zhang, Chong Chen, Yiqun Liu, and Shaoping Ma. 2022. Towards representation alignment and uniformity in collaborative filtering. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1816–1825.

Tianhao Wang, Sheng Wu, Fen Yi, Lidan Kuang, You Wang, and Jin Zhang. 2024a. Hybrid prompt recommendation explanation generation combined with graph encoder. *Neural Processing Letters*, 56(1):32.

Wenjie Wang, Honghui Bao, Xinyu Lin, Jizhi Zhang, Yongqi Li, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. 2024b. Learnable item tokenization for generative recommendation. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 2400–2409.

Xiaochen Wang, Junqing He, Liang Chen, Reza Haf Zhe Yang, Yiru Wang, Xiangdi Meng, Kunhao Pan, and Zhifang Sui. 2024c. Sg-fsm: A self-guiding zero-shot prompting paradigm for multi-hop question answering based on finite state machine. *arXiv preprint arXiv:2410.17021*.

Xinfeng Wang, Jin Cui, Fumiyo Fukumoto, and Yoshimi Suzuki. 2024d. Enhancing high-order interaction awareness in llm-based recommender model. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 11696–11711.

Xinfeng Wang, Jin Cui, Yoshimi Suzuki, and Fumiyo Fukumoto. 2024e. Rdrec: Rationale distillation for llm-based recommendation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 65–74.

Xinfeng Wang, Fumiyo Fukumoto, Jin Cui, Yoshimi Suzuki, Jiyi Li, and Dongjin Yu. 2023. Eedn: Enhanced encoder-decoder network with local and global context learning for poi recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 383–392.

Xinfeng Wang, Fumiyo Fukumoto, Jin Cui, Yoshimi Suzuki, Jiyi Li, and Dongjin Yu. 2024f. Cadrec: Contextualized and debiased recommender model. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 405–415.

Xinfeng Wang, Fumiyo Fukumoto, Jin Cui, Yoshimi Suzuki, and Dongjin Yu. 2024g. Nfarec: A negative feedback-aware recommender model. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 935–945.

Xinyuan Wang, Liang Wu, Liangjie Hong, Hao Liu, and Yanjie Fu. 2024h. Llm-enhanced user-item interactions: Leveraging edge information for optimized recommendations. *arXiv preprint arXiv:2402.09617*.

Wei Wei, Xubin Ren, Jiabin Tang, Qinyong Wang, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. 2024. Llmrec: Large language models with graph augmentation for recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 806–815.

Brandon T Willard and Rémi Louf. 2023. Efficient guided generation for large language models. *arXiv preprint arXiv:2307.09702*.

Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-supervised graph learning for recommendation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, pages 726–735.

Shuyuan Xu, Wenyue Hua, and Yongfeng Zhang. 2024. Openp5: An open-source platform for developing, training, and evaluating llm-based recommender systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 386–394.

Junliang Yu, Xin Xia, Tong Chen, Lizhen Cui, Nguyen Quoc Viet Hung, and Hongzhi Yin. 2023. Xsimgcl: Towards extremely simple graph contrastive learning for recommendation. *IEEE Transactions on Knowledge and Data Engineering*.

Junliang Yu, Hongzhi Yin, Xin Xia, Tong Chen, Lizhen Cui, and Quoc Viet Hung Nguyen. 2022. Are graph augmentations necessary? simple graph contrastive learning for recommendation. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*, pages 1294–1303.

Yakun Yu, Shi-ang Qi, Baochun Li, and Di Niu. 2024. PepRec: Progressive enhancement of prompting for recommendation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17941–17953.

Zhenrui Yue, Sara Rabhi, Gabriel de Souza Pereira Moreira, Dong Wang, and Even Oldridge. 2023. Llamarec: Two-stage recommendation using large language models for ranking. *arXiv preprint arXiv:2311.02089*.

Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. 2021. Soundstream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:495–507.

An Zhang, Yuxin Chen, Leheng Sheng, Xiang Wang, and Tat-Seng Chua. 2024a. On generative agents in recommendation. In *Proceedings of the 47th international ACM SIGIR conference on research and development in Information Retrieval*, pages 1807–1817.

Yang Zhang, Keqin Bao, Ming Yan, Wenjie Wang, Fuli Feng, and Xiangnan He. 2024b. Text-like encoding of collaborative information in large language models for recommendation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9181–9191.

Sen Zhao, Wei Wei, Ding Zou, and Xianling Mao. 2022. Multi-view intent disentangle graph networks for bundle recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 4379–4387.

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 *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 1435–1448. IEEE.

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 *Proceedings of the 29th ACM international conference on information & knowledge management*, pages 1893–1902.

Yaochen Zhu, Liang Wu, Qi Guo, Liangjie Hong, and Jundong Li. 2024. Collaborative large language model for recommender systems. In *Proceedings of the ACM on Web Conference 2024*, pages 3162–3172.

A Appendix

In this section, we present additional experimental details, including baselines, additional results, running time and implementation details.

A.1 Baselines

To evaluate the effectiveness of AGRec for sequential recommendation, we compared it with eleven baselines, which are classified into two groups:

(1) ***Discriminative Decoding-based Approaches***

- **CASER** (Tang and Wang, 2018) treats user interactions as images and employs 2-dimensional convolutions to capture sequential patterns.
- **SASRec** (Kang and McAuley, 2018) exploits Markov Chains to learn user sequential patterns for making recommendations.
- **HGN** (Ma et al., 2019) models users' long- and short-term interests with a novel gating strategy for sequential recommendations.
- **BERT4Rec** (Sun et al., 2019) proposes to leverage the BERT-style cloze task for the sequential recommender algorithm.
- **S^3-Rec** (Zhou et al., 2020) learns users' latent behavioral features via employing a self-supervised learning paradigm.

(2) ***Large Language Model-based Approaches***

- **P5** (Geng et al., 2022) presents a unified recommendation paradigm, which converts multiple recommendation tasks into natural language generation tasks using LLMs.
- **TIGER** (Rajput et al., 2023) characterizes the textual semantics of items as their IDs for LLM-based recommendations.
- **RDRec** (Wang et al., 2024e) proposes a rationale distillation to extract user preferences and item attributes from reviews.
- **ELMRec** (Wang et al., 2024d) explots GNN-based graph embedding to enhance LLMs' interpretation of graph-constructed interactions for recommendation.
- **LR-Rec** (Zheng et al., 2024) utilizes a learning-based vector quantization method with uniform semantic mapping, and a series of specially designed tuning tasks to enhance the integration of collaborative semantics in LLMs.
- **LETTER** (Wang et al., 2024b) integrates hierarchical semantics, collaborative signals, and code assignment diversity to satisfy the essential requirements of identifiers.

Models	Yelp			
	H@5	H@10	N@5	N@10
LR-Rec	0.0182	0.0260	0.0128	0.0151
LETTER	0.0188	0.0275	0.0131	0.0159
AGRec	**0.0295**	**0.0463**	**0.0187**	**0.0243**
Impv.	56.9%	68.4%	42.7%	52.8%

Table 6: Performance comparison of AGRec and SOTA LLM-based baselines, LR-Rec and LETTER, with LLaMA-1B as their backbone on the Yelp dataset.

Models	Yelp			
	H@5	H@10	N@5	N@10
w/o Graph	0.0185	0.0128	0.0272	0.0156
+LightGCN	0.0295	0.0463	0.0187	0.0243
Impv.	59.6%	46.1%	70.2%	55.8%
+SGL	**0.0308**	**0.0494**	**0.0201**	**0.0262**
Impv.	**66.5%**	**57.0%**	**81.6%**	**67.9%**
+XSimGCL	0.0259	0.0320	0.0172	0.0192
Impv.	40.0%	34.4%	17.6%	23.1%

Table 7: Performance comparison of AGRec with various auxiliary GNNs (i.e., LightGCN, SGL, and XSimGCL) on the Yelp dataset.

A.2 Additional Results

We provide additional experimental results on the Yelp dataset. Table 6 compares AGRec with state-of-the-art (SOTA) LLM-based baselines, i.e., LR-Rec and LETTER, all using LLaMA-1B as their backbone. On this dataset, AGRec outperforms the second-best baseline, LETTER, achieving improvements of 42.7% to 68.4% in terms of H@K and N@K with $K = 5$ and 10. This highlights the effectiveness and scalability of AGRec in scenarios with limited resources.

Table 7 presents the impact of different auxiliary GNNs on AGRec for sequential recommendation: LightGCN (He et al., 2020), SGL (Wu et al., 2021), and XSimGCL (Yu et al., 2023). We have the same observation: while SGL with a more complex architecture achieves the best performance on the Yelp dataset, we choose LightGCN as the optimal auxiliary model due to its balance of effectiveness and simplicity. This also demonstrates that AGRec can leverage more advanced GNNs as auxiliary models when sufficient computational resources are available.

Datasets	Stages		
	Pre-processing	Pre-training	Inference
Instruments	13m32s	2h06m45s	48m39s
Arts	21m34s	4h24m49s	1h47m20s
Games	25m55s	5h17m53s	1h51m46s
Yelp	17m12s	3h58m20s	1h09m28s

Table 8: Running time in different stages on four datasets. "h", "m", and "s" indicate "hours" and "minutes", and "seconds" respectively.

A.3 Running Time

In Table 8, we present the running time of AGRec in pre-processing, pre-training, and recommendation inference, respectively. We can see that the running time in these four datasets are acceptable while achieving SOTA performance in the sequential recommendation task. This is because the main computational cost of training and inference time comes from the LLaMA model, which means that AGRec saves a lot of computational cost and tunable parameters.

A.4 Implementation Details

The best hyperparameters of AGRec were sampled as follows: α was set to 0.4, 0.2, 0.7, and 0.7 for Instruments, Arts, Games, and Yelp, respectively. We set the scores of masked tokens as 1e-5 in FSMs. The number n of top items by LightGCN was set to 10. The learning rate was 1e-4 with the weight decay of 1e-2. Consistent with baselines (Zheng et al., 2024; Wang et al., 2024b), the training epoch was set to 4. These hyperparameters were tuned using Optuna[4] (Akiba et al., 2019). All prompts for sequential recommendation are borrowed from LR-Rec and LETTER. The dataset preprocessing also follows LR-Rec and LETTER for a fair comparison. We utilized the parameter-efficient fine-tuning technique, LoRA (Hu et al.), to fine-tune AGRec with LLaMA-1B (Dubey et al., 2024) as its backbone. The rank and parameter alpha of LoRA matrices was set to 16 and 32, respectively. We implemented LC-Rec with LLaMA-7B (Touvron et al., 2023). We implemented LETTER using LLaMA-7B and T5 (Raffel et al., 2020) and reported the higher of the two results, following the original paper (Wang et al., 2024b). For TIGER, we used the public implementation[5] as no official version is available.

As item tokenization is not our main focus, we followed LETTER to implement it, specifically, we used 4-level codebooks for RQ-VAE, where each codebook comprises up to 256 code embeddings with a dimension of 32. Consistent with baselines (Zheng et al., 2024; Wang et al., 2024b), we assigned all these codes as new tokens in the vocabulary. We use a code sequence trie (prefix tree) constructed by the code sequences of all candidate items (Hua et al., 2023; Tan et al., 2024a; Zheng et al., 2024; Wang et al., 2024b) to prevent our AGRec from generating ghost items that are nonexistent in the recommender system. The auxiliary LightGCN model was trained using the Bayesian personalized ranking (BPR) loss (Rendle et al., 2012) before the AGRec fine-tuning and inference processes. The embedding size d of LLaMA-1B was 2,048, and that of LightGCN was set to 512. We implemented and evaluated our AGRec using PyTorch on an NVIDIA RTX 6000 Ada (48GB memories). Our ARGrec only requires 24G GPU memories to train and run.

[4]https://github.com/pfnet/optuna

[5]https://github.com/HonghuiBao2000/LETTER