Inductive Generative Recommendation via Retrieval-based Speculation

Yijie Ding, Jiacheng Li, Julian McAuley, Yupeng Hou*

University of California, San Diego
{yid016, j9li, jmcauley, yphou}@ucsd.edu

Abstract

Generative recommendation (GR) is an emerging paradigm that tokenizes items into discrete tokens and learns to autoregressively generate the next tokens as predictions. While this token-generation paradigm is expected to surpass traditional transductive methods, potentially generating new items directly based on semantics, we empirically show that GR models predominantly generate items seen during training and struggle to recommend unseen items. In this paper, we propose SpecGR, a plug-and-play framework that enables GR models to recommend new items in an inductive setting. SpecGR uses a *drafter* model with inductive capability to propose candidate items, which may include both existing items and new items. The GR model then acts as a *verifier*, accepting or rejecting candidates while retaining its strong ranking capabilities. We further introduce the guided re-drafting technique to make the proposed candidates more aligned with the outputs of generative recommendation models, improving verification efficiency. We consider two variants for drafting: (1) using an auxiliary drafter model for better flexibility, or (2) leveraging the GR model's own encoder for parameter-efficient self-drafting. Extensive experiments on three real-world datasets demonstrate that SpecGR exhibits both strong inductive recommendation ability and the best overall performance among the compared methods.

Code — https://github.com/Jamesding000/SpecGR

Introduction

Generative recommendation (GR) is an emerging paradigm for sequential recommendation tasks (Rajput et al. 2024; Zhai et al. 2024; Zheng et al. 2024). By tokenizing each item into a few discrete tokens (named semantic IDs), models are trained to autoregressively generate the next tokens, which are then parsed as predicted items. Compared to conventional sequential recommendation methods (Kang and McAuley 2018; Sun et al. 2019), GR models scale up more easily and achieve better performance (Rajput et al. 2024; Deng et al. 2025), benefiting from the power of scaling laws (Zhang et al. 2023; Zhai et al. 2024).

Unlike item ID-based transductive models such as SASRec (Kang and McAuley 2018), which cannot recommend new items due to the absence of their IDs in the trained model, GR models are expected to exhibit inductive capabilities by capturing semantic correlations and generating semantic tokens for previously unseen items. However, our empirical analysis shows that GR models systematically assign higher likelihoods to seen semantic ID sequences than to unseen ones. As shown quantitatively in Table 2, GR yields near-zero recommendation performance on unseen items. We attribute this to the models' tendency to overfit to the semantic ID patterns present in the training data (Yang et al. 2024), making it unlikely for their outputs to match the semantic IDs of new items (Figure 1). In scenarios where up-to-date recommendations are critical (*e.g.,* news or short-video platforms), a flexible, on-the-fly inference framework is essential for the practical deployment of GR models.

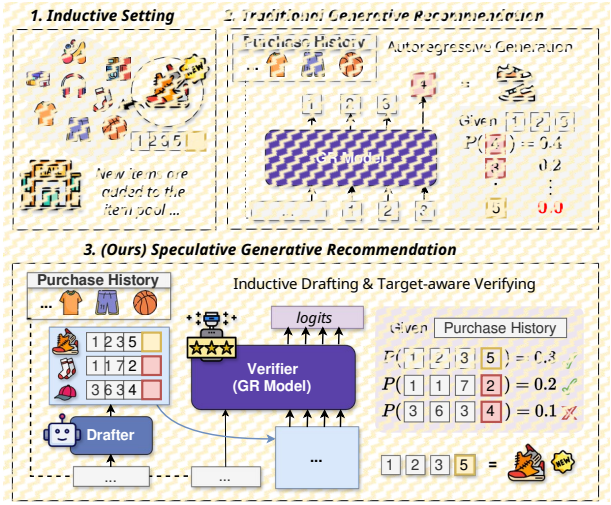

Figure 1: (1 & 2) GR models struggle to generate unseen items in an inductive setting. (3) SpecGR, a draft-then-verify framework, leverages GR models to verify candidates from an inductive drafter, enabling new-item recommendations.

In this work, we aim to develop *inductive* generative recommendation models that can recommend new items on-the-fly. Achieving this goal is non-trivial. In traditional recommendation systems, inductive recommendation (Wu et al. 2021) is typically achieved by incorporating side information and K-nearest neighbor (KNN) search. Although

*Corresponding author.

GR models can also tokenize items with side information into semantic IDs, as previously discussed, they struggle to generate unseen semantic ID patterns through autoregressive decoding (Freitag and Al-Onaizan 2017; Rajput et al. 2024). Recent efforts have explored blending GR outputs with items retrieved by non-GR methods (Rajput et al. 2024; Yang et al. 2024). However, these approaches fail to fully exploit the modeling strengths of GR models, resulting in suboptimal performance.

To this end, we propose **SpecGR** (**Spec**ulative **G**enerative **R**ecommendation), an inductive generative recommendation framework that can be integrated with GR models in a plug-and-play manner. We extend the concept of the drafter-verifier framework in the original speculative decoding technique (Leviathan, Kalman, and Matias 2023; Chen et al. 2023; He et al. 2023b). Rather than relying on a lightweight homologous model for inference acceleration, we explore the integration of models with different paradigms and capabilities. Specifically, we employ a KNN-based inductive model as the drafter to generate small batches of candidate items, while a GR model with stronger recommendation capabilities serves as the verifier, responsible for accepting or rejecting these candidates. Instead of merely blending outputs from different models, SpecGR ensures that all final recommendations are ranked based on scores assigned by the GR verifier. In addition, we propose guided re-drafting to improve the quality of candidate items proposed by the drafter model, leveraging the semantic ID prefixes generated by the verifier GR model. Furthermore, to reduce the overhead of maintaining a separate drafter model, we introduce **SpecGR++** that enables the encoder of the GR model to serve as a drafter.

Extensive experiments are conducted on three public datasets. We split the training and evaluation sets chronologically using fixed timestamp cut-offs. This setup ensures that recommendation models are evaluated in a setting where new items appear over time. The experimental results demonstrate that SpecGR significantly improves the ability of GR models to recommend new items and achieves strong overall performance compared to existing methods.

Related Work

Generative recommendation. Traditional sequential recommendation typically assigns a unique learnable embedding to each item, leading to optimization challenges due to large item vocabulary size (Hidasi et al. 2016; Kang and McAuley 2018; Sun et al. 2019). Generative recommendation (GR) addresses this by tokenizing items into discrete tokens and predicting the next item via autoregressive next-token generation (Rajput et al. 2024; Zheng et al. 2024; Liu, Hou, and McAuley 2024; Hou et al. 2025b). GR has demonstrated improved memory efficiency (Rajput et al. 2024; Hou et al. 2025a), scalability (Zhai et al. 2024), and promising end-to-end retrieval capabilities to unify retrieval and ranking (Deng et al. 2025). Despite these advantages, GR models tend to generate only semantic IDs observed during training, severely limiting their generalizability to unseen items. Despite a few early attempts (Rajput et al. 2024; Yang et al. 2024), this direction remains largely underexplored. In this work, we focus on developing effective frameworks that extend GR models to inductively recommend new items.

Cold-start & inductive recommendation. The item cold-start problem refers to the challenge of recommending items with limited interactions (Zhang et al. 2025; Wei et al. 2021; Zhou, Zhang, and Yang 2023). Common approaches use meta-learning that helps models generalize better from limited interactions, via gradient-based optimization (Lee et al. 2019), task adaptation (Lin et al. 2021; Wu and Zhou 2023), or memory-augmented modules (Dong et al. 2020; Zheng et al. 2021b). Inductive recommendation particularly tackles the challenge of recommending new items without any associated interactions. Existing methods leverage side information like tags or descriptions (Pazzani and Billsus 2007; Zhu et al. 2020; Li et al. 2023), modality representations (Hou et al. 2022, 2023; Sheng et al. 2024), and behavior patterns (Wu et al. 2021, 2020a). However, for GR models, the unique challenge lies in their intrinsic inability to generate unseen items during autoregressive token generation. Recent approaches enhance inductive capability by integrating heuristic candidates (Rajput et al. 2024) or dense retrieval results (Yang et al. 2024) with GR outputs. In this work, we propose a novel framework that converts the role of GR from generation to verification for inductive recommendation.

Speculative decoding. Large language models (LLMs) (Achiam et al. 2023; Zhao et al. 2023) have revolutionized a wide range of applications, yet suffer from high inference latency due to autoregressive decoding and large model size. Speculative decoding was proposed to accelerate LLM inference by leveraging a lightweight, homologous drafter model to generate future token sequences, which are then verified by a stronger target model, achieving lower latency and lossless performance (Leviathan, Kalman, and Matias 2023; Chen et al. 2023). Subsequent efforts have developed efficient drafting strategies (Cai et al. 2024; He et al. 2023b; Elhoushi et al. 2024), improved draft quality to maximize acceptance rate (Xiao et al. 2024; Gloeckle et al. 2024), and applied the framework to diverse domains (Wang et al. 2024b; De Bortoli et al. 2025). Recently, Lin et al. (2024) extended speculative decoding to accelerate top-K sequence generation in the GR setting. In this work, rather than focusing on acceleration, we leverage the draft-verify framework to bring inductive capabilities to generative recommendation models.

Methodology

Problem Setup and Formulation

We follow the inductive sequential recommendation task (Hou et al. 2022; Yuan et al. 2023). The input is a sequence of items $\{x_1, x_2, \ldots, x_w\}$ ordered chronologically based on user interaction time. Each item $x \in I$ has associated text features, such as title, description, and category. Here, w denotes the length of the item sequence. The task is to predict the next item of interest. Note that in the inductive setting, the target item may not appear in the training set, called new or unseen items in the following sections.

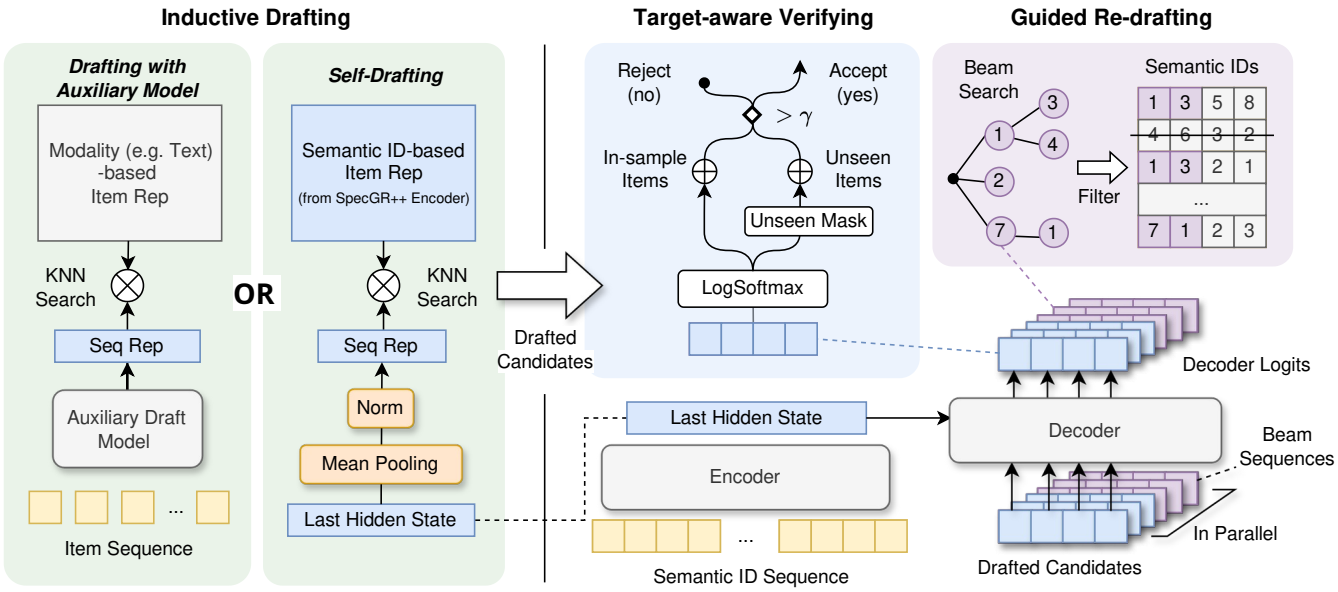

Figure 2: Illustration of the proposed SpecGR method, a draft-and-verify framework that iteratively performs drafting and verification until enough items are accepted. (a) **Inductive Drafting**. The inductive drafter first retrieves a set of candidates that contain new items. We present two drafting methods: using an auxiliary model or the GR's encoder output (namely, self-drafting) for item retrieval. (b) **Target-aware Verifying**. The GR model accepts or rejects the candidates based on the likelihood of being the target. (c) **Guided Redrafting**. If not enough items are accepted, the GR filters the candidate space for the next drafting round based on the generated beam sequences.

In this work, we focus on developing a general inductive recommendation framework for generative recommendation models. Typical GR models like TIGER (Rajput et al. 2024) will tokenize each item x_i into a semantic ID pattern, $\text{ID}_i := [\langle c_1^i \rangle, \langle c_2^i \rangle, \ldots, \langle c_l^i \rangle]$, where l denotes the number of digits of one item's semantic ID pattern, and $\langle c \rangle$ denotes one digit of semantic ID. In this way, the input for GR models can be represented as follows by replacing the items in the original item sequence with the corresponding semantic IDs:

$$X = [\langle \texttt{bos} \rangle, \text{ID}_1, \text{ID}_2, \ldots, \text{ID}_w, \langle \texttt{eos} \rangle], \quad (1)$$

where $\langle \texttt{bos} \rangle$ and $\langle \texttt{eos} \rangle$ are special tokens indicating the start and ending positions of a semantic ID sequence. Then the GR models are trained to generate K semantic ID patterns, which will be further parsed into recommended items with top-K probabilities. In the inductive setting, we assume that new items have been assigned semantic ID patterns. These new items can be parsed if the outputs of the GR models match the new semantic ID patterns.

Speculative Generative Recommendation

We begin by providing an overview of the proposed SpecGR framework, as illustrated in Figure 2. The framework consists of four components: (1) **Inductive Drafting.** Given the same input item sequence as the generative recommendation model, a drafter model with inductive recommendation capabilities proposes a set of candidate items as recommendation drafts. (2) **Target-aware Verifying.** The GR model, acting as a verifier, either accepts or rejects the candidates based on the probability that they could be targets of the input sequence. (3) **Guided Re-drafting.** If the number of accepted items does not meet the required recommendations K, the GR model guides the drafter to re-draft and propose the next set of candidate items. (4) **Adaptive Exiting.** Once K items are accepted, the framework exits and outputs the items, sorted by the scores given by the GR model. This process selects high-quality, unseen items through drafting and verifying, while maintaining the strong recommendation capabilities of generative recommendation models.

Inductive Drafting Instead of expecting GR models to directly generate the semantic IDs of unseen items, we employ an inductive drafter model to first propose "recommendation drafts" that may include unseen items. Given the input item sequence X, the drafter model $\text{D}(\cdot)$ performs inductive drafting by recommending a set of δ candidates $Q = \text{D}(X)$, where $|Q| = \delta$. Note that in the original speculative decoding technique, the drafter model is considered an efficient approximation of the target model (Leviathan, Kalman, and Matias 2023; Chen et al. 2023), functioning as a homologous model to the verifier model. Here, we extend the above restriction in choosing drafter models. We do not require the drafter model to be a GR model but instead an inductive model to bring new capabilities to the following verifier model (in our case, the GR model). This approach allows high-quality unseen items to be introduced into the system, which are then verified by the more expressive generative model and finally included in the recommendations. Hence,

the inductive drafter can be any inductive recommendation model, and we will discuss two effective instantiations in the subsequent Drafting Strategies section.

Target-aware Verifying While inductive drafters excel at recommending unseen items, they are not as effective as GR models in modeling input sequences and providing recommendations. Therefore, after obtaining candidates Q from the inductive drafting process, we use the generative model to verify them, rejecting items with low likelihood.

• ***Target-aware likelihood for ranking.*** Given a candidate item as a potential target, we use the GR model as a query-likelihood model (QLM) (Zhuang et al. 2023; Zhuang, Li, and Zuccon 2021; Nogueira, Lin, and Epistemic 2019). The QLM scores the query, *i.e.,* input sequence and the potential target, by measuring the likelihood of the model consecutively generating the tokens in the query. Previous studies have demonstrated that generative language models, such as T5 (Raffel et al. 2020), exhibit robust zero-shot query-likelihood ranking performance in document retrieval tasks, without explicit fine-tuning for document ranking (Zhuang et al. 2023). Accordingly, conditioning on the input sequence X, we adopt the conditional probability of generating the target semantic ID pattern as the verification score.

• ***Likelihood score calculation.*** However, naively applying the QLM for verification would result in low scores for unseen items. This happens because not all digits of semantic IDs are derived from item semantics. In addition to the tokens learned purely from item semantics, existing methods usually add an extra digit to avoid conflicts, known as the item identification token (Rajput et al. 2024; Liu, Hou, and McAuley 2024). For unseen items, the probability of generating this identification token lies outside the modeling distribution and therefore primarily consists of noise.

To provide a fair verification score for unseen items, we exclude the identification token and calculate only the probability of other digits. The target-aware verification score can be computed as:

$$\mathsf{V}(x_t, X) = \begin{cases} \frac{1}{l}\sum_{i=1}^{l} \log P(c_i^t \mid c_{<i}^t, X) & \text{if } x_t \in I, \\ \frac{1}{l-1}\sum_{i=1}^{l-1} \log P(c_i^t \mid c_{<i}^t, X) & \text{if } x_t \in I^* \setminus I, \end{cases} \tag{2}$$

where $\mathsf{V}(\cdot)$ denotes the verifier model, which takes the input sequence X and the potential target item x_t as inputs, outputing the log-likelihood probability scores. l denotes the total number of digits in each semantic ID pattern, where the last digit is assumed to be the item identification token. $P(\cdot)$ denotes the backbone autoregressive model, which takes semantic ID sequences and outputs the likelihood scores. c_i^t refers to the i-th digit of the semantic ID for the target item x_t. The set $I^* \setminus I$ represents the unseen items.

To alleviate the bias caused by varying lengths of semantic IDs for unseen and existing items, we normalize the log-likelihood scores by the corresponding number of digits. After obtaining the likelihood score, we accept the items if $\mathsf{V}(x_t, X) > \gamma$, where γ is a hyperparameter and can be tuned on the validation set.

Guided Re-drafting If fewer than K items are accepted from the initial batch of drafted candidates, the drafter model $\mathsf{D}(\cdot)$ must generate an additional batch of δ new candidates. Intuitively, since these candidates appear lower in the drafter's ranking, their expected acceptance probability under the verifier $\mathsf{V}(\cdot)$ is substantially reduced. To counteract this decline in acceptance rate, we introduce the *guided re-drafting* mechanism that steers subsequent candidate batches toward regions of the item space that better match the verifier's scoring distribution.

Guided re-drafting operates by steering the drafter models using a set of semantic ID prefixes generated by the verifier models (GR models). Specifically, after verifying the j-th batch of recommendation drafts, the verifier model generates a set of beam sequences B_j using beam search, where each sequence is a j-digit semantic ID prefix. In the next draft-verify iteration, the drafter model is guided to propose only candidates Q_j whose prefixes match those in B_j:

$$Q_j = \{ x_i \mid x_i \in \mathsf{D}(X), (c_1^i, c_2^i, \ldots, c_j^i) \in B_j \}, \tag{3}$$

where B_j denotes the set of semantic ID prefixes, with a hyperparameter β as the set size. Guided re-drafting happens along with the beam search decoding process of the GR model. It is important to note that the total number of draft-verify iterations will not exceed l, which corresponds to the maximum length of the semantic IDs, and is also equal to the maximum number of decoding steps.

Adaptive Exiting SpecGR can adaptively terminate the draft-verify iterations based on the number of candidate items accepted by the verifier (GR) model. When the number of accepted items reaches K, the loop exits, avoiding the need to generate full-length sequences of l. This adaptive approach reduces inference time, as fewer generation steps are required. If, after the final iteration, there are still not enough accepted items, the beam sequences will be appended to the recommendation list until K is reached. As a result, even in the worst case, SpecGR does not incur additional time overhead compared to decoding with beam search. Finally, we rank the recommendation list by using the verification scores of the accepted items, along with the beam scores if items from beam sequences are included.

Drafting Strategies

In this section, we present two methods for drafting: by using an auxiliary draft model, and by reusing the encoder of generative recommendation models (namely SpecGR++).

Auxiliary Draft Model The most straightforward way to draft is to introduce an auxiliary inductive recommendation model. An example is UniSRec (Hou et al. 2022), which uses modality-based item representations for KNN search. When new items are added, their representations can be directly incorporated into the item pool. The model can then retrieve new items if their modality-based representations are similar to the sequence representations.

Self-Drafting via GR Encoder Despite the flexibility of using an auxiliary model as the drafter, issues such as communication latency and distribution shift may arise. Thus, we propose **SpecGR++**, which reuses the encoder module of the generative recommendation model to function as an inductive drafter model. The general idea is to encode both (1) the semantic IDs of a single item, and (2) the input semantic ID sequence of user history, using the same encoder module. Then we apply KNN search to retrieve semantic IDs of both existing and new items.

• ***Semantic ID-based item and sequence representations.*** To derive sequence representations, we use the same input format as our GR model (Equation (1)). To derive item representations, we format the semantic IDs of a single item x_i in the same way as the encoder's input, *i.e.,* $[\langle \texttt{bos} \rangle, \text{ID}_i, \langle \texttt{eos} \rangle]$, where ID_i represents the semantic ID pattern of item x_i. To obtain the item and sequence representations, we take the last hidden state from the GR encoder and apply mean pooling.

• ***Item–sequence contrastive pretraining.*** Prior work shows that hidden states from generative models are not directly suitable as representations (Ni et al. 2022a,b). To obtain strong inductive embeddings, we jointly train the GR encoder with a contrastive objective following (Chen et al. 2020; Hou et al. 2022). Given sequence embeddings $\boldsymbol{s}_j$ and next-item embeddings $\boldsymbol{i}_j$, the contrastive loss is:

$$L_{\text{CL}} = -\frac{1}{B_{\text{emb}}} \sum_{j=1}^{B_{\text{emb}}} \log \frac{\exp(\boldsymbol{s}_j \cdot \boldsymbol{i}_j / \tau)}{\sum_{j'=1}^{B_{\text{emb}}} \exp(\boldsymbol{s}_j \cdot \boldsymbol{i}_{j'} / \tau)}. \quad (4)$$

We optimize L_{CL} jointly with the next-token generation loss:

$$L_{\text{Gen}} = -\frac{1}{B_{\text{gen}}} \sum_{b=1}^{B_{\text{gen}}} \frac{1}{L_b} \sum_{t=1}^{L_b} \log P(c_{b,t} \mid c_{b,<t}, X_b). \quad (5)$$

To ensure sufficient in-batch negatives for the contrastive objective, we use a larger embedding batch (*i.e.*$B_{\text{emb}} > B_{\text{gen}}$). The final multi-task loss is expressed as $L = \lambda_1 L_{\text{CL}} + L_{\text{Gen}}$, where λ_1 is a hyperparameter balancing the two tasks.

• ***Learning-to-rank fine-tuning.*** Following Li et al. (2023), to further enhance the ranking ability of the semantic ID encoder, we continue to fine-tune the encoder using the cross-entropy loss L_{CE} on a larger batch of negative items (equivalent to L_{CL} in Equation (4) with $B_{\text{emb}} = |I|$). To enable efficient large-batch training, the item representations are frozen at the beginning of the fine-tuning phase. The overall loss for fine-tuning can be written as $L' = \lambda_2 L_{\text{CE}} + L_{\text{Gen}}$, where λ_2 is a hyperparameter.

Experiments

In this section, we present experimental results to answer the following Research Questions (**RQ**).

- **RQ1:** How does SpecGR perform compared with state-of-the-art baselines on overall and seen/unseen subsets?
- **RQ2:** Do all of SpecGR's designs take effect?
- **RQ3:** How do key hyperparameters affect SpecGR's performance and efficiency?
- **RQ4:** Is SpecGR an effective plug-and-play framework for different drafters and GR backbones?

Experimental Setup

Datasets We use three categories, Video Games (**Games**), Office Products (**Office**), and Cell Phones and Accessories (**Phones**), from the Amazon Reviews 2023 dataset (Hou et al. 2024a) as our experimental datasets. To assess the performance of SpecGR in real-world settings, we utilize pre-processed benchmarks[1] that exclude users and items with fewer than five interactions. The data is split into training, validation, and test sets based on predefined timestamp cut-offs. Notably, since the datasets are split by timestamps, the validation and test sets naturally include unseen items. This simulates a more realistic scenario in comparison to the widely-used leave-last-out splitting.

Compared Methods We report results for two SpecGR variants: $\text{SpecGR}_{\text{Aux}}$, which uses UniSRec (Hou et al. 2022) as an auxiliary drafter, and SpecGR++, which uses its own encoder module for drafting. We compare SpecGR against the following state-of-the-art methods: ID-based methods such as SASRec (Kang and McAuley 2018); feature+ID-based methods such as FDSA (Zhang et al. 2019) and S^3-Rec (Zhou et al. 2020); modality-based methods such as UniSRec (Hou et al. 2022) and RecFormer (Li et al. 2023); and generative methods including TIGER, TIGER_{C} (Rajput et al. 2024), and LIGER (Yang et al. 2024). Notably, TIGER_{C} employs a heuristic strategy that mixes a fixed proportion of unseen items into TIGER's recommendation list.

Evaluation Setting We adopt Recall@K and NDCG@K as metrics to evaluate the compared methods, where $K \in \{10, 50\}$. In addition, based on whether the target items in the test set are existing items or new items (not shown in the training set), we split our test set into two subsets, named **In-Sample** and **Unseen**, respectively. The model checkpoints of all compared methods that have the best overall performance on the validation set will be evaluated on the test set.

Implementation Details We use UniSRec (Hou et al. 2022) as the auxiliary drafter model for $\text{SpecGR}_{\text{Aux}}$ and TIGER (Rajput et al. 2024) as the GR backbone for both SpecGR variants. Input sequences are truncated to a maximum of 20 items, following Rajput et al. (2024), and the same semantic ID tokenization process is applied. SpecGR++ is trained using a multi-task setup ($\lambda_1 = \lambda_2 = 6.0$) combining generation and contrastive objectives, followed by a learning-to-rank fine-tuning phase. We use $B_{\text{Emb}} = 2048$ and $B_{\text{Gen}} = 256$. Hyperparameters such as the draft threshold (γ), beam size (β), and draft size (δ) are tuned on validation splits.

Performance Analysis (RQ1)

Overall performance. We compare SpecGR with sequential and generative recommendation baselines across three public datasets; results are summarized in Table 1. ID-based and feature-based methods generally show poor performance, especially on sparse datasets (*e.g.,* Phones). Modality-based methods, such as UniSRec and Recformer,

[1] https://huggingface.co/datasets/McAuley-Lab/Amazon-Reviews-2023/tree/main/benchmark/5core/timestamp_w_his

Table 1: Performance comparison of different models. The best and the second-best performance is denoted in bold and underlined fonts, respectively. "R@K" is short for "Recall@K" and "N@K" is short for "NDCG@K". "Improv." denotes the improvement ratio of SpecGR compared to the best-performing baseline model.

Dataset	Metric	ID-based	Feature + ID		Modality-based			Generative			Ours		Improv.
		$\text{SASRec}_{\text{ID}}$	FDSA	S^3-Rec	SASRec_{T}	UniSRec	Recformer	TIGER	TIGER_{C}	LIGER	$\text{SpecGR}_{\text{Aux}}$	SpecGR++	
Games	R@10	0.0186	0.0190	0.0195	0.0179	0.0225	0.0243	0.0222	0.0226	0.0139	**0.0254**	0.0250	+4.53%
	N@10	0.0093	0.0101	0.0094	0.0091	0.0115	0.0111	0.0114	0.0115	0.0068	**0.0128**	0.0124	+10.40%
	R@50	0.0477	0.0496	0.0473	0.0507	0.0621	0.0740	0.0584	0.0611	0.0635	**0.0778**	0.0717	+5.13%
	N@50	0.0162	0.0167	0.0154	0.0161	0.0200	0.0218	0.0193	0.0198	0.0172	**0.0239**	0.0225	+9.72%
Office	R@10	0.0093	0.0095	0.0100	0.0091	0.0119	0.0126	0.0132	0.0130	0.0059	**0.0138**	0.0134	+3.99%
	N@10	0.0047	0.0050	0.0052	0.0048	0.0062	0.0039	0.0071	0.0070	0.0029	**0.0072**	0.0070	+1.68%
	R@50	0.0217	0.0224	0.0234	0.0233	0.0322	0.0340	0.0308	0.0312	0.0268	**0.0360**	0.0332	+5.93%
	N@50	0.0074	0.0078	0.0080	0.0078	0.0105	0.0106	0.0109	0.0110	0.0072	**0.0119**	0.0113	+8.76%
Phones	R@10	0.0052	0.0067	0.0058	0.0072	0.0084	0.0074	0.0090	0.0087	0.0048	0.0099	**0.0101**	+11.90%
	N@10	0.0027	0.0035	0.0028	0.0037	0.0045	0.0036	0.0047	0.0046	0.0022	0.0050	**0.0052**	+10.64%
	R@50	0.0143	0.0184	0.0151	0.0188	0.0233	0.0236	0.0232	0.0233	0.0226	**0.0285**	0.0275	+20.64%
	N@50	0.0047	0.0060	0.0048	0.0062	0.0077	0.0070	0.0078	0.0078	0.0059	**0.0090**	**0.0090**	+14.80%

Table 2: Model performance breakdown on the "in-sample" and "unseen" subsets. The proportions of the test cases in each subset relative to the entire test data have been labeled. The best and second-best results are bolded and underlined.

Model	#Params. (M)	Games						Phones					
		Overall		In-Sample (39.7%)		Unseen (60.3%)		Overall		In-Sample (31.8%)		Unseen (68.2%)	
		R@50	N@50	R@50	N@50	R@50	N@50	R@50	N@50	R@50	N@50	R@50	N@50
UniSRec	2.90	0.0621	0.0200	0.1386	0.0461	0.0118	0.0029	0.0233	0.0077	0.0604	0.0211	0.0060	0.0014
Recformer	233.73	0.0740	0.0218	0.1082	0.0333	0.0514	0.0142	0.0236	0.0070	0.0340	0.0103	**0.0188**	**0.0055**
TIGER	13.26	0.0584	0.0193	0.1472	**0.0486**	-	-	0.0232	0.0078	0.0730	0.0245	-	-
TIGER_{C}	13.26	0.0611	0.0198	0.1447	0.0482	0.0061	0.0011	0.0233	0.0078	0.0691	0.0238	0.0019	0.0003
LIGER	13.26	0.0635	0.0172	0.0438	0.0160	**0.0765**	**0.0179**	0.0226	0.0059	0.0472	0.0107	0.0111	0.0037
$\text{SpecGR}_{\text{Aux}}$	16.16	**0.0778**	**0.0239**	**0.1485**	0.0457	0.0312	0.0096	**0.0285**	**0.0090**	**0.0748**	0.0237	0.0069	0.0021
SpecGR++	13.28	0.0717	0.0225	0.1323	0.0439	0.0318	0.0084	0.0275	**0.0090**	0.0730	**0.0246**	0.0063	0.0017

show improved performance by leveraging powerful text embeddings from Pretrained Language Models (PLMs). Generative recommendation models achieve the best results through autoregressive modeling of fine-grained semantic IDs. Among all models, SpecGR consistently achieves the best overall performance (*e.g.,* up to +14.8% in NDCG@50, and +20.64% in Recall@50). Notably, SpecGR++ attains both better parameter efficiency and comparable performance to $\text{SpecGR}_{\text{Aux}}$, highlighting the GR encoder's effectiveness in learning robust semantic ID-based representations for inductive recommendation.

Subset Analysis. Next, we analyze the detailed performance breakdown on the in-sample and unseen subsets, as shown in Table 2. TIGER achieves strong in-sample performance but fails to generalize inductively on the unseen subset. SpecGR addresses this limitation by integrating inductive drafting with GR-based verification, greatly improving inductive generalization without compromising in-sample quality. Recformer is the best-performing modality-based method. However, its LLM backbone has significantly larger model size compared to other baselines. While LIGER attains higher performance on unseen items via dense retrieval-based candidate blending, its heuristic combination of candidates introduces irrelevant items, yielding a suboptimal trade-off and degraded overall recommendation quality. In contrast, SpecGR employs GR's target-aware likelihood scores to filter inductive candidates, resulting in strong inductive ability and the best overall performance.

Ablation Study (RQ2)

SpecGR Inference Framework. First, we assess the contributions of inference components in SpecGR++, as summarized in Table 3. Inductive drafting (1.1) and likelihood score adjustment (1.2) substantially enhance inductive generalization by incorporating meaningful unseen candidates. Guided re-drafting (1.3) boosts recommendation quality when initial drafting provides insufficient accepted candidates. Removing verification-based re-ranking (1.4) and adaptive exiting (1.5) results in performance degradation, confirming the importance of these modules in maintaining a balance between in-sample accuracy and inductive capability.

SpecGR++ Training Paradigm. Next, we analyze training paradigm variants for SpecGR++. Directly using TIGER's encoder states without dedicated representation learning (2.1) severely limits inductive recommendation. Contrastive pretraining (2.2) significantly improves representation quality, and subsequent fine-tuning (2.3) further refines performance, validating our two-stage training design.

Table 3: Ablation study on SpecGR++ inference and training. The best and second-best results are bolded and underlined.

Variants	Games				Office				Phones			
	R@50	N@50	R@10	N@10	R@50	N@50	R@10	N@10	R@50	N@50	R@10	N@10
(1.1) *w/o* inductive drafting	0.0609	0.0202	0.0235	<u>0.0121</u>	0.0306	0.0109	<u>0.0132</u>	**0.0070**	0.0233	0.0080	0.0092	0.0049
(1.2) *w/o* likelihood score adjustment	<u>0.0712</u>	0.0221	0.0236	0.0119	0.0331	0.0103	0.0118	0.0057	0.0236	0.0081	0.0092	0.0049
(1.3) *w/o* guided re-drafting	0.0611	0.0202	0.0235	<u>0.0121</u>	0.0309	0.0110	<u>0.0132</u>	**0.0070**	0.0264	0.0086	0.0096	0.0050
(1.4) *w/o* item re-ranking	0.0703	0.0219	<u>0.0239</u>	0.0120	**0.0334**	**0.0113**	0.0131	0.0069	0.0264	0.0083	0.0093	0.0047
(1.5) *w/o* adaptive exiting	0.0694	0.0200	0.0203	0.0095	0.0313	0.0108	0.0126	0.0068	<u>0.0265</u>	0.0086	0.0095	0.0050
(2.1) TIGER for SpecGR++	0.0582	0.0192	0.0224	0.0114	0.0302	0.0105	0.0127	0.0067	0.0232	0.0078	0.0090	0.0047
(2.2) *w/o* contrastive pretraining	0.0581	0.0193	0.0221	0.0115	0.0313	0.0108	0.0126	0.0068	0.0234	0.0077	0.0093	0.0050
(2.3) *w/o* fine-tuning	0.0692	**0.0225**	0.0222	0.0111	0.0325	0.0110	0.0129	0.0068	0.0259	<u>0.0087</u>	<u>0.0098</u>	<u>0.0051</u>
SpecGR++	**0.0717**	**0.0225**	**0.0250**	**0.0124**	<u>0.0332</u>	**0.0113**	**0.0134**	**0.0070**	**0.0275**	**0.0090**	**0.0101**	**0.0052**

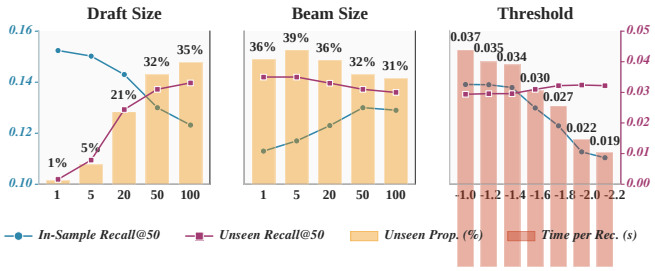

Figure 3: Impact of hyperparameters on SpecGR's performance and efficiency. (Left, middle): Bars show the proportion of unseen items in recommendations. (Right): Bars represent inference latency in seconds. Lines depict the trade-off between in-sample and unseen Recall@50.

GR	Drafter	U-N@50	O-N@50
TIGER	Baseline	–	0.0193
	GR Encoder (SpecGR++)	0.0084	0.0225 (+16.6%)
	Semantic-KNN	<u>0.0085</u>	<u>0.0231</u> (+19.7%)
	UniSRec	**0.0096**	**0.0239** (+23.8%)
DSI	Baseline	–	0.0198
	GR Encoder (SpecGR++)	**0.0061**	<u>0.0217</u> (+9.6%)
	Semantic-KNN	0.0049	<u>0.0217</u> (+9.6%)
	UniSRec	<u>0.0058</u>	**0.0220** (+11.1%)

Table 4: NDCG@50 on the unseen subset (U-N@50) and overall test set (O-N@50) for different GR backbones and drafter configurations on **Video Games** dataset.

Hyperparameter Analysis (RQ3)

We analyze how model hyperparameters affect recommendation behaviors and performance. We conduct hyperparameter analyses of **SpecGR++** on the **Video Games** dataset, with results shown in Figure 3. The base hyperparameters are $\delta = 50$, $\gamma = -1.6$, and $\beta = 50$, with specific parameters adjusted in each plot while keeping others fixed.

- **Draft size (δ)** controls the proportion of unseen items in recommendations, as all unseen items originate from drafting. Increasing draft size enhances inductive performance but may degrade in-sample metrics due to the fixed number of accepted candidates. The optimal value is selected via hyperparameter tuning on the validation set.
- **Beam size (β)** controls the search space for guided re-drafting. As shown in Figure 3, increasing beam size improves in-sample performance but reduces inductive ability.
- **Threshold (γ)** controls the acceptance rate of drafted candidates, impacting the number of decoding steps needed for K recommendations. As shown in Figure 3, it governs the performance-efficiency trade-off. Lower thresholds degrade in-sample performance due to overly easy candidate acceptance. We select γ using the elbow criterion that balances marginal performance gain against additional latency.

Plug-and-Play Framework (RQ4)

To evaluate SpecGR's plug-and-play capability, we integrate it with multiple inductive drafters (SemanticKNN, UniSRec, and the GR encoder (SpecGR++)) and multiple GR backbones (TIGER and DSI (Tay et al. 2022)). Notably, DSI employs hierarchical K-means tokenization to derive item semantic IDs. As shown in Table 4, both GR backbones are originally unable to generate unseen items. Integrating SpecGR improves overall performance by approximately 15% on average, while enabling strong inductive recommendation. This improvement holds across lightweight retrieval (Semantic-KNN), modality-based models (UniSRec), and self-drafting using the GR encoder, demonstrating that SpecGR is robust to different drafting paradigms and input modalities. Thus, SpecGR is model-agnostic to drafter choice and serves as a plug-and-play framework that equips any semantic-ID-based GR model with inductive capability.

Conclusion

In this paper, we propose SpecGR, a plug-and-play framework that extends the capability of generative recommendation models for inductive recommendation. Our method, inspired by speculative decoding, leverages an inductive model as a drafter to propose candidate items and uses the GR model as a verifier to ensure that only high-quality candidates are recommended. We further propose two drafting strategies: (1) using an auxiliary model for flexibility, and (2) using the GR model's own encoder for parameter-efficient self-drafting. Extensive experiments on three public datasets demonstrate strong inductive and overall recommendation performance for SpecGR.

Acknowledgments

This research was partially supported by the National Science Foundation (NSF) via the grant number IIS-2432486.

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Alshehri, M. A.; and Zhang, X. 2022. Generative adversarial zero-shot learning for cold-start news recommendation. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 26–36.

Anderson, P.; Fernando, B.; Johnson, M.; and Gould, S. 2016. Guided open vocabulary image captioning with constrained beam search. *arXiv preprint arXiv:1612.00576*.

Bao, K.; Zhang, J.; Zhang, Y.; Wang, W.; Feng, F.; and He, X. 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*, 1007–1014.

Cai, T.; Li, Y.; Geng, Z.; Peng, H.; Lee, J. D.; Chen, D.; and Dao, T. 2024. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*.

Cao, Y.; Hu, S.; Gong, Y.; Li, Z.; Yang, Y.; Liu, Q.; and Ji, S. 2022. Gift: Graph-guided feature transfer for cold-start video click-through rate prediction. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, 2964–2973.

Chen, C.; Borgeaud, S.; Irving, G.; Lespiau, J.-B.; Sifre, L.; and Jumper, J. 2023. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*.

Chen, H.; Wang, Z.; Huang, F.; Huang, X.; Xu, Y.; Lin, Y.; He, P.; and Li, Z. 2022. Generative adversarial framework for cold-start item recommendation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2565–2571.

Chen, S.; Yu, H.; Yagoobi, J.; and Shao, C. 2025. Reinforcement learning constrained beam search for parameter optimization of paper drying under flexible constraints. *arXiv preprint arXiv:2501.12542*.

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. 2020. A simple framework for contrastive learning of visual representations. In *ICML*, 1597–1607. PMLR.

Contal, E.; and McGoldrick, G. 2024. RAGSys: Item-Cold-Start Recommender as RAG System. *arXiv preprint arXiv:2405.17587*.

Covington, P.; Adams, J.; and Sargin, E. 2016. Deep neural networks for youtube recommendations. In *RecSys*.

Dai, S.; Shao, N.; Zhao, H.; Yu, W.; Si, Z.; Xu, C.; Sun, Z.; Zhang, X.; and Xu, J. 2023. Uncovering chatgpt's capabilities in recommender systems. In *Proceedings of the 17th ACM Conference on Recommender Systems*, 1126–1132.

De Bortoli, V.; Galashov, A.; Gretton, A.; and Doucet, A. 2025. Accelerated diffusion models via speculative sampling. *arXiv preprint arXiv:2501.05370*.

De Cao, N.; Izacard, G.; Riedel, S.; and Petroni, F. 2020. Autoregressive entity retrieval. *arXiv preprint arXiv:2010.00904*.

Deng, J.; Wang, S.; Cai, K.; Ren, L.; Hu, Q.; Ding, W.; Luo, Q.; and Zhou, G. 2025. Onerec: Unifying retrieve and rank with generative recommender and iterative preference alignment. *arXiv preprint arXiv:2502.18965*.

Di Palma, D. 2023. Retrieval-augmented recommender system: Enhancing recommender systems with large language models. In *Proceedings of the 17th ACM Conference on Recommender Systems*, 1369–1373.

Ding, H.; Ma, Y.; Deoras, A.; Wang, Y.; and Wang, H. 2021. Zero-shot recommender systems. *arXiv preprint arXiv:2105.08318*.

Dong, M.; Yuan, F.; Yao, L.; Xu, X.; and Zhu, L. 2020. Mamo: Memory-augmented meta-optimization for cold-start recommendation. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, 688–697.

Du, Y.; Zhu, X.; Chen, L.; Fang, Z.; and Gao, Y. 2022. Metakg: Meta-learning on knowledge graph for cold-start recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 35(10): 9850–9863.

Elhoushi, M.; Shrivastava, A.; Liskovich, D.; Hosmer, B.; Wasti, B.; Lai, L.; Mahmoud, A.; Acun, B.; Agarwal, S.; Roman, A.; et al. 2024. Layer skip: Enabling early exit inference and self-speculative decoding. *arXiv preprint arXiv:2404.16710*.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, 1126–1135. PMLR.

Freitag, M.; and Al-Onaizan, Y. 2017. Beam search strategies for neural machine translation. *arXiv preprint arXiv:1702.01806*.

Geng, S.; Liu, S.; Fu, Z.; Ge, Y.; and Zhang, Y. 2022. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *Proceedings of the 16th ACM Conference on Recommender Systems*, 299–315.

Gloeckle, F.; Idrissi, B. Y.; Rozière, B.; Lopez-Paz, D.; and Synnaeve, G. 2024. Better & faster large language models via multi-token prediction. *arXiv preprint arXiv:2404.19737*.

He, J.; Liu, R.; Zhuang, F.; Lin, F.; Niu, C.; and He, Q. 2018. A general cross-domain recommendation framework via Bayesian neural network. In *2018 IEEE International Conference on Data Mining (ICDM)*, 1001–1006. IEEE.

He, Z.; Xie, Z.; Jha, R.; Steck, H.; Liang, D.; Feng, Y.; Majumder, B. P.; Kallus, N.; and McAuley, J. 2023a. Large language models as zero-shot conversational recommenders. In *Proceedings of the 32nd ACM international conference on information and knowledge management*, 720–730.

He, Z.; Zhong, Z.; Cai, T.; Lee, J. D.; and He, D. 2023b. Rest: Retrieval-based speculative decoding. *arXiv preprint arXiv:2311.08252*.

Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; and Tikk, D. 2016. Session-based Recommendations with Recurrent Neural Networks. In *ICLR*.

Hou, Y.; He, Z.; McAuley, J.; and Zhao, W. X. 2023Learning Vector-Quantized Item Representation for Transferable Sequential Recommenders. In *WWW*.

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

Hou, Y.; Li, J.; Shin, A.; Jeon, J.; Santhanam, A.; Shao, W.; Hassani, K.; Yao, N.; and McAuley, J. 2025a. Generating long semantic ids in parallel for recommendation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, 956–966.

Hou, Y.; Mu, S.; Zhao, W. X.; Li, Y.; Ding, B.; and Wen, J.-R. 2022. Towards universal sequence representation learning for recommender systems. In *SIGKDD*.

Hou, Y.; Ni, J.; He, Z.; Sachdeva, N.; Kang, W.-C.; Chi, E. H.; McAuley, J.; and Cheng, D. Z. 2025b. ActionPiece: Contextually Tokenizing Action Sequences for Generative Recommendation. *arXiv preprint arXiv:2502.13581*.

Hou, Y.; Zhang, J.; Lin, Z.; Lu, H.; Xie, R.; McAuley, J.; and Zhao, W. X. 2024b. Large Language Models are Zero-Shot Rankers for Recommender Systems. In *ECIR*.

Kang, W.-C.; and McAuley, J. 2018. Self-attentive sequential recommendation. In *ICDM*.

Kim, J.; Kim, E.; Yeo, K.; Jeon, Y.; Kim, C.; Lee, S.; and Lee, J. 2024. Content-based Graph Reconstruction for Cold-start Item Recommendation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1263–1273.

Lee, H.; Im, J.; Jang, S.; Cho, H.; and Chung, S. 2019. Melu: Meta-learned user preference estimator for cold-start recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1073–1082.

Leviathan, Y.; Kalman, M.; and Matias, Y. 2023.Fast inference from transformers via speculative decoding. In *ICML*.

Li, J.; Wang, M.; Li, J.; Fu, J.; Shen, X.; Shang, J.; and McAuley, J. 2023. Text is all you need: Learning language representations for sequential recommendation. In *SIGKDD*, 1258–1267.

Lin, X.; Wu, J.; Zhou, C.; Pan, S.; Cao, Y.; and Wang, B. 2021. Task-adaptive neural process for user cold-start recommendation. In *Proceedings of the Web Conference 2021*, 1306–1316.

Lin, X.; Yang, C.; Wang, W.; Li, Y.; Du, C.; Feng, F.; Ng, S.-K.; and Chua, T.-S. 2024. Efficient Inference for Large Language Model-based Generative Recommendation. *arXiv preprint arXiv:2410.05165*.

Liu, J.; Liu, C.; Zhou, P.; Lv, R.; Zhou, K.; and Zhang, Y. 2023a. Is chatgpt a good recommender? a preliminary study. *arXiv preprint arXiv:2304.10149*.

Liu, S.; Ounis, I.; Macdonald, C.; and Meng, Z. 2020. A heterogeneous graph neural model for cold-start recommendation. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, 2029–2032.

Liu, T.; Gao, C.; Wang, Z.; Li, D.; Hao, J.; Jin, D.; and Li, Y. 2023b. Uncertainty-aware Consistency Learning for Cold-Start Item Recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2466–2470.

Liu, W.; Zheng, X.; Su, J.; Zheng, L.; Chen, C.; and Hu, M. 2023c. Contrastive proxy kernel stein path alignment for cross-domain cold-start recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 35(11): 11216–11230.

Liu, Z.; Hou, Y.; and McAuley, J. 2024. Multi-Behavior Generative Recommendation. In *CIKM*.

Muennighoff, N.; Su, H.; Wang, L.; Yang, N.; Wei, F.; Yu, T.; Singh, A.; and Kiela, D. 2024. Generative representational instruction tuning. *arXiv preprint arXiv:2402.09906*.

Ni, J.; Abrego, G. H.; Constant, N.; Ma, J.; Hall, K.; Cer, D.; and Yang, Y. 2022a. Sentence-T5: Scalable Sentence Encoders from Pre-trained Text-to-Text Models. In *Findings of ACL*.

Ni, J.; Qu, C.; Lu, J.; Dai, Z.; Abrego, G. H.; Ma, J.; Zhao, V.; Luan, Y.; Hall, K.; Chang, M.-W.; et al. 2022b. Large Dual Encoders Are Generalizable Retrievers. In *EMNLP*.

Nogueira, R.; Lin, J.; and Epistemic, A. 2019. From doc2query to docTTTTTquery. *Online preprint*, 6(2).

Pazzani, M. J.; and Billsus, D. 2007. Content-based recommendation systems. In *The adaptive web: methods and strategies of web personalization*, 325–341. Springer.

Post, M.; and Vilar, D. 2018. Fast Lexically Constrained Decoding with Dynamic Beam Allocation for Neural Machine Translation.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140): 1–67.

Rajput, S.; Mehta, N.; Singh, A.; Hulikal Keshavan, R.; Vu, T.; Heldt, L.; Hong, L.; Tay, Y.; Tran, V.; Samost, J.; et al. 2024. Recommender systems with generative retrieval. In *NeurIPS*.

Sheng, L.; Zhang, A.; Zhang, Y.; Chen, Y.; Wang, X.; and Chua, T.-S. 2024. Language Models Encode Collaborative Signals in Recommendation. *arXiv preprint arXiv:2407.05441*.

Singh, A. P.; and Gordon, G. J. 2008. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 650–658.

Sun, F.; Liu, J.; Wu, J.; Pei, C.; Lin, X.; Ou, W.; and Jiang, P. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. In *CIKM*.

Sun, R.; Li, X.; Akella, A.; and Konstan, J. A. 2024. Large Language Models as Conversational Movie Recommenders: A User Study. *arXiv preprint arXiv:2404.19093*.

Tang, J.; Wu, S.; Sun, J.; and Su, H. 2012. Cross-domain collaboration recommendation. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 1285–1293.

Tay, Y.; Tran, V.; Dehghani, M.; Ni, J.; Bahri, D.; Mehta, H.; Qin, Z.; Hui, K.; Zhao, Z.; Gupta, J.; et al. 2022. Transformer memory as a differentiable search index. *Advances in Neural Information Processing Systems*, 35: 21831–21843.

Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Volkovs, M.; Yu, G.; and Poutanen, T. 2017. Dropoutnet: Addressing cold start in recommender systems. *Advances in neural information processing systems*, 30.

Wang, T.; Zhuang, F.; Zhang, Z.; Wang, D.; Zhou, J.; and He, Q. 2021. Low-dimensional alignment for cross-domain recommendation. In *Proceedings of the 30th ACM international conference on information & knowledge management*, 3508–3512.

Wang, W.; Chen, B.; Liu, B.; Wang, X.; Yang, L.; Jiang, W.; Ning, W.; and Guan, J. 2024a. Mutual Information Assisted Graph Convolution Network for Cold-Start Recommendation. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 6785–6789. IEEE.

Wang, Y.; Yao, Q.; Kwok, J. T.; and Ni, L. M. 2020. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3): 1–34.

Wang, Z.; Wang, Z.; Le, L.; Zheng, H. S.; Mishra, S.; Perot, V.; Zhang, Y.; Mattapalli, A.; Taly, A.; Shang, J.; et al. 2024b. Speculative rag: Enhancing retrieval augmented generation through drafting. *arXiv preprint arXiv:2407.08223*.

Wei, Y.; Wang, X.; Li, Q.; Nie, L.; Li, Y.; Li, X.; and Chua, T.-S. 2021. Contrastive learning for cold-start recommendation. In *Proceedings of the 29th ACM International Conference on Multimedia*, 5382–5390.

Wu, J.; Chang, C.-C.; Yu, T.; He, Z.; Wang, J.; Hou, Y.; and McAuley, J. 2024. Coral: collaborative retrieval-augmented large language models improve long-tail recommendation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 3391–3401.

Wu, L.; Yang, Y.; Chen, L.; Lian, D.; Hong, R.; and Wang, M. 2020a. Learning to transfer graph embeddings for inductive graph based recommendation. In *SIGIR*, 1211–1220.

Wu, Q.; Zhang, H.; Gao, X.; Yan, J.; and Zha, H. 2021. Towards open-world recommendation: An inductive model-based collaborative filtering approach. In *ICML*, 11329–11339. PMLR.

Wu, T.; Chio, E. K.-I.; Cheng, H.-T.; Du, Y.; Rendle, S.; Kuzmin, D.; Agarwal, R.; Zhang, L.; Anderson, J.; Singh, S.; et al. 2020b. Zero-shot heterogeneous transfer learning from recommender systems to cold-start search retrieval. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, 2821–2828.

Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; and Philip, S. Y. 2020c. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1): 4–24.

Wu, Z.; and Zhou, X. 2023. M2eu: Meta learning for cold-start recommendation via enhancing user preference estimation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1158–1167.

Xiao, B.; Shi, C.; Nie, X.; Yang, F.; Deng, X.; Su, L.; Chen, W.; and Cui, B. 2024. Clover: Regressive Lightweight Speculative Decoding with Sequential Knowledge. *arXiv preprint arXiv:2405.00263*.

Xie, R.; Liu, Q.; Wang, L.; Liu, S.; Zhang, B.; and Lin, L. 2022. Contrastive cross-domain recommendation in matching. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, 4226–4236.

Yang, L.; Paischer, F.; Hassani, K.; Li, J.; Shao, S.; Li, Z. G.; He, Y.; Feng, X.; Noorshams, N.; Park, S.; et al. 2024. Unifying generative and dense retrieval for sequential recommendation. *arXiv preprint arXiv:2411.18814*.

Yuan, Z.; Yuan, F.; Song, Y.; Li, Y.; Fu, J.; Yang, F.; Pan, Y.; and Ni, Y. 2023. Where to go next for recommender systems? id-vs. modality-based recommender models revisited. In *SIGIR*, 2639–2649.

Zhai, J.; Liao, L.; Liu, X.; Wang, Y.; Li, R.; Cao, X.; Gao, L.; Gong, Z.; Gu, F.; He, M.; et al. 2024. Actions speak louder than words: Trillion-parameter sequential transducers for generative recommendations. In *ICML*.

Zhang, C.; Zhang, H.; Wu, S.; Wu, D.; Xu, T.; Gao, Y.; Hu, Y.; and Chen, E. 2024. NoteLLM-2: Multimodal Large Representation Models for Recommendation. *arXiv preprint arXiv:2405.16789*.

Zhang, G.; Hou, Y.; Lu, H.; Chen, Y.; Zhao, W. X.; and Wen, J.-R. 2023. Scaling Law of Large Sequential Recommendation Models. *arXiv preprint arXiv:2311.11351*.

Zhang, T.; Zhao, P.; Liu, Y.; Sheng, V. S.; Xu, J.; Wang, D.; Liu, G.; Zhou, X.; et al. 2019. Feature-level deeper self-attention network for sequential recommendation. In *IJCAI*.

Zhang, W.; Bei, Y.; Yang, L.; Zou, H. P.; Zhou, P.; Liu, A.; Li, Y.; Chen, H.; Wang, J.; Wang, Y.; et al. 2025. Cold-Start Recommendation towards the Era of Large Language Models (LLMs): A Comprehensive Survey and Roadmap. *arXiv preprint arXiv:2501.01945*.

Zhao, W. X.; Mu, S.; Hou, Y.; Lin, Z.; Chen, Y.; Pan, X.; Li, K.; Lu, Y.; Wang, H.; Tian, C.; et al. 2021. Recbole: Towards a unified, comprehensive and efficient framework for recommendation algorithms. In *CIKM*, 4653–4664.

Zhao, W. X.; Zhou, K.; Li, J.; Tang, T.; Wang, X.; Hou, Y.; Min, Y.; Zhang, B.; Zhang, J.; Dong, Z.; et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Zheng, B.; Hou, Y.; Lu, H.; Chen, Y.; Zhao, W. X.; and Wen, J.-R. 2024. Adapting Large Language Models by Integrating Collaborative Semantics for Recommendation. In *ICDE*.

Zheng, J.; Ma, Q.; Gu, H.; and Zheng, Z. 2021a. Multi-view denoising graph auto-encoders on heterogeneous information networks for cold-start recommendation. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*, 2338–2348.

Zheng, Y.; Liu, S.; Li, Z.; and Wu, S. 2021b. Cold-start sequential recommendation via meta learner. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 4706–4713.

Zhou, K.; Wang, H.; Zhao, W. X.; Zhu, Y.; Wang, S.; Zhang, F.; Wang, Z.; and Wen, J.-R. 2020. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *CIKM*, 1893–1902.

Zhou, Z.; Zhang, L.; and Yang, N. 2023. Contrastive collaborative filtering for cold-start item recommendation. In *Proceedings of the ACM Web Conference 2023*, 928–937.

Zhu, F.; Chen, C.; Wang, Y.; Liu, G.; and Zheng, X. 2019. Dtcdr: A framework for dual-target cross-domain recommendation. In *Proceedings of the 28th ACM international conference on information and knowledge management*, 1533–1542.

Zhu, F.; Wang, Y.; Chen, C.; Zhou, J.; Li, L.; and Liu, G. 2021. Cross-domain recommendation: challenges, progress, and prospects. *arXiv preprint arXiv:2103.01696*.

Zhu, Z.; Sefati, S.; Saadatpanah, P.; and Caverlee, J. 2020. Recommendation for new users and new items via randomized training and mixture-of-experts transformation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1121–1130.

Zhuang, S.; Li, H.; and Zuccon, G. 2021. Deep query likelihood model for information retrieval. In *Advances in Information Retrieval: 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28–April 1, 2021, Proceedings, Part II 43*, 463–470. Springer.

Zhuang, S.; Liu, B.; Koopman, B.; and Zuccon, G. 2023. Open-source large language models are strong zero-shot query likelihood models for document ranking. *arXiv preprint arXiv:2310.13243*.

Additional Related Work: Cold-Start Recommendation

The item cold-start problem (ICS) (Zhang et al. 2025), a longstanding challenge in recommender systems, requires models to recommend newly added items with limited or no historical interaction data, and therefore external knowledge have to leveraged. In what follows, we review existing works based on four major external sources: (1) content features (2) graph relations (3) cross-domain information and (4) world knowledge from LLMs.

Content features Due to the lack of interaction records, early works naturally leveraged content features (e.g., titles, descriptions, and categories) to address cold-start recommendation. In scenarios with a few interactions, studies focused on data-efficient learning to quickly improve cold-start performance. Meta-learning (Wang et al. 2020) is a widely adopted approach that pretrains models on warm interactions to capture general interaction patterns and then adapts parameters to new cold-start items using limited data. Early works achieved this through gradient-based optimization for rapid parameter adaptation (Lee et al. 2019; Finn, Abbeel, and Levine 2017; Dong et al. 2020). Later approaches incorporated task relationships to better transfer warm knowledge to cold-start scenarios using techniques such as soft-clustering (Lin et al. 2021) and domain-specific neural architectures (Wu and Zhou 2023). In more challenging ICS scenarios with no interaction data, studies focus on deriving content-based item representations while still maintaining strong performance on warm items. Early works explored perturbation training strategies to reduce reliance on interactions and encourage models to learn generalized recommendations solely from content features (Volkovs, Yu, and Poutanen 2017; Zhu et al. 2020). Others formulated item representation learning as an explicit knowledge alignment task, bridging warm interaction-based and content-based representations through methods such as contrastive learning (Wei et al. 2021; Zhou, Zhang, and Yang 2023) and generative adversarial networks (GANs) (Chen et al. 2022; Alshehri and Zhang 2022).

Graph relations In recent years, graph neural networks (Wu et al. 2020c) have gained attention for modeling user-item interaction graphs in recommendation, with its power in modeling higher-order relationships beyond direct interactions and content features. However, a key challenge remains in effectively propagating information to cold-start items, where interaction signals are sparse. One typical methods trains the model to infer probable edges on the original use item interaction graph for cold-start items to enable information aggregation. Typical training tasks involves masking and reconstruction (Kim et al. 2024), maximizing mutual information (Wang et al. 2024a), and uncertainty estimation (Liu et al. 2023b). On the other hand, another stream of work enrich the existing interaction graph with more fine-grained connections to propagate information to cold-start items. The constructed edges includes semantic links (Cao et al. 2022; Liu et al. 2020), multi-view feature extractions (Zheng et al. 2021a), and knowledge relations (Du et al. 2022).

Cross-domain information To mitigate data sparsity for cold-start item recommendation, researchers have explored using cross-domain interactions to learn generalizable patterns and transfer knowledge from data-rich domains (Zhu et al. 2019, 2021; Xie et al. 2022). However, a key challenge lies in designing effective transfer techniques that account for discrepancies between domains. Early works primarily relied on overlapping information between the source and target domain, including items (Singh and Gordon 2008; Zhu et al. 2021) or attributes (Tang et al. 2012). To address distributional differences, distribution alignment (Liu et al. 2023c; Wang et al. 2021) and invariant representation learning (Wu et al. 2020b; He et al. 2018) has been proposed to build robust item representations to facilitate knowledge transfer. Recently, with the rise of pretrained language models (PLMs), text representations have proven to be powerful semantic bridges, reducing the need for explicit entity overlaps between domains (Ding et al. 2021; Geng et al. 2022; Hou et al. 2022). To learn universal representations from text, many approaches leverage pretraining across multiple domains, followed by fine-tuning on domain-specific tasks to enable seamless cross-domain transfer (Hou et al. 2022, 2023; Li et al. 2023).

World knowledge from LLMs In recent years, the development of Large Language Models (LLMs) (Zhao et al. 2023; Achiam et al. 2023; Touvron et al. 2023) has enabled their application across a wide range of recommendation scenarios due to their well-learned world knowledge from web-scale internet data, which allow them to easily understand cold-start items. One direct application, the 'LLM for Rec' methods, mainly involves prompting (Liu et al. 2023a; Dai et al. 2023; Hou et al. 2024b), finetuning (Zhang et al. 2024; Bao et al. 2023), retrieval-augmented generation (RAG) (Di Palma 2023; Contal and McGoldrick 2024; Wu et al. 2024), and conversational dialogue systems (He et al. 2023a; Sun et al. 2024). While these approaches require minimal pre-training, they suffer from high inference latency and hallucinations. To mitigate these issues, recent methods use LLMs to generate modality-based item representations(Hou et al. 2022; Li et al. 2023; Hou et al. 2023) and train backbone models on these frozen representations, achieving state-of-the-art performance in cold-start scenarios.

Additional Implementation Details

SpecGR++ Training Details

We train SpecGR++ with Distributed Data Parallel (DDP) using 4 GPUs (NVIDIA RTX A6000, 48GB each, Ubuntu 22.04.5 LTS). The random seed is fixed to 42 for reproducibility. During pretraining, we compute the generation loss L_{Gen} with a batch size of 256, and the contrastive loss L_{CL} with an effective batch size of 2048 (negatives gathered across GPUs). We train for up to 200,000 steps with early stopping, selecting learning rates from $\{0.001, 0.0003\}$ For fine-tuning, SpecGR++ is trained for 15 epochs with a batch size of 256 and a learning rate of 10^{-4}. The draft threshold

Table A1: Average validation Recall@50 for SpecGR++ across different training embedding batch sizes.

Batch Size	Avg. R@50
1024	0.0598
2048	0.0692
4096	0.0769

γ is tuned between -1.1 and -1.8 (step size 0.1), and beam size β and draft size δ are set to 50 or K. For SpecGR$_{\text{Aux}}$, it is trained on a single GPU with the same hyperparameters but without distributed negative gathering.

Embedding Batch Size Ablation

During SpecGR++ training, to ensure the model receives sufficient negative training signals, we leverage much larger batch sizes for embedding task than generative task. We report the performance against the effective batch size, which equals the batch size $\times$ number of gpus in the distributed setting Table A1. We can see that larger embedding batch size positively affects the performance of embeddings from SpecGR's encoder.

Baseline Model Reproductions

We reproduce TIGER by following the instructions provided in its original implementation (Rajput et al. 2024). For Recformer, we utilize pretrained checkpoints from a popular reproduction repository[2] and fine-tune them on our processed datasets. We implement LIGER by closely following the method details, pseudocode, and hyperparameter configurations outlined in the original paper (Lin et al. 2024). For item text embeddings, we use representations obtained from a pretrained T5 sentence encoder. All other baseline models (e.g., SASRec variants and modality-based methods) are implemented using the open-source RecBole library (Zhao et al. 2021). We conduct thorough hyperparameter tuning for all baseline models and report results using the best-performing configuration for fair comparison.

TIGER$_{\text{C}}$ Implementation

Following Rajput et al. (2024), during inference, we use TIGER to generate top-K candidates and retrieve unseen items whose semantic prefix (*i.e.,* semantic IDs without the last identifier token) appears in this list. A hyperparameter ϵ controls the maximum proportion of unseen items included. After retrieving x unseen candidates, where $x <= \epsilon \cdot K$, we append them after the first $K - x$ generated candidates to finalize the recommendation. We tune ϵ on each dataset and report the best results in Table 3.

SpecGR++ Training Details

Case Study: Guided Redrafting

From Table 3, guided re-drafting leverages the generative backbone to refine redrafted candidates, enhancing overall performance. To analyze its mechanism and impact, we conduct a case study under a fixed threshold ($\gamma = -1.4$). First, Table A3 reports the comparison of acceptance rate per decoding step between naive drafting (w/o) and guided re-drafting (w). Under a strict threshold, we can see that the step-1 acceptance rate is low, highlighting the need for re-drafting. Guided re-drafting consistently increases the number of accepted items in later steps, improving verification efficiency. Furthermore, Table A4 presents the impact of guided re-drafting on overall and subset-specific performance under a fixed threshold. As shown, guided re-drafting yields improvements across all categories, particularly in the unseen subset, where recall and NDCG scores rise by approximately 10%. This validates that leveraging redrafted tokens to constrain the drafting scope results in higher-quality candidates in subsequent steps.

Table A2: Statistics of the datasets. "New%" denotes the proportion of interactions with unseen target items. "#Items" and "#Inter." are in thousands (K).

Dataset	Items		Train	Valid		Test	
	#Items	New%	#Inter.	#Inter.	New%	#Inter.	New%
Games	25.6	10.3	645.3	33.1	27.9	41.5	60.3
Office	77.6	15.1	1230.2	136.1	16.2	211.3	59.4
Phones	111.5	15.1	1841.5	232.9	33.0	297.4	68.3

Table A3: Acceptance rate (%) per decoding step with and without guided re-drafting when $\gamma = -1.4$.

Step	1	2	3	4
w/o	10.34	5.93	4.25	2.63
w	10.34	6.62	5.51	5.75

Ablation: Comparison with Ensemble Methods

We compare SpecGR$_{\text{Aux}}$ with ensemble-based variants combining GR and modality-based models, as shown in Table A5. The *Score-based Ensemble* linearly combines TIGER's likelihood scores and UniSRec's ranking scores. The *Ranking-based Ensemble* averages item ranking positions from both models to mitigate score scale mismatches. The *2-Stage Ensemble* selects top-K items using UniSRec and re-ranks them with TIGER's scores. All ensemble methods have unseen performance bounded by the two base models, TIGER and UniSRec, since simple score aggregation or re-ranking does not introduce inductive capabilities. In contrast, SpecGR uses guided re-drafting to effectively leverage the backbone's modeling capability for enhancing unseen candidates' quality. The results illustrate the effectiveness of the proposed SpecGR methods.

Inference Speed Acceleration

Generative recommendation (GR) models rely on autoregressive next-token generation, which incurs high inference latency. SpecGR addresses this by employing speculative

[2]https://github.com/AaronHeee/RecFormer

Table A4: Performance comparison with and without guided re-drafting when $\gamma = -1.4$.

	Overall		Unseen		In-Sample	
	R@50	**N@50**	**R@50**	**N@50**	**R@50**	**N@50**
w/o	0.0702	0.0218	0.0288	0.0076	0.1331	0.0434
w	**0.0721**	**0.0223**	**0.0317**	**0.0082**	**0.1333**	**0.0436**

Table A5: Performance comparison of SpecGR $_{\text{Aux}}$ (UniSRec drafter) against ensemble variants. The best and second-best scores are bolded and underlined.

Model	**Overall**		**Unseen**		**In-Sample**	
	R@50	**N@50**	**R@50**	**N@50**	**R@50**	**N@50**
Single Model						
TIGER	0.0584	0.0193	–	–	0.1472	0.0486
UniSRec	0.0621	0.0200	0.0118	0.0029	0.1386	0.0461
Ensemble Method						
Score-based	0.0571	0.0191	0.0050	0.0009	0.1333	0.0456
Ranking-based	0.0678	0.0218	0.0056	0.0011	**0.1624**	**0.0532**
2-Stage	0.0621	0.0205	0.0118	0.0026	0.1386	0.0477
SpecGR (UniSRec)	**0.0778**	**0.0239**	**0.0312**	**0.0096**	0.1485	0.0457

retrieval-based drafting, where candidate items proposed by the drafter can be efficiently verified by the GR verifier, significantly reducing the number of required autoregressive decoding steps.

Drafting and Verification Latency

To clearly illustrate SpecGR's efficiency, we report detailed inference latency results (in seconds) separated into drafting (D) and verification (V) stages across various draft sizes. As shown in Table A6, SpecGR++ achieves remarkably low drafting latency due to its efficient encoder-based drafting mechanism, demonstrating significant overall speed improvements compared to TIGER.

Table A6: Empirical inference latency (seconds) across drafting (D) and verification (V) stages.

Draft Size	**20 (D/V)**	**50 (D/V)**	**100 (D/V)**
SpecGR (UniSRec)	0.0110 / 0.0261	0.0112 / 0.0249	0.0112 / 0.0227
SpecGR++	**0.0003** / 0.0261	**0.0003** / 0.0249	**0.0003** / 0.0235
TIGER	- / 0.0403	- / 0.0403	- / 0.0403

Acceleration w.r.t. Generation Length

We analyze how SpecGR reduces inference latency as the generation length increases. Specifically, we illustrate the acceleration factor w.r.t. the number of digits in semantic IDs. We train SpecGR++ with a TIGER backbone across different semantic ID lengths (ranging from 2 to 16 digits) and report the acceleration factor relative to the TIGER baseline. The acceleration factor is defined as the inverse ratio of inference time. As shown in Figure A1, in standard GR settings with 4-digit semantic IDs, SpecGR achieves a 1.7× inference speedup while maintaining strong overall performance (see the Experiments section). Moreover, longer semantic IDs lead to even higher acceleration as fewer decoding steps are needed per item.

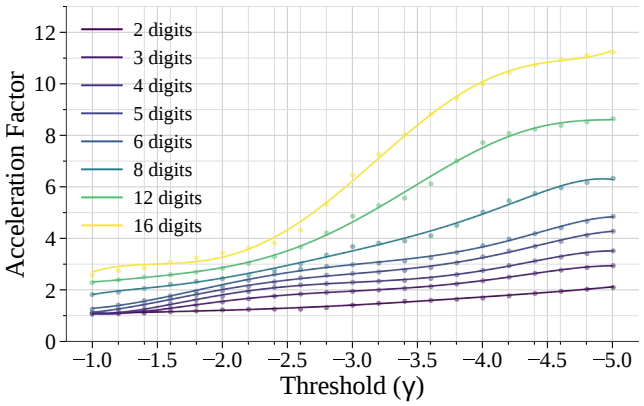

Figure A1: Inference speed acceleration factor w.r.t. different numbers of semantic ID digits.

Theoretical Time Complexity

We further analyze the expected time complexity of SpecGR, where each item is represented by l semantic tokens. Let K be the number of items to recommend, δ the draft size per iteration, and p the acceptance probability of a drafted item. We denote the runtime for a single forward pass in the GR encoder and decoder as $C_{encoder}$ and $C_{decoder}$, respectively. A standard GR model (e.g., TIGER) must decode all l tokens for K beams, yielding a complexity:

$$\text{Cost}_{\text{TIGER}} = O(C_{encoder} + l \cdot C_{decoder}).$$

For SpecGR, inductive drafting requires one forward pass and a KNN search, denoted as C_{draft}. Verification and beam search happen in parallel with a cost of $C_{decoder}$ per iteration. The expected number of iterations to obtain K valid recommendations is bounded by:

$$T = O\left(\frac{K}{\delta p}\right) \leq O(l).$$

Thus, SpecGR's expected time complexity is:

$$\text{Cost}_{\text{SpecGR}} = O(C_{draft} + C_{encoder} + T \cdot C_{decoder}),$$

with an expected acceleration factor:

$$\frac{C_{encoder} + l \cdot C_{decoder}}{C_{draft} + C_{encoder} + \frac{K}{\delta p} \cdot C_{decoder}}.$$

Empirical Acceptance Rates and Speedup

We empirically measure the acceptance probability p for various drafting methods and report their corresponding acceleration factors in Table A7. For the SemanticKNN drafter, item sequences are encoded using mean text embeddings from the most recent five items, and top-K items are retrieved via KNN search. Across all methods, SpecGR consistently outperforms traditional GR models.

Table A7: Empirically measured acceptance rate p and expected acceleration factor for different drafters at $\gamma = -1.4$.

Drafter	Acceptance Rate (p)	Acceleration Factor
SpecGR++ Encoder	0.44	1.72
UniSRec	0.35	1.38
SemanticKNN	0.20	1.11

Scalability Analysis

SpecGR exhibits favorable scalability in multiple dimensions compared to traditional GR models:

- **Model Size.** As the GR model scales, assuming $C_{encoder} \approx C_{decoder}$, SpecGR maintains a constant acceleration factor $\frac{l+1}{\frac{K}{\delta p}+1}$, demonstrating robust scalability.
- **Semantic ID Length.** The acceleration factor grows linearly as the semantic ID length increases, i.e., $O\left(\frac{l \cdot \delta \cdot p}{K}\right)$, highlighting SpecGR's advantage in scenarios requiring long sequences, as quantitatively shown in Figure A1.

Additional Further Analysis

We briefly discuss SpecGR's capabilities compared to traditional methods. Leveraging the draft-then-verify framework, SpecGR seamlessly integrates into diverse recommendation scenarios, adapting to dynamic needs (Table A9). For instance, when deployed as a ranker model, it exploits autoregressive generation for efficient subset ranking. Additionally, its tunable hyperparameters enable real-time controllable inductive ability and controllable inference speed. See the section Other New Capabilities for broader implications.

Subset Ranking

In this section, we assess new capability of SpecGR in efficient subset ranking compared to traditional subset ranking methods for GR models. Since most deep learning-based recommendation methods unavoidably introduce high latency, they are typically used as ranking models rather than retrieval models (Covington, Adams, and Sargin 2016; Hou et al. 2024b). In the subset ranking setting, the model of interest is applied as a ranker to rank a given subset of items ($I_r \subset I, |I_r| \ll |I|$). Subset ranking is out-of-the-box for traditional sequential recommendation models by selecting a subset of candiates when performing KNN search. However, for generative recommendation models, subset ranking presents a challenge as they inherently recommend items by searching for the top$K$ decoding paths across the entire item space.

Batch scoring (BS) A simple approach to address this is through batch scoring, which involves splitting the item subset into fixed-size batches and scoring them consecutively with the generative model. However, this method grows linearly with batch size and is impractical for large subsets, as shown in Figure A2.

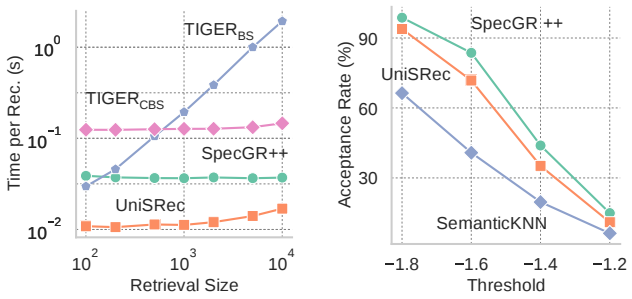

Figure A2: (Left) Inference latency comparison for subset ranking. Both x- and y-axis use log scale. (Right) Acceptance rate comparison for different drafting strategies.

Table A8: Comparison of different models based on parameter efficiency and training time.

Model	Trainable (M)			Non-trainable (M)	Training Time (h)
	Total	Non-emb	Emb		
$\text{SASRec}_{\text{ID}}$	7.24	0.10	7.13	0	3.6
UniSRec	2.90	2.90	0	85.62	18.3
Recformer	233.73	106.32	127.41	0	226.0
TIGER	13.26	13.11	0.15	0	16.2
TIGER_{C}	13.26	13.11	0.15	0	16.2
$\text{SpecGR}_{\text{Aux}}$	16.16	16.02	0.15	85.62	34.5
SpecGR++	13.28	13.13	0.15	14.27	42.8

Constrained beam search (CBS) An enhanced method is to use constrained beam search (Anderson et al. 2016; Post and Vilar 2018; De Cao et al. 2020). Constrained beam search constructs a trie using all allowed prefixes to restrict the search space at each decoding step. However, this approach introduces significant computational overhead. The time complexity for CBS is $O(T \cdot n_b \cdot 2^C)$, where T is the sequence length, n_b is the beam size, and C is the number of constraints (Chen et al. 2025). The large computational overhead and exponential growth with the number of constraints makes it inefficient for large-scale ranking tasks, as shown quantitatively in Figure A2.

SpecGR for subset ranking SpecGR effectively addresses this issue by restricting the drafter model's range to a specified subset. This ensures that all recommendations originate from within the subset. As demonstrated in Figure A2, SpecGR achieves a $3.5\times$ speedup for subset sizes $< 10^4$ compared to TIGER with constrained beam search (denoted as CBS). Moreover, SpecGR maintains a time complexity that is bounded by its full ranking complexity as the retrieval size increases, making it a highly efficient solution for subset ranking tasks.

Other New Capabilities

As a direct impact of the adjustable hyperparameters analyzed in the Experiments section, SpecGR possesses two other new capabilities: controllable inductive ability and controllable inference speed.

Table A9: Comparison of SpecGR against existing models across different scenarios and model capabilities.

Model	Recommendation Scenario		Model Capability		
	Efficient Subset Ranking	Inductive Recommendation	Auto-regressive Generation	Controllable Inductive Ability	Controllable Inference Speed
$\text{SASRec}_{\text{ID}}$	"	%	%	%	%
UniSRec	"	"	%	%	%
Recformer	"	"	%	%	%
TIGER	%	%	"	%	%
TIGER_{C}	%	"	"	"	%
$\text{SpecGR}_{\text{Aux}}$	"	"	"	"	"
SpecGR++	"	"	"	"	"

Controllable inductive ability allows platforms to dynamically adjust their recommendation strategy based on seasonal demand, favoring new items during certain periods while prioritizing established products at others. For instance, e-commerce platforms could increase the inductive ability of SpecGR to promote new items, and reduce it during clearance sales.

Controllable inference speed enables platforms to trade off model performance for faster inference speed during high-traffic periods, ensuring responsive user experiences.

In summary, as shown in Table A9, SpecGR inherits the architecture of generative recommendation, allowing effective scaling on large datasets. It also extends high performance and low inference speed to broader real-life recommendation settings, adapting to specific data characteristics and recommendation needs.

Further discussion on SpecGR++ Architecture

SpecGR++ utilizes its encoder as the self-drafter, effectively eliminating the need for maintaining a separate draft model for drafting, resulting in a more integrated method during inference. In this section, we will study the additional advantages of the SpecGR++ compared to the SpecGR with an auxiliary model.

Parameter Efficiency and Speed

First, SpecGR++ uses intermediate encoder outputs for drafting, resulting in nearly no additional computational cost and parameter sizes compared to GR. We report the total number of parameters and training time required for different methods in Table A8. As we can see, SpecGR inherents GR's advantages for scaling for the larger dataset as it assigns most of the parameters into non-embedding layers. Due to the additional embedding training tasks, SpecGR++ training time is slightly longer than training a drafter model and TIGER, and is 2.6x more GPU hours than training a TIGER. However, we believe that it is worthwhile to consider the acceleration during inference time. We've also provided a distributed training implementation in the released code.

Unified Representation Space

Because both the drafter and verifier use the encoder's representation, we observe a higher acceptance rate for the self-drafter compared to drafting with auxiliary models. This leads to better recommendation efficiency, where less decoding step is required.

Notably, we also observe a slight increase in generative performance after SpecGR++ pretraining compared to generation-only training (*e.g.,* TIGER). Recent studies have shown similar results, indicating that with a unified representation space, a model can maintain high performance in both generative and embedding tasks in NLP (Muennighoff et al. 2024). Our study further confirms that generation and representation are not conflicting tasks in the recommendation setting but rather two complementary approaches to solving the same problem. We look forward to seeing future research in recommendation systems that explores the unification and overlap between generative recommendation and representational recommendation.