AGP: Auto-Guided Prompt Refinement for Personalized Reranking in Recommender Systems

Chen Wang[*]
University of Illinois at Chicago
Chicago, Illinois, USA
cwang266@uic.edu

Mingdai Yang[*]
The University of Chicago
Chicago, Illinois, USA
frankyang@uchicago.edu

Zhiwei Liu[†]
Salesforce AI Research
Palo Alto, California, USA
zhiweiliu@salesforce.com

Pan Li
Georgia Institute of Technology
Atlanta, Georgia, USA
pan.li@scheller.gatech.edu

Linsey Pang
Salesforce AI Research
Palo Alto, California, USA
panglinsey@gmail.com

Qingsong Wen
Squirrel Ai Learning
Bellevue, Washington
USA
qingsongedu@gmail.com

Philip S. Yu
University of Illinois at Chicago
Chicago, Illinois, USA
psyu@uic.edu

ABSTRACT

Reranking plays a critical role in recommendation systems by refining initial predictions to better reflect user preferences. While large language models (LLMs) have shown promise in enhancing reranking through contextual reasoning, they still rely heavily on manually crafted prompts—an approach that is both labor-intensive and difficult to scale. Although prompt optimization has been studied in domains like question answering and news recommendation, its adaptation to general item recommendation remains limited due to the unstructured and inconsistent nature of item metadata. To address these challenges, we propose **Auto-Guided Prompt Refinement** (AGP), a novel framework that *automatically refines user profile generation prompts* instead of reranking prompts directly. AGP leverages **position-based feedback**, which encodes item-level ranking misalignments, and introduces **batched training with aggregated feedback** to ensure robust and generalizable prompt updates. Experimental results on *Amazon Movies & TV, Yelp, and Goodreads* demonstrate AGP's effectiveness. With only 100 training users, AGP improves NDCG@10 by **5.61%**, **2.46%**, and **6.18%** when reranking SASRec, and by **9.36%**, **7.98%**, and **20.68%** when reranking LightGCN. These results highlight AGP's potential as a scalable, automated solution for LLM-based personalized reranking. **Code:** https://github.com/ChenMetanoia/AGP

CCS CONCEPTS

• **Information systems → Recommender systems**.

[*]These authors contributed equally to this work.
[†]Corresponding author

WSDM '26, February 22–26, 2026, Boise, ID, USA
© 2026 ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/3779211.3795737

KEYWORDS

Prompt Optimization, Recommender Systems, Large Language Models (LLMs), Collaborative Filtering, Reranking

ACM Reference Format:
Chen Wang, Mingdai Yang, Zhiwei Liu, Pan Li, Linsey Pang, Qingsong Wen, and Philip S. Yu. 2026. AGP: Auto-Guided Prompt Refinement for Personalized Reranking in Recommender Systems. In *Proceedings of Proceedings of the 19th ACM International Conference on Web Search and Data Mining (WSDM '26)*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3779211.3795737

1 INTRODUCTION

Reranking, which refines initial recommendations to better align with user preferences, plays a pivotal role in improving recommendation quality [9, 10, 13]. Recent advancements in large language models (LLMs) have shown promise for reranking tasks by capturing complex user interests via contextual understanding [2, 12, 18, 19].

However, their effectiveness relies heavily on manually crafting prompts – a labor-intensive process that demands significant domain expertise and limits scalability. Moreover, manually designed prompts struggle to address the complexity and diversity of user preferences. For instance, crafting prompts to capture nuanced user interests from user-item interactions, such as item titles or descriptions, often requires iterative trial-and-error. This process is not only time-consuming but also prone to suboptimal results due to its reliance on intuition rather than systematic optimization. Moreover, static prompts fail to adapt to dynamic datasets and evolving user behaviors, limiting their ability to deliver personalized recommendations at scale.

Existing research on prompt optimization largely focuses on tasks like question answering [14, 15, 17], mathematical reasoning [16, 17], and news recommendation [11]. *RecPrompt* [11] introduces a self-tuning prompting framework incorporating TopicScore to enhance explainability in news recommendations. However, this approach relies on structured content and topical consistency, making it less applicable to heterogeneous item recommendation scenarios, where item metadata can be inconsistent, unstructured, and user-generated. Current optimization methods [5, 11] usually rely on aggregated ranking metrics like AUC or NDCG, which are useful for performance evaluation but insufficient for direct optimization guidance. Input reranking approaches [3] reorder items based on relevance and exposure but do not offer structured feedback to refine user preference modeling. A more interpretable and systematic strategy is needed to close the gap between LLM-based reranking and explicit user feedback.

To address these challenges, in this paper we propose **AGP: Auto-Guided Prompt Refinement for Personalized Reranking**, a novel framework that optimizes *user profile generation prompts* rather than directly modifying reranking prompts. By refining user profiles, AGP enables LLMs to better capture personalized interests, leading to more effective and explainable reranking. AGP improves optimization through a **batched training with aggregated feedback** mechanism. Instead of refining prompts on a per-user basis, AGP evaluates multiple users simultaneously, preventing overfitting to individual cases and enhancing generalization. During each training iteration, AGP systematically analyzes why the generated user profiles fail to prioritize ground-truth items. To guide the refinement process, AGP introduces **position-based feedback**, which explicitly signals ranking misalignments by comparing predicted and ideal item positions. For example, if a relevant item is ranked 3rd instead of 1st, AGP encodes this discrepancy as a structured textual signal. Unlike traditional ranking metrics such as NDCG, which offer only aggregate performance scores, position-based feedback provides clear and interpretable correction cues. These correction signals are aggregated across batches of users to generate a concise summary of necessary prompt adjustments. The user profile generation prompt is then refined by modifying its content based on this summarized feedback—such as adjusting phrasing, emphasizing particular item features, or reordering information. This process is repeated iteratively, enabling AGP to continuously refine the prompt over multiple training rounds. Integrating structured feedback and batch-level refinement, AGP enables more interpretable, scalable personalized reranking with LLMs.

Our Contributions.

- **User Profile Generation Prompt.** We propose refining a *user profile generation prompt*—as opposed to a single-step rerank prompt—to more effectively capture user preferences from noisy textual data. This design enables more nuanced and robust personalization.
- **Position-Based Feedback.** We introduce a *position-based feedback* mechanism that pinpoints item-level ranking misalignments, providing interpretable signals for iterative refinement. This approach proves more actionable than relying solely on aggregated metrics like NDCG.
- **Batched Training and Summarized Feedback.** We devise a *batched* optimization framework that aggregates feedback across multiple users, mitigating overfitting to individual quirks and ensuring updates remain broadly applicable.

2 METHODOLOGY

In this section, we formalize the reranking task, introduce our Auto Prompt Optimization Framework, and detail each of its components. We first define the problem setup and notation, then elaborate on the user profile generation process, the evaluation and feedback mechanism, and finally the iterative prompt optimization strategy.

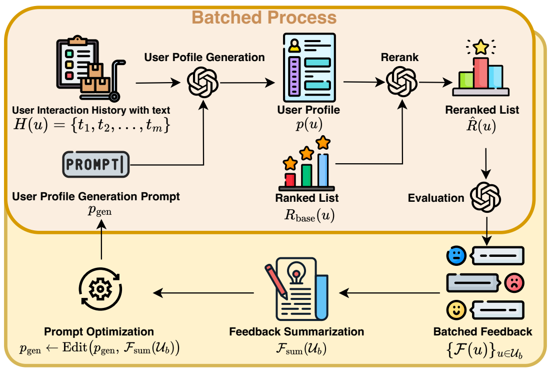

Batched Process
User Pofile Generation
Rerank
User Interaction History with text
$H(u) = \{t_1, t_2, \ldots, t_m\}$
User Profile
$p(u)$
Reranked List
$\hat{R}(u)$
PROMPT
User Profile Generation Prompt
p_{gen}
Ranked List
$R_{\text{base}}(u)$
Evaluation
Prompt Optimization
$p_{\text{gen}} \leftarrow \text{Edit}(p_{\text{gen}}, \mathcal{F}_{\text{sum}}(\mathcal{U}_b))$
Feedback Summarization
$\mathcal{F}_{\text{sum}}(\mathcal{U}_b)$
Batched Feedback
$\{\mathcal{F}(u)\}_{u \in \mathcal{U}_b}$

Figure 1: The pipeline of the AutoGuidePrompt (AGP) framework, illustrating the process from user interaction history to optimized reranked lists.

2.1 Problem Setup and Notation

Given a set of users $\mathcal{U}$, each user $u \in \mathcal{U}$ has an interaction history $H(u) = \{t_1, t_2, \ldots, t_m\}$, where each t_j is the *textual title* of an interacted item. Unlike ID-based approaches, this representation leverages semantic knowledge, enabling LLMs to infer preferences.

A recommender system provides a *baseline ranking* $R_{\text{base}}(u) = \{i_1, i_2, \ldots, i_k\}$, where items are ranked by predicted relevance. The goal is to refine $R_{\text{base}}(u)$ into an optimized ranking $R_{\text{LLM}}(u)$ that better aligns with user preferences, and the reranking function is:

$$f : (\mathcal{U}, R_{\text{base}}) \rightarrow R_{\text{LLM}}. \tag{1}$$

Challenges. Applying LLMs to item recommendations presents two key challenges:

- **Disparate and Noisy Item Information:** Unlike structured domains like news recommendation, item representations in recommender systems vary widely, with metadata that can be **incomplete, inconsistent, or redundant** (e.g., short titles, varying descriptions, or noisy user-generated content). This variability makes it difficult for LLMs to infer structured user preferences.
- **Lack of Direct Optimization Signals:** Existing methods (e.g., RecPrompt) optimize prompts using aggregated ranking metrics (AUC, NDCG), which measure performance but do not directly provide optimization guidance. A more interpretable strategy is needed to refine prompts using explicit ranking signals.

To address these, we propose **AGP**, which optimizes *user profile generation prompts* instead of reranking prompts, incorporates structured feedback, and iteratively refines prompts to improve personalization and ranking quality.

2.2 User Profile Generation with a Learned Prompt

AGP optimizes a shared *profile-generation prompt* p_{gen} to construct personalized user profiles. Unlike manually crafted prompts, p_{gen} is iteratively refined through batch training to capture generalizable patterns across users.

For each user u, AGP generates a profile based on two inputs: (1) the user's text-based interaction history $H(u)$ and (2) the current version of p_{gen}. The LLM synthesizes these inputs to generate a structured profile:

$$p(u) = \text{LLM}\big(H(u),\, p_{\text{gen}}\big). \tag{2}$$

Since $H(u)$ consists of item titles, the LLM extracts thematic preferences (e.g., *science fiction, deep learning*) while p_{gen} provides structure and emphasis.

The generated profile $p(u)$ is then used to *rerank* the baseline recommendation list $R_{\text{base}}(u)$, where the LLM refines rankings as:

$$\hat{R}(u) = \text{LLM}\big(p(u),\, R_{\text{base}}(u)\big). \tag{3}$$

This two-step process enhances LLM reasoning by summarizing user interests before reranking, allowing informed and context-aware ranking decisions. Since AGP refines profile generation rather than modifying raw rankings, it preserves user-specific insights while generalizing effectively across different users.

2.3 Position-Based Evaluation and Feedback

To refine reranking, we introduce *position-based feedback*, a more interpretable alternative to aggregated metrics like NDCG. Given a user u, we define a set of ground-truth relevant items $G(u) \subseteq \hat{R}(u)$ in the reranked list $\hat{R}(u)$. For each item $i \in G(u)$, we record its actual position $r_{\hat{R}}(i,u)$ and compare it to its target position $r_{\text{target}}(i,u)$. If an item should ideally be ranked in the top-3 but appears at position 5, the LLM receives a correction signal.

This process generates feedback signals:

$$\mathcal{F}(u) = \Big\{\big(r_{\hat{R}}(i,u),\, r_{\text{target}}(i,u)\big) \;\Big|\; i \in G(u)\Big\}. \tag{4}$$

These signals specify ranking deviations, providing direct and interpretable instructions for refinement. Unlike aggregated ranking metrics, which provide an overall evaluation score, position-based feedback delivers explicit corrections for each item, making adjustments more targeted.

Position-based feedback offers two key advantages. First, it improves interpretability by providing explicit positional corrections rather than relying on abstract scores. Second, it enables fine-grained ranking adjustments, ensuring that high-relevance items receive stronger refinements while less critical items undergo minor tweaks. This process naturally integrates with AGP's optimization strategy by offering direct ranking signals that guide systematic improvements.

2.4 Batch Formation and Optimization

AGP refines the profile-generation prompt p_{gen} through iterative batch training. In each iteration, a batch $\mathcal{U}_b$ of users is sampled. For each user $u \in \mathcal{U}_b$, a profile $p(u) = \text{LLM}\big(H(u),\, p_{\text{gen}}\big)$ is generated and used to rerank the candidate list into $\hat{R}(u)$. Position-based feedback $\mathcal{F}(u)$, described in Sec. 2.3, identifies discrepancies between predicted and ground-truth item ranks.

Rather than applying user-specific feedback directly, AGP instructs the LLM to summarize the batch feedback $\{\mathcal{F}(u) \mid u \in \mathcal{U}_b\}$ into a generalized editing instruction:

$$\mathcal{F}_{\text{sum}}(\mathcal{U}_b) = \text{LLM-Summarize}\big(\{\mathcal{F}(u)\}\big). \tag{5}$$

This abstracts away personalized details and reveals shared refinement signals. For example, if one user prefers "historical dramas set in 18th-century Europe" and another prefers "war documentaries from the 1940s," the LLM may summarize the instruction as *"Highlight temporal aspects of item content when modeling user preferences."*

This generalized instruction guides prompt refinement:

$$p_{\text{gen}} \leftarrow \text{Edit}\big(p_{\text{gen}},\, \mathcal{F}_{\text{sum}}(\mathcal{U}_b)\big), \tag{6}$$

where $\text{Edit}(\cdot)$ applies the instruction to produce a more transferable and effective prompt.

AGP repeats this process over multiple iterations, continuously generating profiles, collecting feedback, and refining the prompt. By summarizing batch-level cues into general instructions, AGP improves personalization while maintaining strong generalization to unseen users.

3 EXPERIMENTS

3.1 Experimental Settings

We evaluate AGP on three public datasets: *Amazon Movies & TV* (95K users, 43K items), *Yelp* (65K users, 33K items), and *Goodreads* (13K users, 25K items). We adopt a leave-one-out strategy: the latest interaction is used for testing, and the second-latest for validation. Two pretrained recommenders—**LightGCN** [6] and **SASRec** [7]—generate top-10 candidates per user. From each dataset, we randomly sample 300 users (ensuring ground-truth item inclusion) for evaluation. The prompt is trained on a disjoint set of 100 users for up to 10 epochs. We vary history lengths {5, 10, 20} and batch sizes {5, 10, 20}. To avoid data leakage, we ensure strict separation between training and test users. Ground-truth item ranks are used only during training to simulate position-based feedback, which mirrors standard supervised learning. At inference time, AGP reranks items without any access to ground-truth information.

AGP is compared against two LLM baselines: *LLM-Dir* (direct ranking) and *LLM-CoT* (chain-of-thought reranking [8]). We test four LLMs: GPT-4o, GPT-4o-Mini, GPT-o3-Mini, and DeepSeek-V3 [4], using **NDCG@10** as the evaluation metric.

Relation to Prior Prompting Methods: Recent frameworks like RecPrompt [11] and GoT [1] are task-specific and not directly comparable. RecPrompt focuses on news recommendation with clean metadata and explanation-based prompts, without batch-level generalization or ranking supervision. GoT targets symbolic tasks via graph-based reasoning, not personalization. In contrast, AGP addresses general item recommendation under noisy metadata by refining prompts with batched, position-aware feedback.

3.2 Results and Discussion

Table 1 summarizes key findings: **(1) AGP effectiveness in item recommendation:** AGP consistently improves reranking performance across all datasets and backbone recommenders, showing that prompt refinement via feedback is effective. This reduces reliance on manual prompt tuning and improves ranking quality, especially for **personalized recommendation tasks**. **(2) LLM reranker performance on SASRec vs. LightGCN:** LLM-based rerankers yield greater improvements on SASRec than LightGCN, likely due to better alignment with SASRec's sequential modeling. In contrast, LightGCN relies on graph-level interactions that are less compatible with language-based reasoning. **(3) Yelp dataset challenges:** The smallest gains occur on Yelp, primarily due to its sparse and noisy text. Business names are often ambiguous (e.g., "Luna" could refer to various businesses), and descriptions are frequently vague or missing, making it hard for LLMs to infer item semantics. This is reflected in the weak performance of LLM-Dir and LLM-CoT, which often underperform the base recommenders on Yelp. Despite these limitations, AGP improves performance by refining prompts with collective position-based feedback, demonstrating robustness under low-signal conditions. **(4) AGP vs. LLM strength:** While strong LLMs like GPT-4o naturally perform better, AGP yields consistent improvements across model sizes. Notably, AGP boosts weaker models like o3-Mini and DeepSeek-V3, which struggle with prompt-only reranking. This shows that AGP's gains stem from its feedback-driven optimization, not just LLM power, and generalize well across LLM scales.

3.3 Ablation Study

To evaluate the impact of design choices in our framework, we conduct an ablation study using GPT-4o on the AMZ dataset, focusing on three key aspects: the effect of summarization in reducing overfitting, the influence of batch size and sequence length on reranking performance, and the effectiveness of position-based feedback.

Summarization Impact on Overfitting: We analyze the effect of summarization by comparing training and test performance with and without summarization. As shown in Figure (2, left), summarization prevents excessive fitting to the training data while improving test performance, enhancing generalization by filtering redundant information and refining prompt optimization. **Effectiveness of Batch Size and Sequence Length:** We examine how batch size and sequence length impact reranking effectiveness. Figure (2, middle) shows that the best performance is achieved with a batch size of 10 and a sequence length of 5, suggesting an optimal balance between convergence and generalization. Larger batch sizes stabilize training, while shorter sequences reduce noise from long interaction histories, highlighting the importance of careful hyperparameter selection for improving LLM reranking. **Effectiveness of Position-Based Feedback (PBF):** We further investigate the impact of PBF on reranking quality across datasets. Figure 3 demonstrates that incorporating PBF improves both *NDCG@10* and average ranking position across all datasets. The gains are more significant in datasets with high variability in ranking scores, indicating that PBF helps LLMs better capture relative item importance. This result highlights its potential in enhancing ranking stability and improving personalization in LLM-based reranking systems.

Table 1: Reranking performance (N@10) across datasets. "T100" denotes AGP trained on 100 users. Bold values indicate the best performance within the same LLM model.

Model	Method	N@10		
		AMZ	YELP	GR
LightGCN	Base	0.513	0.501	0.474
	+ LLM-Dir (4o)	0.547	0.491	0.555
	+ LLM-CoT (4o)	0.551	0.491	0.560
	+ AGP-T100 (4o)	**0.553**	**0.502**	**0.572**
	+ LLM-Dir (4o-Mini)	0.553	0.484	0.548
	+ LLM-CoT (4o-Mini)	0.553	0.481	0.547
	+ AGP-T100 (4o-Mini)	**0.561**	**0.493**	**0.553**
	+ LLM-Dir (o3-Mini)	0.542	0.538	0.540
	+ LLM-CoT (o3-Mini)	0.543	0.531	0.542
	+ AGP-T100 (o3-Mini)	**0.558**	**0.541**	**0.548**
	+ LLM-Dir (DeepSeek)	0.546	0.496	0.533
	+ LLM-CoT (DeepSeek)	0.547	0.498	0.544
	+ AGP-T100 (DeepSeek)	**0.551**	**0.504**	**0.564**
SASRec	Base	0.659	0.528	0.599
	+ LLM-Dir (4o)	0.671	0.510	0.624
	+ LLM-CoT (4o)	0.664	0.521	0.610
	+ AGP-T100 (4o)	**0.696**	**0.530**	**0.636**
	+ LLM-Dir (4o-Mini)	0.654	0.502	**0.631**
	+ LLM-CoT (4o-Mini)	0.656	0.507	0.615
	+ AGP-T100 (4o-Mini)	**0.658**	**0.517**	0.627
	+ LLM-Dir (o3-Mini)	0.658	0.527	0.587
	+ LLM-CoT (o3-Mini)	0.663	0.528	0.585
	+ AGP-T100 (o3-Mini)	**0.683**	**0.541**	**0.595**
	+ LLM-Dir (DeepSeek)	0.648	0.512	0.622
	+ LLM-CoT (DeepSeek)	0.655	0.519	0.610
	+ AGP-T100 (DeepSeek)	**0.687**	**0.523**	**0.636**

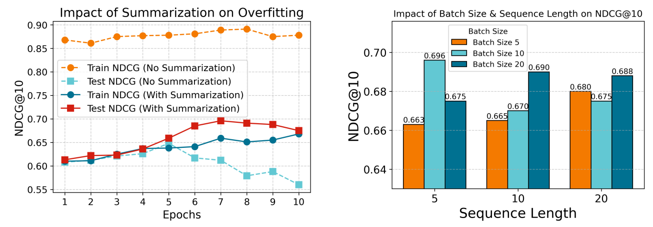


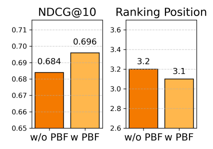
Figure 2: Ablation study on summarization (left) and batch size/sequence length impact (right) using GPT-4o on the AMZ dataset. Summarization reduces overfitting, while batch size 10 and sequence length 5 yield optimal ranking performance.

3.4 Training Efficiency

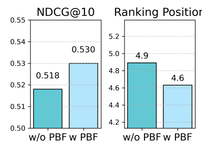
We evaluate the efficiency of AGP training by analyzing the API call calculations and performance scaling. The total API calls per training stage follow the formula:

$$\text{API Calls} = (\text{batch\_size} \times 3 + 2) \times \frac{100}{\text{batch\_size}} \tag{7}$$

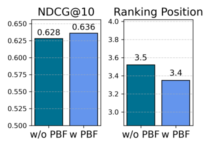
where 3 corresponds to generating a user profile, reranking, and computing the loss per user, while 2 accounts for summarization



(a) Amazon Dataset



(b) Yelp Dataset



(c) Goodreads Dataset

Figure 3: Ablation study on Position-Based Feedback.

and prompt optimization per batch. To further assess AGP's efficiency, we trained it on the AMZ dataset with GPT-4o using 700 users instead of 100. The NDCG@10 increased marginally from 0.696 to 0.705 (a 1.29% increase), demonstrating that AGP maintains competitive performance with significantly fewer training samples. This result underscores AGP's effectiveness and efficiency, reducing computational costs while preserving high reranking quality.

4 CONCLUSION

We propose **AGP**, a framework that optimizes *user profile generation prompts* to enhance LLM-based reranking. AGP employs **batched feedback** and **position-based feedback** to improve generalization and ranking accuracy without manual prompt tuning. Extensive experiments confirm its effectiveness across datasets and model sizes. Future work will explore applying AGP to fully open-source LLMs to assess its performance under constrained compute budgets and extend its applicability in practical deployment scenarios.

REFERENCES

[1] Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, et al. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 17682–17690.

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

[3] Mohsen Dehghankar and Abolfazl Asudeh. 2024. Rank It, Then Ask It: Input Reranking for Maximizing the Performance of LLMs on Symmetric Tasks. *arXiv preprint arXiv:2412.00546* (2024).

[4] DeepSeek-AI et al. 2024. DeepSeek-V3 Technical Report. arXiv:2412.19437 [cs.CL] https://arxiv.org/abs/2412.19437

[5] Jingtong Gao, Bo Chen, Xiangyu Zhao, Weiwen Liu, Xiangyang Li, Yichao Wang, Zijian Zhang, Wanyu Wang, Yuyang Ye, Shanru Lin, et al. 2024. LLM-enhanced Reranking in Recommender Systems. *arXiv preprint arXiv:2406.12433* (2024).

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

[7] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 197–206.

[8] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems* 35 (2022), 22199–22213.

[9] Xinyi Li, Yifan Chen, Benjamin Pettit, and Maarten De Rijke. 2019. Personalised reranking of paper recommendations using paper content and user behavior. *ACM Transactions on Information Systems (TOIS)* 37, 3 (2019), 1–23.

[10] Xiao Lin, Xiaokai Chen, Chenyang Wang, Hantao Shu, Linfeng Song, Biao Li, and Peng Jiang. 2024. Discrete conditional diffusion for reranking in recommendation. In *Companion Proceedings of the ACM on Web Conference 2024*. 161–169.

[11] Dairui Liu, Boming Yang, Honghui Du, Derek Greene, Neil Hurley, Aonghus Lawlor, Ruihai Dong, and Irene Li. 2024. RecPrompt: A Self-tuning Prompting Framework for News Recommendation Using Large Language Models. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management* (Boise, ID, USA) *(CIKM '24)*. Association for Computing Machinery, New York, NY, USA, 3902–3906. https://doi.org/10.1145/3627673.3679987

[12] Sichun Luo, Bowei He, Haohan Zhao, Wei Shao, Yanlin Qi, Yinya Huang, Aojun Zhou, Yuxuan Yao, Zongpeng Li, Yuanzhang Xiao, et al. 2024. Recranker: Instruction tuning large language model as ranker for top-k recommendation. *ACM Transactions on Information Systems* (2024).

[13] Changhua Pei, Yi Zhang, Yongfeng Zhang, Fei Sun, Xiao Lin, Hanxiao Sun, Jian Wu, Peng Jiang, Junfeng Ge, Wenwu Ou, et al. 2019. Personalized re-ranking for recommendation. In *Proceedings of the 13th ACM conference on recommender systems*. 3–11.

[14] Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic Prompt Optimization with "Gradient Descent" and Beam Search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 7957–7968.

[15] Antonio Sabbatella, Andrea Ponti, Ilaria Giordani, Antonio Candelieri, and Francesco Archetti. 2024. Prompt Optimization in Large Language Models. *Mathematics* 12, 6 (2024), 929.

[16] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, YK Li, Yu Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300* (2024).

[17] Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. 2025. Optimizing generative AI by backpropagating language model feedback. *Nature* 639, 8055 (2025), 609–616.

[18] Haobo Zhang, Qiannan Zhu, and Zhicheng Dou. 2025. Enhancing Reranking for Recommendation with LLMs through User Preference Retrieval. In *Proceedings of the 31st International Conference on Computational Linguistics*. 658–671.

[19] Yongfeng Zhang, Zhiwei Liu, Qingsong Wen, Linsey Pang, Wei Liu, and Philip S Yu. 2024. AI Agent for Information Retrieval: Generating and Ranking. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*. 5605–5607.