Uncovering Inference Computation Scaling for Feature Augmentation in Recommendation Systems

Weihao Liu
Renmin University of China
Beijing, China
weihaoliu@ruc.edu.cn

Zhaocheng Du
Huawei Noah's Ark Lab
Shenzhen, China
zhaochengdu@huawei.com

Haiyuan Zhao
Renmin University of China
Beijing, China
haiyuanzhao@ruc.edu.cn

Wenbo Zhang
Renmin University of China
Beijing, China
zhangwenbo@ruc.edu.cn

Xiaoyan Zhao
The Chinese University of Hong Kong
Hong Kong, China
xzhao@se.cuhk.edu.hk

Gang Wang
Huawei Noah's Ark Lab
Shenzhen, China
wanggang110@huawei.com

Zhenhua Dong
Huawei Noah's Ark Lab
Shenzhen, China
dongzhenhua@huawei.com

Xiao Zhang
Renmin University of China
Beijing, China
zhangx89@ruc.edu.cn

Jun Xu*
Renmin University of China
Beijing, China
junxu@ruc.edu.cn

Abstract

Large language models have emerged as powerful tools for feature augmentation in recommendation systems. However, existing methods relying on quick inference often suffer from *incomplete feature coverage* and *insufficient specificity* in feature descriptions, making it difficult to capture fine-grained user preferences and weakening overall performance. Inspired by the success of *inference scaling* in math and coding tasks, we explore whether scaling inference can address these limitations and enhance feature quality.

Our experiments show that scaling inference leads to significant improvements in recommendation performance, with a 12% increase in NDCG@10. The gains can be attributed to two key factors: feature quantity and specificity. In particular, models using extended Chain-of-Thought (CoT) reasoning generate a greater number of detailed and precise features, offering deeper insights into user preferences and overcoming the limitations of quick inference. We further investigate the factors influencing feature quantity, revealing that model choice and search strategy play critical roles in generating a richer and more diverse feature set. To our knowledge, this is the first work to apply inference scaling to feature augmentation in recommendation systems, bridging advances in reasoning tasks to enhance personalized recommendation.

ACM Reference Format:
Weihao Liu, Zhaocheng Du, Haiyuan Zhao, Wenbo Zhang, Xiaoyan Zhao, Gang Wang, Zhenhua Dong, Xiao Zhang, and Jun Xu. 2025. Uncovering Inference Computation Scaling for Feature Augmentation in Recommendation Systems. In *Proceedings of Proceedings of the Nineteenth ACM Conference on Recommender Systems (RecSys '25 Workshop).* ACM, New York, NY, USA, 11 pages. https://doi.org/XXXXXXX.XXXXXXX

*Jun Xu is the corresponding author.

RecSys '25 Workshop, Prague, Czech Republic

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Large language models (LLMs) have recently become a prevalent and effective approach to feature augmentation in recommendation systems [35, 37, 86, 87]. By leveraging their generative capabilities, LLMs can produce a range of item [1, 49] or user-level [65, 66] features—ranging from high-level product attributes [68] to nuanced decision factors [56]—thereby enhancing recommendation performance[1]. However, most existing methods rely on quick inference (also known as System-1 thinking) [30] to generate these features, which often leads to issues such as incomplete feature coverage and insufficient specificity [29]. As a result, these fast-thinking models lack depth analysis and miss some critical dimensions of user preferences, so the resulting features may fall short in capturing the subtle decision-making factors that differentiate preferred items from disliked ones.

A concrete example illustrates this limitation: when using a standard LLM like gpt-4o-mini [43] to analyze a user's interactions with musical instruments, it generates a vague feature such as "Component Functionality." However, this feature lacks the specific details that truly matter to a musician's decision-making process, like "Enhanced Durability," "Surface Look," and "Material Type." In contrast, when using a model with extended Chain-of-Thought (CoT) [67] reasoning, like o1-mini[45] or o3-mini [46], these more detailed and personalized features are uncovered, resulting in a richer and more precise feature set. This example highlights how deeper reasoning allows LLMs to better capture the nuances of user preferences, emphasizing the need for inference scaling to overcome limitations of quick thinking.

[1]In this paper, a *feature* refers to a plain-text string generated by an LLM. It can be encoded for use in traditional recommenders or used directly in LLM-as-Rec prompts.

These observations motivate us to ask: can *inference scaling*—an approach that allocates more computation to allow deeper reasoning [44, 55]—help overcome the limitations of quick inference in recommendation settings? Inspired by its success in domains such as math [64] and code generation [85], we explore whether deeper reasoning can also uncover richer, more interpretable user-level features for better recommendations. Unlike prior work that focuses on scaling model size or data volume [80, 81, 83], we examine how extended reasoning chains improve feature generation. Specifically, we investigate the following research questions:

RQ1: Can inference scaling improve recommendation performance, and if so, to what extent? We compare baseline LLMs (e.g., gpt-4o-mini) with models that use extended reasoning (e.g., o1-mini and o3-mini). Our experiments show that models with long-CoT reasoning outperform those without, achieving a 12% increase in NDCG@10. This demonstrates that extended reasoning, which has proven effective in math and coding tasks, can also enhance recommendation models. Interestingly, while algorithms like Beam Search [69] and Monte Carlo Tree Search (MCTS) [7] have achieved significant success in other domains, they do not perform as well as the Best-of-N [12] strategy for feature augmentation in recommendation tasks. These findings suggest that deeper inference, combined with the appropriate search strategy, can lead to substantial improvements in recommendation performance.

RQ2: Which factors drive these performance gains? We observe that performance improvements are largely driven by the *increased feature quantity* and *enhanced specificity achieved* through inference scaling. We use feature quantity as a proxy for reasoning depth: more unique valid features indicate a deeper exploration of user preferences, thereby addressing the issue of incomplete feature coverage. To assess this, we plot the number of unique valid features generated by different LLMs against their recommendation performance, as shown in Figure 1. Our results reveal a positive correlation: models like DeepSeek-R1 [23] and o3-mini, which generate the most features, also deliver the best performance. In addition, using an LLM-as-a-judge [88] approach, we find that inference scaling models consistently produce more detailed and precise features compared to non-scaled models, effectively mitigating the problem of insufficient specificity.

RQ3: Given that feature quantity influences performance, what factors affect the number of generated features? We investigate how model families, long-CoT reasoning, and model size impact the overall quantity of generated features. Additionally, we explore the effect of search strategies. Our results show that feature generation is not merely the sum of individual steps—pursuing local optimality at the step level does not always lead to overall optimality. Step-level methods like Beam Search and MCTS underperform compared to the solution-level Best-of-N approach in producing a larger number of useful features. These findings highlight the important interaction between models and search strategies in uncovering a more comprehensive set of features.

In summary, our contributions are as follows:

(1) To our knowledge, we are the first to explore inference scaling to address limitations in feature augmentation for recommendation systems, bridging insights from reasoning tasks to personalization.

(2) We show that inference scaling leads to better recommendation performance by increasing feature quantity and specificity.

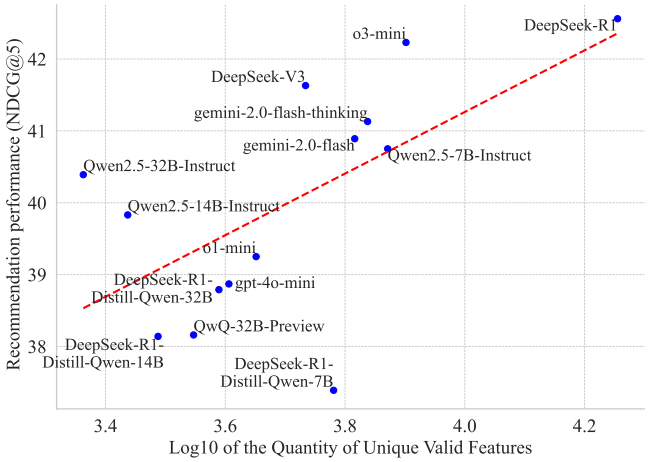


Figure 1: The positive correlation between recommendation performance and the number of unique valid features generated by different LLMs. The red dotted line represents the best-fit line of the data.

(3) We provide an in-depth analysis of how different models and search algorithms influence feature generation, highlighting their role in enhancing performance.

2 Probing Inference Scaling for Recommendation Systems

In this section, we provide an overview of the tasks, our inference scaling strategies, and the experimental setup.

2.1 Task Formulation

We adopt a typical sequential recommendation setting [19]. Let U be the set of users and I the set of items. Each item $i \in I$ has a unique ID and some side information (like title and description). The rating $r_{u,i} \in \{0, 1\}$ shows whether user u liked (1) or disliked (0) item i. Given a user $u \in U$ and their interaction history $\mathcal{H}_u = (i_1, i_2, \ldots, i_{|\mathcal{H}_u|})$, we use LLMs to infer the features that differentiate items the user likes from those the user does not. The features are represented as $\mathcal{F}_u = \{\mathbf{f}_u^1, \mathbf{f}_u^2, \ldots, \mathbf{f}_u^{|\mathcal{F}|}\}$, where each $\mathbf{f}_u^i$ is a text-based factor (e.g., "Age Range" or "Complexity Level" as shown in Figure 2). The final goal is to learn a sequential recommendation model $\Phi(\mathcal{H}_u, \mathcal{F}_u)$ that predicts the user's next target item $i^* \in I$ from a set of candidates.

2.2 Inference Scaling Procedure

Our evaluation framework is shown in Figure 2. Inspired by [79], we frame our inference scaling procedure from a reinforcement learning (RL) perspective with the following elements:

Policy Model is responsible for generating features that capture user preferences. Given a user's interaction history, including item titles, descriptions, and user ratings (all directly from the original dataset), the model is prompted to find differences between liked and disliked items and generate potential features for recommendations. We experiment with 14 different LLMs as policy models to evaluate how their outputs affect performance.

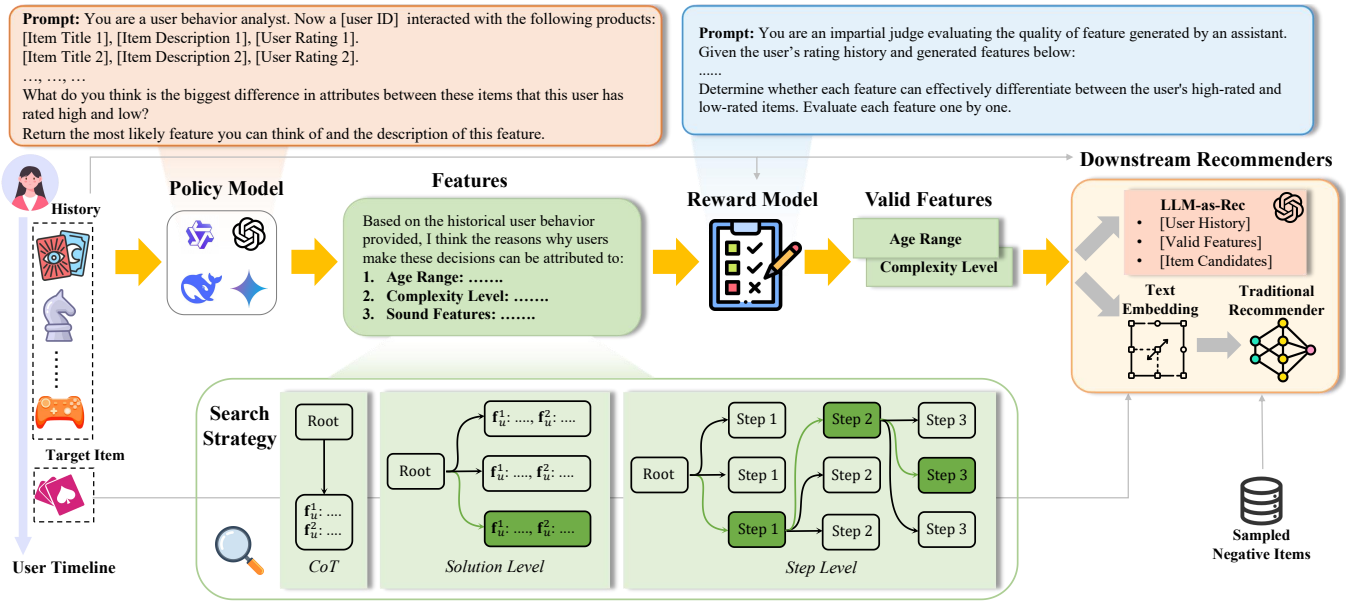


Figure 2: The whole evaluation framework of LLMs for feature augmentation in recommendation systems. The policy model generates user preference features using different search strategies. These features are then evaluated by a reward model to identify those that effectively differentiate between liked and disliked items. The valid features are used to enhance both LLM-based and traditional recommendation models. Among the search strategies, Best-of-N operates at the solution level, while Beam Search and MCTS operate at the step level.

These include: (1) **GPT** series: gpt-4o-mini [43], o1-mini [45], o3-mini [46]. (2) **Qwen** series: Qwen2.5-Instruct in 7B, 14B, and 32B sizes [72], plus QwQ-32B-Preview [61]. (3) **Gemini** series: gemini-2.0-flash [16] and gemini-2.0-flash-thinking-exp [16]. (4) **DeepSeek** series: DeepSeek-V3 [36] and DeepSeek-R1 [23], including its distilled Qwen-based models (7B, 14B, 32B)[2].

Reward Model evaluates the features generated by the policy model, designating them as *valid* if they successfully distinguish between a user's liked and disliked items. Only those valid features are used in downstream recommendation tasks. Specifically, we use Qwen2.5-7B-Instruct [72] as the reward model.

Search Strategy guides how the policy model explores and selects features. Various strategies can be employed, such as Best-of-N [12], Beam Search [69], or Monte Carlo Tree Search (MCTS) [7]. These methods help balance the exploration of new features with the reuse of promising ones. They are consistent for all policy models. Details of each strategy can be found in Appendix A.1.

2.3 Downstream Recommendation Tasks

To evaluate how useful the generated features are, we conduct comprehensive benchmarking across different types of sequential recommendations, including LLM-as-Rec and traditional recommendation models.

2.3.1 LLM-as-Rec. We use valid features, user history, and candidate items as input to LLMs. Specially, we adopt gpt-4o-mini to perform the following recommendation tasks: (1) **Direct Recommendation (DR)** [38]: A listwise ranking method where the model ranks C candidate items based on valid features and the user's history. The prompt is in Appendix Figure 6. Each item in the history contains a title, a textual description, and the user's rating, all taken directly from the original dataset. (2) **In-Context Learning (ICL)** [14]: Similar to DR, but uses a one-shot prompt example to help the model rank candidates. (3) **Next Item Prediction (NIP)** [21]: The model predicts the next item a user is likely to interact with, based on their history and a set of C candidate items.

2.3.2 Traditional Recommenders. We also incorporate generated features into traditional recommendation models using the LHUC framework [58]. Specifically, for each user u, we join their valid features $\mathcal{F}_u$ into a string, encode it with bge-m3 [9] into text embedding E_u, and map it through two MLP layers to match the hidden size of the recommendation model. Denote the recommendation model as R, the user's sequential representation is:

$$R(\mathcal{H}_u) \circ sigmoid(MLP(E_u)),$$

where $\circ$ denotes the element-wise multiplication. In this way, we can incorporate the user's features into the sequential embedding.

We evaluate the following state-of-the-art sequential recommendation models: (1) **SASRec** [31]: A representative baseline for sequential recommendation, which utilizes a Transformer-based [63] architecture to model users' sequential interactions. (2) **LRURec** [77]: A recently proposed model using linear recurrent units [22, 47] for faster inference.

[2]We refer to DeepSeek-R1-Distill-Qwen-7B simply as DeepSeek-R1-7B, and similarly for 14B and 32B.

Table 1: Statistics of the two datasets.

Dataset	#Users	#Items	#Interactions
Toys	3,962	11,119	65,099
Instruments	1,411	6,317	23,804

2.4 Experimental Setup

2.4.1 Datasets. We conducted experiments on two public datasets: **Toys** and **Instruments**, which correspond to the "Toys and Games" and "Musical Instruments" categories in the Amazon dataset [42]. We selected these datasets because their moderate size makes our experimental pipeline feasible when testing across 14 different LLMs, keeping both time and financial costs manageable. The statistics for each dataset are provided in Table 1.

2.4.2 Metrics. For DR, ICL, and traditional recommendation models, we report **HIT@K** and **NDCG@K** for $K \in \{5, 10\}$ as the evaluation metrics, which are widely used in sequential recommendation [59, 60, 74]. NDCG places higher weight on items ranked at the top, while HIT treats all positions equally and essentially measures recall. For the NIP setting, we use **HIT** to measure prediction accuracy. In the LLM-as-Rec setting, to ensure that LLM does not produce invalid outputs (such as missing or duplicate items), we also report the **Valid Rate**, which measures the proportion of compliant rankings.

2.4.3 Details. We adopted the *leave-one-out* strategy, commonly used in previous work [3, 13, 89]. For each user u, the final interacted item $i_{|\mathcal{H}_u|}$ is set aside as the test item, while all previously interacted items constitute the training set [24]. For training traditional recommendation models, each positive interaction is paired with a randomly selected negative item, and we optimize the models using BPR Loss [50]. Both SASRec and LRURec use 2 layers, a dropout rate of 0.3, and a learning rate of 0.001. The batch sizes are 128 for the Toys dataset and 32 for Instruments. During inference, we randomly sample $C - 1$ negative items for each user and rank them together with the ground-truth item [31]. Following [34, 71], We set $C = 20$ for LLM-as-Rec and $C = 1000$ for traditional recommendation models. For Best-of-N and Beam Search, we use $N = 4$ beams and $M = 1$ new branch per beam due to the computational time limit. The maximum output length is set to 8192 tokens. For Beam Search and MCTS, the tree depth (i.e., maximum step count) is set to 50. For more details on the model implementation, please refer to the code link.

3 Impact of Inference Scaling on Recommendation Performance (RQ1)

In this section, we first investigate whether inference scaling can enhance the performance of recommendation systems.

3.1 Effect of Different Policy Model Features

We examine how the choice of policy model influences recommendation quality when features are generated with a basic CoT strategy. The results for LLM-as-Rec and traditional recommendation models are shown in Table 2 and 3, respectively. Because each setting uses a different number of candidate items, the two tables should not be compared side by side. Key findings are summarized below:

(1) Positive Impact of Features: Overall, adding distinguishable features that capture user preferences improves the recommendation performance of the LLM. For instance, DeepSeek-R1 achieves an NDCG@10 of 48.86 in DR, which is a 12% improvement over the baseline (43.50). Furthermore, the Valid Rate improves after feature augmentation, indicating that the features validated by the reward model are helping the LLM complete the recommendation task more effectively.

(2) Long-CoT vs. Non-Long-CoT Models: For LLM-as-Rec, we observe that models with long-CoT processes, such as o1-mini and o3-mini, outperform their non-long-CoT counterparts (gpt-4o-mini). For instance, for DR on the Toys dataset, the NDCG@5 metric improves from 38.87 (for gpt-4o-mini) to 39.25 (for o1-mini) and 42.23 (for o3-mini). Similarly, gemini-2.0-flash-thinking-exp shows an improvement over the non-long-CoT gemini-2.0-flash, confirming that long-CoT reasoning provides a significant advantage in feature generation. While DeepSeek-V3 performs slightly worse than DeepSeek-R1 on the Instruments dataset, the latter outperforms DeepSeek-V3 on the larger Toys dataset. One notable exception is QwQ, which underperforms relative to the Qwen2.5-Instruct series. We speculate that this is due to QwQ generating fewer features, a point we will further explore in Section 5. For traditional recommendation models, distilled versions of DeepSeek-R1 and gemini-2.0-flash-thinking-exp also lead to strong performance, suggesting the potential of integrating long-CoT reasoning into traditional frameworks.

(3) Transfer of Advanced Reasoning to Recommendation: Models like DeepSeek-R1, gemini-2.0-flash-thinking-exp, and o3-mini stand out for their superior performance, with the first two excelling on different datasets for LLM-as-Rec. These models also perform strongly in math and coding topics of Chatbot Arena [11], indicating that **the advanced inference scaling techniques driving their success in reasoning-intensive tasks also benefit recommendation**. We believe this transfer stems from the out-of-distribution (OOD) nature of recommendation data: due to highly personalized user behavior, recommendation data is generally not used during the pre-training of most LLMs [17, 36]. As suggested by [75], training LLMs on long-CoT data can improve generalization to OOD tasks, enabling them to reason more effectively about nuanced user preferences.

In summary, we demonstrate that augmenting recommendation systems with LLM-generated features significantly improves performance, especially when models use long-CoT reasoning. Moreover, policy models that excel in math and coding tasks tend to generate superior features for capturing user preferences. This convergence underscores the potential of inference scaling as a general strategy for enhancing recommendation systems.

3.2 Effect of Different Search Strategies

In math and coding tasks, models often cannot arrive at a final solution in a single step. Instead, they often rely on search algorithms—either exploring multiple solution paths in parallel [5] or iteratively refining a candidate solution [40]. Naturally, one might

Table 2: Performance of LLM-as-Rec with features from various policy models. The baseline performance, labeled as "w/o features", reflects results without the inclusion of additional features. Bold denotes the highest scores across all policy models. For brevity, '%' is omitted from scores in subsequent tables and figures.

Dataset	Policy Model	DR					ICL					NIP
		Valid Rate	NDCG@5	HIT@5	NDCG@10	HIT@10	Valid Rate	NDCG@5	HIT@5	NDCG@10	HIT@10	HIT
Toys	w/o features	94.80	37.21	51.01	43.50	70.67	99.33	35.46	49.69	42.06	70.28	24.46
	Qwen2.5-7B-Instruct	97.28	40.75	56.72	46.84	75.54	99.67	39.26	54.88	45.73	74.98	27.13
	Qwen2.5-14B-Instruct	97.38	39.83	55.24	46.20	74.98	99.52	38.22	53.47	44.90	74.22	26.23
	Qwen2.5-32B-Instruct	96.95	40.39	54.94	46.95	75.41	99.71	39.46	55.14	45.91	75.23	26.75
	DeepSeek-R1-7B	96.90	37.39	52.36	44.15	73.33	99.52	36.24	51.37	43.09	72.69	24.27
	DeepSeek-R1-14B	95.90	38.14	52.56	44.71	73.05	99.57	35.81	49.86	43.12	72.65	25.08
	DeepSeek-R1-32B	97.38	38.79	53.97	45.58	75.02	100.00	37.11	52.84	43.85	73.63	25.56
	QwQ-32B-Preview	96.09	38.16	53.10	44.37	72.41	99.71	35.32	50.65	42.22	72.12	24.75
	gpt-4o-mini	97.14	38.87	52.97	45.64	74.13	99.71	38.16	52.89	44.78	73.46	26.51
	o1-mini	97.19	39.25	53.29	46.13	74.68	99.81	38.82	53.94	45.44	74.58	25.99
	o3-mini	97.23	42.23	**56.89**	48.11	75.28	99.86	40.69	55.97	47.25	**76.41**	29.04
	DeepSeek-V3	97.28	41.63	56.47	48.15	**76.72**	99.90	40.98	55.80	47.06	74.65	28.42
	DeepSeek-R1	97.23	**42.56**	56.74	**48.86**	76.31	99.62	**42.35**	**58.83**	**47.66**	75.30	**29.33**
	gemini-2.0-flash	97.38	40.89	55.63	47.12	75.07	99.76	40.76	56.64	46.82	75.48	27.61
	gemini-2.0-flash-thinking-exp	97.42	41.13	55.07	47.35	74.40	99.76	41.24	57.31	47.20	75.72	28.85
Instruments	w/o features	95.33	40.77	56.13	46.85	75.23	99.75	36.01	53.14	43.23	75.59	23.99
	Qwen2.5-7B-Instruct	97.42	44.10	58.33	50.24	77.53	99.63	39.73	54.44	46.66	75.80	27.68
	Qwen2.5-14B-Instruct	98.28	39.28	53.44	46.67	76.47	99.63	35.87	52.10	44.07	77.53	27.68
	Qwen2.5-32B-Instruct	97.79	43.28	58.11	49.92	78.62	99.88	39.95	57.27	47.13	79.56	27.68
	DeepSeek-R1-7B	96.56	41.88	57.96	48.37	77.96	99.88	36.80	53.94	44.45	77.83	25.22
	DeepSeek-R1-14B	96.43	40.31	55.23	47.75	78.44	99.63	35.55	52.72	43.01	76.05	24.48
	DeepSeek-R1-32B	97.66	40.13	55.04	46.99	76.20	99.63	38.66	55.93	45.23	76.30	25.46
	QwQ-32B-Preview	95.69	43.20	59.25	49.50	78.66	99.63	39.22	55.80	46.64	78.77	25.22
	gpt-4o-mini	97.79	41.86	57.48	48.76	78.99	100.00	40.79	58.43	47.34	78.84	29.77
	o1-mini	98.15	46.21	62.66	51.65	79.57	99.88	43.04	61.08	48.81	79.06	30.01
	o3-mini	97.54	45.60	59.90	52.12	80.08	99.88	43.20	59.85	49.31	78.82	30.01
	DeepSeek-V3	97.91	48.10	62.69	53.84	80.65	99.63	44.59	62.10	51.04	82.10	**32.10**
	DeepSeek-R1	98.52	47.82	62.05	53.36	79.28	99.63	44.29	61.73	50.60	81.48	30.75
	gemini-2.0-flash	97.54	48.98	64.56	55.05	**83.23**	99.88	45.65	64.53	51.54	82.76	31.49
	gemini-2.0-flash-thinking-exp	98.77	**49.96**	**65.63**	**55.50**	82.57	100.00	**47.41**	**64.82**	**53.54**	**83.76**	31.24

ask whether these same algorithms are equally beneficial for feature generation in a recommendation context, where the model's goal is to uncover user preferences. In this study, we evaluate four search strategies for generating features: CoT, Best-of-N, Beam Search, and MCTS.

To balance effectiveness and computational efficiency, we evaluate these strategies on the smaller Instruments dataset, using Qwen2.5-7B-Instruct and Qwen2.5-14B-Instruct as our policy models. The reward model evaluates each strategy by counting the number of *valid* features it produces—features that clearly distinguish between items a user likes and dislikes. The final output is selected based on the highest number of valid features.

As shown in Table 4, all strategies outperform the baseline CoT approach. For example, with Qwen2.5-14B-Instruct as the policy model, CoT alone achieves an NDCG@5 of 39.28. In comparison, advanced search algorithms achieve significantly better results: Best-of-N (43.11), Beam Search (41.09), and MCTS (40.64). When Qwen2.5-7B-Instruct is used in the ICL task, Beam Search slightly outperforms Best-of-N. However, as the policy model size increases (from 7B to 14B), Best-of-N maintains its advantage over step-level methods like Beam Search and MCTS. Overall, Best-of-N shows the largest performance gains especially when the policy model is sufficiently large, likely due to its ability to select the best features from multiple generated outputs.

These findings suggest that while step-level strategies like Beam Search and MCTS are effective in domains where precise incremental steps are crucial (such as solving mathematical proofs [70] or playing games like Go [54]), Best-of-N is more effective in identifying the most relevant features for recommendation tasks. We will further explore the reasons behind this trend in Section 5.2.

4 Advantages of Inference Scaling for Feature Augmentation(RQ2)

In the previous section, we showed that features generated through inference scaling improve recommendation performance. However, it's still unclear *why* this improvement happens—specifically, what advantages long-CoT LLMs offer over those non-long-CoT models.

4.1 Increased Number of Unique Features

To investigate this aspect, we compare the features generated by gpt-4o-mini and o1-mini in subsection 3.1. For each model, we collect all valid features generated across users, then measure both the total and the number of *unique* features after removing duplicates through clustering. In particular, we use the bge-m3 [9] to embed features and DBSCAN [18] to cluster them. Since users can exhibit diverse decision factors [25], deeper and more personalized reasoning allows the LLM to uncover the unique preferences and decision-making factors of each user. As the model performs more in-depth analysis of these varied factors, it is likely to generate a greater number of distinct features, each reflecting different aspects of the user's preferences. Therefore, the number of unique features can serve as a valuable metric for assessing the extent to which the model has captured the complexity and individuality of

Table 3: Performance of Traditional Recommenders with features from various policy models.

Dataset	Policy Model	SASRec				LRURec			
		NDCG@5	HIT@5	NDCG@10	HIT@10	NDCG@5	HIT@5	NDCG@10	HIT@10
Toys	w/o features	4.74	7.28	6.38	12.41	4.72	7.48	6.47	12.97
	Qwen2.5-7B-Instruct	5.54	8.50	7.25	13.83	5.38	8.34	7.34	14.41
	Qwen2.5-14B-Instruct	5.53	8.42	7.19	13.60	5.47	8.55	7.44	14.66
	Qwen2.5-32B-Instruct	5.55	8.60	7.19	13.70	5.49	8.55	7.41	14.54
	DeepSeek-R1-7B	5.40	8.34	7.11	13.68	5.48	8.52	7.42	14.54
	DeepSeek-R1-14B	5.46	8.29	7.17	13.63	5.47	8.47	7.39	14.46
	DeepSeek-R1-32B	**5.59**	8.47	**7.34**	13.91	**5.57**	**8.57**	7.46	14.46
	QwQ-32B-Preview	5.53	8.42	7.21	13.63	5.39	8.50	7.41	**14.77**
	gpt-4o-mini	5.43	8.47	7.11	13.68	5.54	8.42	**7.48**	14.44
	o1-mini	5.47	8.47	7.20	13.91	5.49	**8.57**	7.33	14.29
	o3-mini	5.43	8.27	7.25	**13.93**	5.44	8.42	7.36	14.41
	DeepSeek-V3	5.35	8.17	7.11	13.63	5.53	8.52	7.45	14.49
	DeepSeek-R1	5.46	8.27	7.28	**13.93**	5.49	**8.57**	7.35	14.36
	gemini-2.0-flash	5.56	**8.65**	7.13	13.55	5.42	8.39	7.39	14.49
	gemini-2.0-flash-thinking-exp	5.57	8.55	7.27	13.88	5.43	8.45	7.38	14.51
Instruments	w/o features	5.31	6.98	5.91	8.83	4.53	5.77	5.39	8.47
	Qwen2.5-7B-Instruct	5.79	7.69	6.81	10.96	5.96	7.33	6.79	9.89
	Qwen2.5-14B-Instruct	5.66	7.40	6.70	10.60	5.92	6.98	**6.99**	**10.39**
	Qwen2.5-32B-Instruct	5.72	7.62	6.72	10.75	6.03	7.40	6.80	9.82
	DeepSeek-R1-7B	**6.25**	**8.33**	**7.20**	**11.32**	5.82	7.19	6.68	9.89
	DeepSeek-R1-14B	5.96	8.19	6.88	11.03	5.94	7.19	6.82	9.89
	DeepSeek-R1-32B	5.87	7.90	6.68	10.39	6.00	7.40	6.85	10.04
	QwQ-32B-Preview	5.79	7.62	6.71	10.46	5.95	7.19	6.90	10.11
	gpt-4o-mini	5.82	7.83	6.85	11.03	6.10	**7.47**	6.86	9.82
	o1-mini	5.72	7.47	6.64	10.32	5.82	7.12	6.70	9.82
	o3-mini	5.76	7.76	6.76	10.89	5.81	7.26	6.59	9.68
	DeepSeek-V3	6.08	8.19	6.78	10.39	5.91	7.12	6.82	9.96
	DeepSeek-R1	5.73	7.76	6.45	9.96	5.77	7.33	6.52	9.75
	gemini-2.0-flash	5.86	7.83	6.78	10.75	6.04	7.33	6.93	10.11
	gemini-2.0-flash-thinking-exp	5.85	7.97	6.85	11.10	**6.12**	**7.47**	6.93	10.04

Table 4: Performance of LLM-as-Rec with features from various search strategies.

Policy Model	Search Strategy	DR					ICL					NIP
		Valid Rate	NDCG@5	HIT@5	NDCG@10	HIT@10	Valid Rate	NDCG@5	HIT@5	NDCG@10	HIT@10	HIT
w/o features		95.33	40.77	56.13	46.85	75.23	99.75	36.01	53.14	43.23	75.59	23.99
Qwen2.5-7B -Instruct	CoT	97.42	44.10	58.33	50.24	77.53	99.63	39.73	54.44	46.66	75.80	27.68
	Best-of-N	97.54	**44.91**	**60.66**	**51.22**	**80.20**	99.75	39.70	56.23	46.64	77.68	**28.04**
	Beam Search	97.42	43.70	59.09	49.85	78.41	99.63	**40.06**	**56.67**	**46.92**	**77.90**	26.20
	MCTS	97.42	41.87	56.44	48.12	75.88	99.51	39.43	56.37	45.81	76.27	26.57
Qwen2.5-14B -Instruct	CoT	98.28	39.28	53.44	46.67	76.47	99.63	35.87	52.10	44.07	**77.53**	**27.68**
	Best-of-N	97.91	**43.11**	**58.54**	**49.22**	77.76	99.88	**40.06**	**56.77**	**46.73**	77.34	**27.68**
	Beam Search	97.79	41.09	55.97	47.79	76.73	99.75	36.62	52.90	43.93	75.59	27.31
	MCTS	97.79	40.64	54.72	48.17	**77.99**	99.51	37.99	54.39	44.95	76.14	24.72

Table 5: Total and unique feature counts generated by two policy models on different datasets.

Dataset	Policy Model	#Total features	#Unique features
Toys	gpt-4o-mini	25,536	4,037
	o1-mini	31,822	4,482
Instruments	gpt-4o-mini	8,156	2,766
	o1-mini	11,443	4,086

user decision-making. The results, presented in Table 5, show that o1-mini generates more features overall and more unique features than gpt-4o-mini, indicating that extended CoT reasoning enables a more comprehensive exploration of user preferences.

To further analyze the relationship between feature quantity and recommendation performance, we use the larger Toys dataset to minimize noise and yield robust insights. For each policy model, we plot the number of unique valid features (log-scaled, base 10) against DR performance. Results in Figure 1 indicate a positive correlation: as the number of unique valid features increases, recommendation performance improves. This finding suggests that the quantity of unique valid features can be considered a reasonable indicator of the

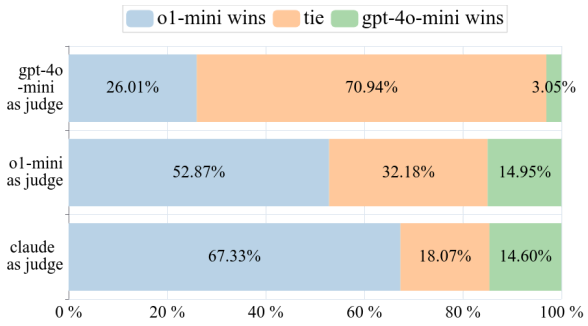


Figure 3: Win-Tie-Lose Comparisons on specificity of features from gpt-4o-mini and o1-mini.

depth of personalized reasoning, and can serve as a useful predictor of recommendation performance.

4.2 More Detailed and Specific Descriptions

Beyond generating more features, we find that o1-mini also produces features that are more specific and clearly described compared to gpt-4o-mini. To evaluate this, we use an LLM-as-a-judge [88] approach. We present pairs of feature descriptions—one from gpt-4o-mini and one from o1-mini—to three judge models: gpt-4o-mini, o1-mini, and claude-3-5-sonnet [2]. To mitigate position bias [53], each feature pair is evaluated twice, with the positions of the generated features swapped between the two rounds. We consider a feature description superior only when both evaluations are consistent; otherwise, the result is recorded as a tie.

Figure 3 shows that all judges unanimously rate o1-mini's features as more specific and better described than gpt-4o-mini. Notably, even though LLMs often exhibit self-enhancement bias [6, 15, 88]—where they tend to favor their own generated content—gpt-4o-mini still rated o1-mini's descriptions more favorably. This further supports that o1-mini produces more detailed and precise features.

In summary, the key benefits of inference scaling in feature generation are: (1) it generates more features, and (2) these generated features are more specific and detailed. These advantages suggest that longer reasoning chains enable models to explore a richer space of possible features, providing more meaningful insights into user preferences for recommendation tasks.

5 Factors Influencing the Number of Generated Features (RQ3)

In previous sections, we showed that feature quantity plays an important role in recommendation performance. Here, we examine what factors influence this quantity—focusing on the impact of model choice and search strategy.

5.1 Model Selection

Following the procedure in Section 4, we collect all features generated by each policy model across all users. The reward model then identifies which features effectively distinguish users' liked items from disliked ones. After clustering and removing duplicates, we compare the number of unique features from each policy model.

Figure 4 illustrates the growth of unique features as a function of the total number of valid features on the Toys dataset. The horizontal axis represents the total number of valid, distinguishable features generated across all users, while the vertical axis indicates the number of unique features—serving as a proxy for the depth of personalization. Three key observations emerge:

(1) Model Families. The DeepSeek and Gemini families attain the top points on the vertical and horizontal axes, respectively. DeepSeek-R1 generates the highest number of unique valid features, whereas gemini-2.0-flash-thinking-exp produces a large number of features overall but fewer truly unique ones—indicating high quantity but somewhat weaker personalization. Within the GPT family, o3-mini stands out for strong uniqueness, second only to DeepSeek-R1. By contrast, the Qwen series produces fewer features in total (under 20K) and fewer unique features (under 5K).

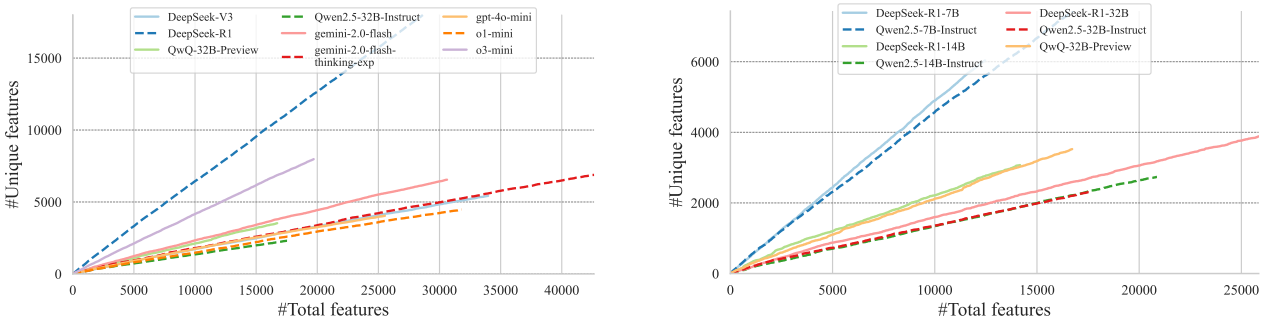
(2) Long-CoT. When comparing models within the same family, those incorporating extended reasoning—via SFT [28], RL [41], or other advanced training techniques [23]—produce significantly more unique features than their base counterparts. For example, in Figure 4(a), DeepSeek-R1 produces over three times the number of unique features compared to DeepSeek-V3, while o3-mini outperforms gpt-4o-mini by over 50%. Similarly, gemini-2.0-flash-thinking-exp surpasses gemini-2.0-flash, and the distilled versions of DeepSeek-R1 for Qwen exhibit steeper slopes than standard Qwen models of the same scale (Figure 4(b)). This suggests that explicitly producing a richer chain-of-thought (rather than keeping it implicit in model parameters) helps capture more granular decision factors, facilitating the discovery of a broader range of user-specific features. One exception is QwQ, which generates fewer features than Qwen2.5-7B-Instruct, aligning with the results in Section 3.1 that QwQ underperforms Qwen in downstream recommendations.

(3) Model Size. Overall, larger models tend to generate more unique features. For example, DeepSeek-R1 (671B) generates over 17,500 unique features, whereas its distilled 7B, 14B, and 32B versions produce only a few thousand. However, size is not always a guarantee of better personalization. Within the 7B–32B range, the 7B variants often surpass their 14B and 32B counterparts in personalization capacity across both Qwen and distilled DeepSeek models. Notably, smaller models that benefit from long-CoT training can even outperform larger models without extended reasoning. For instance, DeepSeek-V3 (671B) generates around 5K unique features, whereas DeepSeek-R1-7B produces over 7K—highlighting the substantial role of long chain-of-thought processes.

In summary, long-CoT training and model family are influential factors in feature generation. In highly personalized domains such as recommendation, where user decision factors vary widely, a detailed, case-by-case approach that leverages explicit and extended reasoning proves particularly advantageous.

5.2 Search Strategies

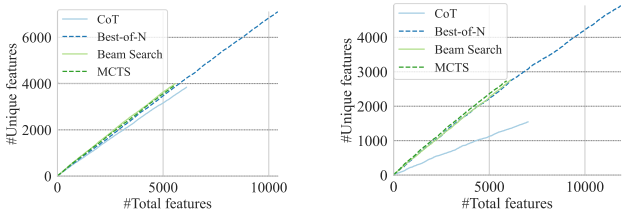
We next investigate how search strategies affect the quantity of generated features. Figure 5 presents results on the Instruments dataset. All strategies exhibit a roughly linear relationship between the total and unique feature quantity. Notably, Best-of-N consistently produces the highest total and unique feature counts, while Beam



(a) Comparison across model families.

(b) Model-size comparison across DeepSeek and Qwen.

Figure 4: Number of unique features generated by each policy model versus the total number of features on the Toys dataset.



(a) Using Qwen2.5-7B-Instruct as the policy model

(b) Using Qwen2.5-14B-Instruct as the policy model

Figure 5: Comparison of different search strategies on the Instruments dataset.

Search and MCTS generate fewer. This aligns with their recommendation performance in Section 3.2, where Best-of-N outperforms the other search strategies.

The underlying reason lies in the difference between step-level and solution-level optimization. In tasks like math [70] or games like Go [54], each intermediate step must be correct to avoid disrupting the final outcome; local missteps can ruin the entire sequence, making step-level search essential for achieving a globally optimal result. In contrast, user-feature generation in recommendation is less tightly coupled: each newly proposed feature—whether relevant or not—does not necessarily affect subsequent features. Even if some features are ineffective, the model can still generate valid and valuable features afterward. For instance, generating features such as appearance, price, or durability are relatively independent, so discovering useful features after ineffective ones can still improve the overall result. Therefore, step-level search provides limited benefits in recommendation tasks, whereas solution-level strategies like Best-of-N are more effective in identifying the most promising solution without requiring each incremental step to be optimal.

6 Related Work

LLMs for Rec. LLMs can enhance recommendation systems by providing contextual understanding and reasoning abilities [27]. Recent work on LLM-based augmentation in recommendation generally follows two main directions. The first approach leverages LLMs for text embedding, integrating user and item attributes into unified representations [26, 33, 52, 76, 84]. The second approach focuses on generating additional information for recommendations. For instance, KAR [68] instructs LLMs to produce item descriptions and user preference rationales, while LLM-ESR [39] summarizes user preferences. RLMRec [49] describes both user and item attributes, incorporating collaborative information, and LLM-CF [56] uses a chain-of-thought to represent user decisions. RecSAVER [62] generates post hoc rationales which could be seen as interaction-level features and distills them into a smaller model for rating prediction. ReasoningRec [4] and EXP3RT [32] additionally generate user and item profiles, and reason whether the user will like/dislike the item before making recommendations. Reason4Rec [20] breaks the reasoning process into multiple steps, training each step independently to improve performance. However, most of these methods rely on relatively fast inference, frequently resulting in incomplete coverage and insufficient specificity [29].

Inference Computation Scaling. Inference scaling involves allocating more computational resources to the inference stage, allowing LLMs to reason based on additional prior steps. This shift from System-1 (fast thinking) to System-2 (slow thinking) reasoning enhances their ability to process complex tasks [10, 29]. The relationship between performance and inference time was first examined by Snell et al. [55], and recent studies [8, 48, 51, 73, 82] have demonstrated promising results in math and coding tasks. Moreover, Yue et al. [78] investigates inference scaling for Retrieval Augmented Generation, and Sun et al. [57] employs test-time scaling to address challenges in complex multi-step reasoning. However, none of these works have investigated whether inference scaling can help address the incomplete coverage problem in recommendation tasks, and we aim to explore how this technique can improve personalized recommendation systems through feature augmentation.

7 Conclusion

In this paper, we demonstrate that inference scaling can significantly enhance feature augmentation for recommendation systems. Our experiments show that long-CoT models, compared to traditional fast inference methods, generate more detailed and specific features that better capture user decision factors, resulting in markedly improved recommendation performance. In contrast

Direct Recommendation Prompt for LLMs

A user has interacted with the following items:
[Item Title 1], [Item Description 1], [User Rating 1].
[Item Title 2], [Item Description 2], [User Rating 2].
…, …, …
Here is the summary of the preferences of the user:
{Features $\mathcal{F}_u$ generated by LLMs}
Based on the history, please rank the following 20 candidates in order of priority from highest to lowest:
[Candidate Title 1], [Candidate Description 1].
[Candidate Title 2], [Candidate Description 2].
…, …
Output format: a python list.

Figure 6: The prompt of LLM direct recommendation. Bold text represents portions removed when features from LLMs are not included.

to faster inference methods, long-CoT reasoning overcomes challenges like incomplete feature coverage and insufficient specificity. Additionally, we observe that the quantity and specificity of features are closely linked to the accuracy and effectiveness of the recommendations. Overall, our findings highlight the potential of inference scaling to improve user preference modeling and personalization in recommendation systems, paving the way for future research on its broader application across personalized tasks.

A Appendix

A.1 Search Strategies

In this section, we introduce the search algorithms in detail:

(a) CoT. LLM explicitly outputs its intermediate reasoning steps. This has been the default approach in our experiments.

(b) Best-of-N [12]. The policy model generates N complete outputs for each user. Each output includes a set of proposed features, which are evaluated by the reward model. A feature is marked as *valid* if it effectively distinguishes between items the user liked and disliked. The final output is chosen as the one with the highest number of valid features.

(c) Beam Search [69]. The policy model first produces N partial outputs (beams), each ending at the first occurrence of `\n\n`—which we treat as a step. The reward model evaluates each beam based on the number of valid features and keeps the top N/M beams. Each of these retained beams is then expanded by generating M new branches, maintaining a total of N beams at each stage. This process repeats until complete outputs are generated.

(d) MCTS [7]. This lookahead method builds a search tree from the initial prompt (root) to a complete feature set (leaf). In each round, it performs four phases: **(1) Selection**: Traverse the tree from the root to an unexplored node with the highest UCB score. Each node represents a partial output that ends with `\n\n`.**(2) Expansion**: Expand the selected node by generating the next step in the output—i.e., continue the output until the next `\n\n` is reached. **(3) Evaluation (Rollout)**: From the expanded node, simulate a full output and use the reward model to assess its quality. **(4) Backpropagation**: Propagate the estimated reward back through the tree to update earlier nodes and inform future selections.

CoT and Best-of-N operate at the **solution level**, selecting among multiple fully generated outputs. In contrast, beam search and MCTS operate at the **step level**, intervening on the policy model's partial outputs as they unfold [79].

References

[1] Arkadeep Acharya, Brijraj Singh, and Naoyuki Onoe. 2023. Llm based generation of item-description for recommendation system. In *Proceedings of the 17th ACM Conference on Recommender Systems*. 1204–1207.

[2] Anthropic. 2024. Claude 3.5 sonnet. https://www.anthropic.com/news/claude-3-5-sonnet

[3] Immanuel Bayer, Xiangnan He, Bhargav Kanagal, and Steffen Rendle. 2017. A generic coordinate descent framework for learning from implicit feedback. In *Proceedings of the 26th international conference on world wide web*. 1341–1350.

[4] Millennium Bismay, Xiangjue Dong, and James Caverlee. 2024. ReasoningRec: Bridging Personalized Recommendations and Human-Interpretable Explanations through LLM Reasoning. *arXiv preprint arXiv:2410.23180* (2024).

[5] Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787* (2024).

[6] Jonathon D Brown. 1986. Evaluations of self and others: Self-enhancement biases in social judgments. *Social cognition* 4, 4 (1986), 353–376.

[7] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games* 4, 1 (2012), 1–43.

[8] Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. 2024. AlphaMath Almost Zero: process Supervision without process. *arXiv preprint arXiv:2405.03553* (2024).

[9] Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. M3-Embedding: Multi-Linguality, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation. In *Findings of the Association for Computational Linguistics ACL 2024*. 2318–2335.

[10] Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wangxiang Che. 2025. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567* (2025).

[11] Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E Gonzalez, et al. 2024. Chatbot arena: An open platform for evaluating llms by human preference. *arXiv preprint arXiv:2403.04132* (2024).

[12] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168* (2021).

[13] Sunhao Dai, Changle Qu, Sirui Chen, Xiao Zhang, and Jun Xu. 2024. Recode: Modeling repeat consumption with neural ode. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2599–2603.

[14] Sunhao Dai, Ninglu Shao, Haiyuan Zhao, Weijie Yu, Zihua Si, Chen Xu, Zhongxiang Sun, Xiao Zhang, and Jun Xu. 2023. Uncovering chatgpt's capabilities in recommender systems. In *Proceedings of the 17th ACM Conference on Recommender Systems*. 1126–1132.

[15] Sunhao Dai, Chen Xu, Shicheng Xu, Liang Pang, Zhenhua Dong, and Jun Xu. 2024. Bias and unfairness in information retrieval systems: New challenges in the llm era. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 6437–6447.

[16] Google Deepmind. 2024. Introducing Gemini 2.0: our new AI model for the agentic era. https://blog.google/technology/google-deepmind/google-gemini-ai-update-december-2024/

[17] 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* (2024).

[18] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. 226–231.

[19] Hui Fang, Danning Zhang, Yiheng Shu, and Guibing Guo. 2020. Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations. *ACM Transactions on Information Systems (TOIS)* (2020), 1–42.

[20] Yi Fang, Wenjie Wang, Yang Zhang, Fengbin Zhu, Qifan Wang, Fuli Feng, and Xiangnan He. 2025. Large Language Models for Recommendation with Deliberative User Preference Alignment. *arXiv preprint arXiv:2502.02061* (2025).

[21] 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. 299–315.
[22] Albert Gu, Karan Goel, and Christopher Ré. 2021. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396* (2021).
[23] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948* (2025).
[24] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*. 173–182.
[25] Xinyue He, Qi Liu, and Sunho Jung. 2024. The impact of recommendation system on user satisfaction: A moderated mediation approach. *Journal of Theoretical and Applied Electronic Commerce Research* 19, 1 (2024), 448–466.
[26] Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. 2024. Bridging language and items for retrieval and recommendation. *arXiv preprint arXiv:2403.03952* (2024).
[27] Jie Huang and Kevin Chen-Chuan Chang. 2022. Towards reasoning in large language models: A survey. *arXiv preprint arXiv:2212.10403* (2022).
[28] Zhen Huang, Haoyang Zou, Xuefeng Li, Yixiu Liu, Yuxiang Zheng, Ethan Chern, Shijie Xia, Yiwei Qin, Weizhe Yuan, and Pengfei Liu. 2024. O1 Replication Journey–Part 2: Surpassing O1-preview through Simple Distillation, Big Progress or Bitter Lesson? *arXiv preprint arXiv:2411.16489* (2024).
[29] Yixin Ji, Juntao Li, Hai Ye, Kaixin Wu, Jia Xu, Linjian Mo, and Min Zhang. 2025. Test-time Computing: from System-1 Thinking to System-2 Thinking. *arXiv preprint arXiv:2501.02497* (2025).
[30] Daniel Kahneman. 2011. Thinking, fast and slow. *Farrar, Straus and Giroux* (2011).
[31] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*. 197–206.
[32] Jieyong Kim, Hyunseo Kim, Hyunjin Cho, SeongKu Kang, Buru Chang, Jinyoung Yeo, and Dongha Lee. 2024. Review-driven Personalized Preference Reasoning with Large Language Models for Recommendation. *arXiv preprint arXiv:2408.06276* (2024).
[33] Dong-Ho Lee, Adam Kraft, Long Jin, Nikhil Mehta, Taibai Xu, Lichan Hong, Ed H Chi, and Xinyang Yi. 2024. STAR: A Simple Training-free Approach for Recommendations using Large Language Models. *arXiv preprint arXiv:2410.16458* (2024).
[34] Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, Xiang Wang, and Xiangnan He. 2024. Llara: Large language-recommendation assistant. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1785–1795.
[35] Zijie Lin, Yang Zhang, Xiaoyan Zhao, Fengbin Zhu, Fuli Feng, and Tat-Seng Chua. 2025. IGD: Token Decisiveness Modeling via Information Gain in LLMs for Personalized Recommendation. *arXiv preprint arXiv:2506.13229* (2025).
[36] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437* (2024).
[37] Chang Liu, Yimeng Bai, Xiaoyan Zhao, Yang Zhang, Fuli Feng, and Wenge Rong. 2025. DiscRec: Disentangled Semantic-Collaborative Modeling for Generative Recommendation. *arXiv preprint arXiv:2506.15576* (2025).
[38] Junling Liu, Chao Liu, Peilin Zhou, Renjie Lv, Kang Zhou, and Yan Zhang. 2023. Is chatgpt a good recommender? a preliminary study. *arXiv preprint arXiv:2304.10149* (2023).
[39] Qidong Liu, Xian Wu, Yejing Wang, Zijian Zhang, Feng Tian, Yefeng Zheng, and Xiangyu Zhao. 2024. Llm-esr: Large language models enhancement for long-tailed sequential recommendation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
[40] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2024. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems* 36 (2024).
[41] Yingqian Min, Zhipeng Chen, Jinhao Jiang, Jie Chen, Jia Deng, Yiwen Hu, Yiru Tang, Jiapeng Wang, Xiaoxue Cheng, Huatong Song, et al. 2024. Imitate, explore, and self-improve: A reproduction report on slow-thinking reasoning systems. *arXiv preprint arXiv:2412.09413* (2024).
[42] Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*. 188–197.
[43] OpenAI. 2024. GPT-4o mini: advancing cost-efficient intelligence. https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/
[44] OpenAI. 2024. Learning to Reason with LLMs. https://openai.com/index/learning-to-reason-with-llms/
[45] OpenAI. 2024. OpenAI o1 System Card. https://cdn.openai.com/o1-system-card-20241205.pdf
[46] OpenAI. 2025. OpenAI o3-mini. https://openai.com/index/openai-o3-mini/
[47] Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. 2023. Resurrecting recurrent neural networks for long sequences. In *International Conference on Machine Learning*. 26670–26698.
[48] Zhenting Qi, Mingyuan Ma, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. 2024. Mutual reasoning makes smaller llms stronger problem-solvers. *arXiv preprint arXiv:2408.06195* (2024).
[49] 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*. 3464–3475.
[50] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. 452–461.
[51] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300* (2024).
[52] Leheng Sheng, An Zhang, Yi Zhang, Yuxin Chen, Xiang Wang, and Tat-Seng Chua. 2024. Language Representations Can be What Recommenders Need: Findings and Potentials. *arXiv preprint arXiv:2407.05441* (2024).
[53] Lin Shi, Chiyu Ma, Wenhua Liang, Weicheng Ma, and Soroush Vosoughi. 2024. Judging the judges: A systematic investigation of position bias in pairwise comparative assessments by llms. *arXiv preprint arXiv:2406.07791* (2024).
[54] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. 2016. Mastering the game of Go with deep neural networks and tree search. *nature* 529, 7587 (2016), 484–489.
[55] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314* (2024).
[56] Zhongxiang Sun, Zihua Si, Xiaoxue Zang, Kai Zheng, Yang Song, Xiao Zhang, and Jun Xu. 2024. Large language models enhanced collaborative filtering. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 2178–2188.
[57] Zhongxiang Sun, Qipeng Wang, Weijie Yu, Xiaoxue Zang, Kai Zheng, Jun Xu, Xiao Zhang, Song Yang, and Han Li. 2025. ReARTeR: Retrieval-Augmented Reasoning with Trustworthy Process Rewarding. *arXiv preprint arXiv:2501.07861* (2025).
[58] Pawel Swietojanski, Jinyu Li, and Steve Renals. 2016. Learning hidden unit contributions for unsupervised acoustic model adaptation. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* (2016).
[59] Qiaoyu Tan, Jianwei Zhang, Jiangchao Yao, Ninghao Liu, Jingren Zhou, Hongxia Yang, and Xia Hu. 2021. Sparse-interest network for sequential recommendation. In *Proceedings of the 14th ACM international conference on web search and data mining*. 598–606.
[60] Jiakai Tang, Sunhao Dai, Teng Shi, Jun Xu, Xu Chen, Wen Chen, Wu Jian, and Yuning Jiang. 2025. Think Before Recommend: Unleashing the Latent Reasoning Power for Sequential Recommendation. *arXiv preprint arXiv:2503.22675* (2025).
[61] Qwen Team. 2024. QwQ: Reflect Deeply on the Boundaries of the Unknown. https://qwenlm.github.io/blog/qwq-32b-preview/
[62] Alicia Y Tsai, Adam Kraft, Long Jin, Chenwei Cai, Anahita Hosseini, Taibai Xu, Zemin Zhang, Lichan Hong, Ed H Chi, and Xinyang Yi. 2024. Leveraging LLM Reasoning Enhances Personalized Recommender Systems. *arXiv preprint arXiv:2408.00802* (2024).
[63] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 6000–6010.
[64] Jun Wang, Meng Fang, Ziyu Wan, Muning Wen, Jiachen Zhu, Anjie Liu, Ziqin Gong, Yan Song, Lei Chen, Lionel M Ni, et al. 2024. Openr: An open source framework for advanced reasoning with large language models. *arXiv preprint arXiv:2410.09671* (2024).
[65] Lei Wang and Ee-Peng Lim. 2023. Zero-shot next-item recommendation using large pretrained language models. *arXiv preprint arXiv:2304.03153* (2023).
[66] Yan Wang, Zhixuan Chu, Xin Ouyang, Simeng Wang, Hongyan Hao, Yue Shen, Jinjie Gu, Siqiao Xue, James Y Zhang, Qing Cui, et al. 2023. Enhancing recommender systems with large language model reasoning graphs. *arXiv preprint arXiv:2308.10835* (2023).
[67] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems* 35 (2022), 24824–24837.
[68] Yunjia Xi, Weiwen Liu, Jianghao Lin, Xiaoling Cai, Hong Zhu, Jieming Zhu, Bo Chen, Ruiming Tang, Weinan Zhang, and Yong Yu. 2024. Towards open-world recommendation with knowledge augmentation from large language models. In *Proceedings of the 18th ACM Conference on Recommender Systems*. 12–22.
[69] Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, James Xu Zhao, Min-Yen Kan, Junxian He, and Michael Xie. 2023. Self-evaluation guided beam search for reasoning.

Advances in Neural Information Processing Systems 36 (2023), 41618–41650.

[70] Huajian Xin, ZZ Ren, Junxiao Song, Zhihong Shao, Wanjia Zhao, Haocheng Wang, Bo Liu, Liyue Zhang, Xuan Lu, Qiushi Du, et al. 2024. Deepseek-prover-v1. 5: Harnessing proof assistant feedback for reinforcement learning and monte-carlo tree search. *arXiv preprint arXiv:2408.08152* (2024).

[71] Wujiang Xu, Zujie Liang, Jiaojiao Han, Xuying Ning, Wenfang Lin, Linxun Chen, Feng Wei, and Yongfeng Zhang. 2024. Slmrec: empowering small language models for sequential recommendation. *arXiv e-prints* (2024), arXiv–2405.

[72] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115* (2024).

[73] An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, et al. 2024. Qwen2. 5-math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122* (2024).

[74] Yuhao Yang, Chao Huang, Lianghao Xia, Chunzhen Huang, Da Luo, and Kangyi Lin. 2023. Debiased contrastive learning for sequential recommendation. In *Proceedings of the ACM web conference 2023*. 1063–1073.

[75] Xinhao Yao, Ruifeng Ren, Yun Liao, and Yong Liu. 2025. Unveiling the Mechanisms of Explicit CoT Training: How Chain-of-Thought Enhances Reasoning Generalization. *arXiv preprint arXiv:2502.04667* (2025).

[76] Zheng Yuan, Fajie Yuan, Yu Song, Youhua Li, Junchen Fu, Fei Yang, Yunzhu Pan, and Yongxin Ni. 2023. Where to go next for recommender systems? id-vs. modality-based recommender models revisited. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2639–2649.

[77] Zhenrui Yue, Yueqi Wang, Zhankui He, Huimin Zeng, Julian McAuley, and Dong Wang. 2024. Linear recurrent units for sequential recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 930–938.

[78] Zhenrui Yue, Honglei Zhuang, Aijun Bai, Kai Hui, Rolf Jagerman, Hansi Zeng, Zhen Qin, Dong Wang, Xuanhui Wang, and Michael Bendersky. 2024. Inference scaling for long-context retrieval augmented generation. *arXiv preprint arXiv:2410.04343* (2024).

[79] Zhiyuan Zeng, Qinyuan Cheng, Zhangyue Yin, Bo Wang, Shimin Li, Yunhua Zhou, Qipeng Guo, Xuanjing Huang, and Xipeng Qiu. 2024. Scaling of search and learning: A roadmap to reproduce o1 from reinforcement learning perspective. *arXiv preprint arXiv:2412.14135* (2024).

[80] Jiaqi Zhai, Lucy Liao, Xing Liu, Yueming Wang, Rui Li, Xuan Cao, Leon Gao, Zhaojie Gong, Fangda Gu, Jiayuan He, Yinghai Lu, and Yu Shi. 2024. Actions speak louder than words: trillion-parameter sequential transducers for generative recommendations. In *Proceedings of the 41st International Conference on Machine Learning (ICML'24)*. 26 pages.

[81] Buyun Zhang, Liang Luo, Yuxin Chen, Jade Nie, Xi Liu, Daifeng Guo, Yanli Zhao, Shen Li, Yuchen Hao, Yantao Yao, et al. 2024. Wukong: Towards a scaling law for large-scale recommendation. *arXiv preprint arXiv:2403.02545* (2024).

[82] Di Zhang, Xiaoshui Huang, Dongzhan Zhou, Yuqiang Li, and Wanli Ouyang. 2024. Accessing gpt-4 level mathematical olympiad solutions via monte carlo tree self-refine with llama-3 8b. *arXiv preprint arXiv:2406.07394* (2024).

[83] Gaowei Zhang, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, and Ji-Rong Wen. 2024. Scaling law of large sequential recommendation models. In *Proceedings of the 18th ACM Conference on Recommender Systems*. 444–453.

[84] Yang Zhang, Keqin Bao, Ming Yan, Wenjie Wang, Fuli Feng, and Xiangnan He. 2024. Text-like Encoding of Collaborative Information in Large Language Models for Recommendation. *arXiv preprint arXiv:2406.03210* (2024).

[85] Yuxiang Zhang, Shangxi Wu, Yuqi Yang, Jiangming Shu, Jinlin Xiao, Chao Kong, and Jitao Sang. 2024. o1-coder: an o1 replication for coding. *arXiv preprint arXiv:2412.00154* (2024).

[86] Yang Zhang, Wenxin Xu, Xiaoyan Zhao, Wenjie Wang, Fuli Feng, Xiangnan He, and Tat-Seng Chua. 2025. Reinforced Latent Reasoning for LLM-based Recommendation. *arXiv preprint arXiv:2505.19092* (2025).

[87] Xiaoyan Zhao, Yang Deng, Wenjie Wang, Hong Cheng, Rui Zhang, See-Kiong Ng, Tat-Seng Chua, et al. 2025. Exploring the Impact of Personality Traits on Conversational Recommender Systems: A Simulation with Large Language Models. *arXiv preprint arXiv:2504.12313* (2025).

[88] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems* 36 (2024).

[89] 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*. 1893–1902.