

1 A Appendix

2 A.1 Notations

Table 2: Description of major notations, ordered by appearance.

Notation	Description
$\mathcal{U}, \mathcal{I}, \mathcal{R}$	user set, item set, interaction set
u, i, \mathcal{A}	user, item, attribute set of user or item
f	scoring function
$\mathcal{X}, \text{Rel}(\cdot, \cdot), \text{Div}(\cdot)$	set of recommended items, relevance metric, diversity metric
\mathcal{S}	set of sampled items for constructing ToP
Prompt, LLM(\cdot)	input prompt for LLM, output of LLM
$\mathcal{T}, \mathcal{V}, \mathcal{E}$	tree, node set of \mathcal{T} , edge set of \mathcal{T}
v, e	node of \mathcal{T} , edge of \mathcal{T}
\mathcal{R}_u	user u 's interactions
$\langle \cdot, \cdot \rangle, \text{Enc}(\cdot)$	cosine similarity, pretrained language encoder
freq_i	frequency of the preference associated with item i in user u 's history
$s(u, i), \lambda$	item selection score, hyperparameter
$\mathcal{R}', \mathcal{R}^+$	synthetic interactions, generated interactions from selected items
\mathcal{L}, θ	loss of recommender, parameter of recommender
$\ell(\cdot; \cdot), \nabla$	user local loss, gradient
Inf_u	influence of user u

3 A.2 Supplementary to the method

4 **Illustration of prompts.** We provide an illustration of the prompts used to complete the essential
5 processes of ToP-Rec, as summarized in Figure 4.

6 **Details of k-means-based item sampling.** In constructing ToP, we use a k-means-based method
7 to sample a smaller, text-rich subset of items for the LLM, ensuring it has sufficient knowledge of
8 item distribution range (cf. Eq.(1)). We now detail the k-means-based sampling method. First, we
9 encode each item's text-based attributes with a pretrained language model BERT [1]. Then, to avoid
10 excessive semantic repetition, we perform k-means clustering on the item embeddings and randomly
11 sample items from each cluster. Finally, we aggregate the sampled items and instruct the LLM to
12 partition the user preference space, forming a hierarchical tree structure.

13 **Details of ToP refinement.** To find candidate items matching the preferences of any user, we
14 preassign every item to a leaf node in ToP (cf. Eq.(3)). However, this assignment may lead to slight
15 load imbalance, with varying numbers of items assigned to different leaf nodes. Empirically, we
16 observe that the imbalance can carry over into the recommended items. Therefore, after assigning all
17 items, we introduce a refinement mechanism to improve load balance, involving two operations.

18 (1) Merge. Underloaded sibling leaves are merged to create a load-balanced leaf. This is achieved by
19 integrating the assigned items and instructing the LLM to summarize their corresponding preferences,
20 thus forming a merged leaf.

21 (2) Split. Leaves with excessive load are split into several load-balanced leaves. Given an excessively
22 loaded leaf, the LLM is prompted to further refine the corresponding preference into several finer-
23 grained ones, reassigning the items accordingly.

24 In practice, underloaded and excessively loaded leaves are identified using two predefined lower and
25 upper thresholds, enabling self-stabilization through refinement. To prevent infinite refinement, we
26 set the maximum number of operations as 10.

27 **Theoretical analysis of user selection.** To quantify user influence based on gradient alignment,
28 we assess the alignment between each user's local gradient and the model's parameter trajectory

Example prompts used for ToP construction, preference reasoning, and item matching.

Notation: Prompt_{ToP}.

Usage: ToP construction.

Content: You are an expert system designed to classify and organize user preferences based on the following information of item examples. Your task is to generate a multi-level tree structure for user preferences, called Tree of Preference (ToP), progressively refining them from broad to specific preferences. Each node represents a type of user preference, and each edge signifies a finer preference division from its parent node.

Constraints:

- Each node should be divided into 3-5 finer preferences (branches), except for leaf nodes.
- Use diverse preference-dividing criteria at each level.
- Nodes must represent clear, actionable preferences.

...

Items samples: {sampled_items_information}.

Return ToP in the following format: {ToP_format}.

Notation: Prompt_{PR}.

Usage: Preference reasoning.

Content: You are an expert system designed to capture user latent preferences through rationale reasoning. You will be provided with the user's profile, observed interactions with items, and a multi-level tree structure representing different user preferences, organized from broad to specific preferences (Tree of Preference, ToP). Using the profile and interactions, your task is to identify the user's latent preferences by exploring the ToP top-down, following the coarse-to-fine paths that best match the user's behavior and reasoning the rationale behind their actions, while identifying any unobserved preferences not reflected in the history.

Constraints:

- Perform a breadth-first search. At each level, select the preferences that best match the user behavior, storing the corresponding nodes.
- For subsequent levels, activate only the child nodes of the stored nodes at the previous level, continuing the selection until reaching final level.
- Reevaluate the exploration to identify any unobserved preferences, adding them if found, following the root-to-leaf path.
- Control the total number of selected paths as {number_of_paths}. The final output should be the leaf nodes of all selected paths.

...

User profile: {user_profile}.

User historical interactions: {user_interactions}.

ToP: {ToP_content}.

Return selected leaf nodes in the following format: {leaf_nodes_format}, along with a concise explanation of each selection in the format of {reason_format}.

Notation: Prompt_{IM}.

Usage: Item matching.

Content: You are an expert system designed to match items with user preferences. You will be provided with items and their specific information, along with a multi-level Tree of Preference (ToP) representing user preferences, structured from broad to specific divisions, where each node represents a preference type and each edge further refines the preferences. Your task is to find the most relevant preference for each item in ToP, moving through the levels of ToP based on the item's information, and return the final leaf node.

Constraints:

- At each level, select only the most appropriate node from the available options.
- For the next level, select only from the child nodes of the node selected in the previous level.

...

ToP: {ToP_content}.

Items information: {items_information}.

Return the selected leaf node in the following format: {leaf_node_format}.

Figure 4: Illustration of the prompts used in this work.

Table 3: Dataset statistics.

Dataset	# Interactions	# Users	# Items	% Density
Twitter	40,223	2,118	7,199	0.2639%
Weibo	661,783	14,663	5,711	0.7903%
Amazon	134,857	3,598	8,747	0.4285%

during gradient descent (see Definition 1). The intuition is that the stronger the alignment between a user’s optimization direction and that of the recommender model, the greater their contribution to the model’s improvement. Here, we present the theoretical analysis supporting our approach, starting with a theorem that establishes the upper bound of the convergence step in gradient descent (GD).

Theorem A.1. *Let $\mathcal{L}(\theta)$ be a convex loss function with learnable parameter θ , and let $\theta^* \in \arg \min_{\theta} \mathcal{L}(\theta)$ denote the optimal parameter. Define the initial error $e_0 := \mathcal{L}(\theta^0) - \mathcal{L}(\theta^*)$, and let $\nabla \mathcal{L}(\theta^t)$ denote the gradient at optimization step t . Assume \mathcal{L} is β -Lipschitz smooth. Then the number of steps T required to achieve $\mathcal{L}(\theta^T) - \mathcal{L}(\theta^*) \leq \varepsilon$ satisfies:*

$$T \leq \frac{2\beta(e_0 - \varepsilon)}{\min_{0 \leq t \leq B} \|\nabla \mathcal{L}(\theta^t)\|^2}, \quad (5)$$

where $B = \mathcal{O}(\frac{1}{\varepsilon})$.

Proof. Since \mathcal{L} is a convex function with β -Lipschitz smoothness, we can obtain the following inequality: $\mathcal{L}(\theta') \leq \mathcal{L}(\theta) + \nabla \mathcal{L}(\theta)^T(\theta' - \theta) + \frac{\beta}{2}\|\theta' - \theta\|^2$. Suppose the learning rate is α , we substitute the gradient descent update $\theta^{t+1} = \theta^t - \alpha \nabla \mathcal{L}(\theta^t)$:

$$\mathcal{L}(\theta^{t+1}) \leq \mathcal{L}(\theta^t) - \alpha \|\nabla \mathcal{L}(\theta^t)\|^2 + \frac{\alpha^2 \beta}{2} \|\nabla \mathcal{L}(\theta^t)\|^2. \quad (6)$$

Choosing the optimal learning rate $\alpha = \frac{1}{\beta}$ (maximizing the descent under smoothness), we can obtain:

$$\mathcal{L}(\theta^{t+1}) \leq \mathcal{L}(\theta^t) - \frac{1}{2\beta} \|\nabla \mathcal{L}(\theta^t)\|^2. \quad (7)$$

The decrease in the objective function per iteration has the lower bound:

$$\mathcal{L}(\theta^t) - \mathcal{L}(\theta^{t+1}) \geq \frac{1}{2\beta} \|\nabla \mathcal{L}(\theta^t)\|^2. \quad (8)$$

Since each epoch t reduces the error by at least $\frac{\|\nabla \mathcal{L}(\theta^t)\|^2}{2\beta}$, and from [2] we know that given ε , the convergence rate of GD method is $\mathcal{O}(\frac{1}{\varepsilon})$, thus the minimum number of epochs T satisfies $T \leq \frac{2\beta(e_0 - \varepsilon)}{\min_{t \leq B} \|\nabla \mathcal{L}(\theta^t)\|^2}$, where $B = \mathcal{O}(\frac{1}{\varepsilon})$. \square

Theorem 1 demonstrates the inverse relation between the convergence step and the gradient norm, which inspires our definition of the user influence. We use the inner product summation to measure the user influence, which essentially prioritizes users whose gradients exhibit persistent positive projection onto the global update trajectory. Users with high influence contribute more significantly to increasing the gradient norm’s magnitude, thereby enhancing the training efficiency and convergence performance according to Theorem 1.

A.3 More experiment settings

Dataset details. The statistics of the datasets used in this work are summarized in Table 3.

- **Twitter** [3]: This dataset is originally collected from Twitter for bot detection. We remove bot users, considering human tweet user as user and tweet as item. We consider user behaviors such as likes and retweets as interactions. User attributes include name, description, location, and more, and item attributes include content, retweet count, etc.

59 • **Weibo** [4]: This dataset is collected from Weibo, one of China’s largest social media platforms. We
60 treat microblog posts as items and reposts as interactions. User attributes include name, gender,
61 number of followers, etc. Item attributes include content and repost count.

62 • **Amazon** [5]: Amazon is an e-commerce dataset. Following [6], seven categories are combined:
63 Automotive, Cell Phones and Accessories, Clothing, Shoes and Jewelry, Electronics, Grocery and
64 Gourmet Food, Home and Kitchen, and Movies and TV. Attributes include item category, brand,
65 features, description, user reviews, and others.

66 For all datasets, we follow [7] to extract a subset of the original data, then drop items with no attributes
67 and apply 10-core filters (5-core for Twitter), resulting in a dataset that retains relatively informative
68 and active entities. For each user, we split their interactions into train, validation, and test sets with a
69 ratio of 0.6:0.2:0.2.

70 **Baseline details.** To ensure fairness, we use the same recommender backbone for ToP-Rec and
71 the baselines. For reranking methods like MMR and DPP, we found that using 10 times the top- k
72 value as the candidate list can bring the performance close to its peak. For LLM4Rerank-A and
73 LLM4Rerank-AD, we use twice the top- k value due to their instability. We categorize the nine
74 comparative methods used in this work into three types.

75 Heuristic methods:

76 • **Random**: Randomly generate a fix number of user-item interactions and add them to the original
77 history interaction set, then train the recommender.

78 • **MMR** [8]: A heuristic algorithm optimizing the trade-off between relevance and diversity, selecting
79 items that are pertinent to the query and minimally redundant.

80 • **DPP** [9]: DPP is a parametric model for selecting a diverse subset from a larger pool of items. [9]
81 accelerates DPP by proposing an algorithm that significantly speeds up the greedy MAP inference.

82 Conventional diversity-enhancing methods:

83 • **Box** [10]: A framework that enhances diversity and accuracy using box embedding to create
84 hypercubes for users and items, with a similarity scoring function to measure their relationship.

85 • **LCD-UC** [10]: To better balance accuracy and diversity, LCD-UC adds an L-Step, C-Step, and
86 D-Step design, along with an uncertainty masking mechanism, on top of box embedding.

87 • **CDM** [11]: A recommendation framework leveraging contrastive context encoder with attention
88 mechanisms, distilling knowledge from MMR-based teacher output, and combining scores for
89 diverse results.

90 LLM-based diversified recommenders:

91 • **LLM4Rerank-A** [12]: A reranking framework addressing accuracy, diversity, and fairness by
92 abstracting requirements into interconnected nodes and dynamically adjusting aspect priorities
93 through a Chain-of-Thought process. For LLM4Rerank-A, we set the LLM’s focus solely on
94 improving accuracy during reranking.

95 • **LLM4Rerank-AD** [12]: Similar to LLM4Rerank-A, the difference lies in setting the LLM’s focus
96 on improving both accuracy and diversity during reranking.

97 • **LLMRec-MMR** [13]: LLMRec uses LLMs to enhance recommendation relevance through three
98 graph augmentation strategies: reinforcing user-item interactions, improving item attributes, and
99 refining user profiling. MMR is then applied to balance diversity by reranking the list.

100 **More implementation details.** All experiments are conducted on a machine of Ubuntu 20.04
101 system with AMD EPYC 7763 (756GB memory) and NVIDIA RTX3090 GPU (24GB memory). All
102 models are implemented in PyTorch version 2.5.1 with CUDA version 11.8 and Python 3.10.15.

103 A.4 More experiment results

104 **Empirical analysis of user selection.** The effectiveness of influential user selection in ToP-Rec is
105 mainly influenced by three factors. First, the number of user groups determines group size, affecting
106 user selection granularity. Second, the dimension of gradient reduction impacts the amount of retained
107 information, and the step length k affects the influence evaluation period. We analyze their impact on
108 overall performance.

Figure 5(a) and (b) illustrate the changes in diversity and relevance on Twitter with different group numbers (maintaining a similar total number of augmented users) and different reduced dimensions. Increasing group size or dimensionality improves recall and category entropy, but with smaller marginal returns. This suggests that a small number of groups or dimensionality provides sufficient gains while keeping computational costs lower. Figure 5(c) visualizes the impact of step k , showing that smaller values (within 5) maintain low costs while yielding better performance. Larger step sizes degrade performance, likely due to the extended training history losing dynamic user influence.

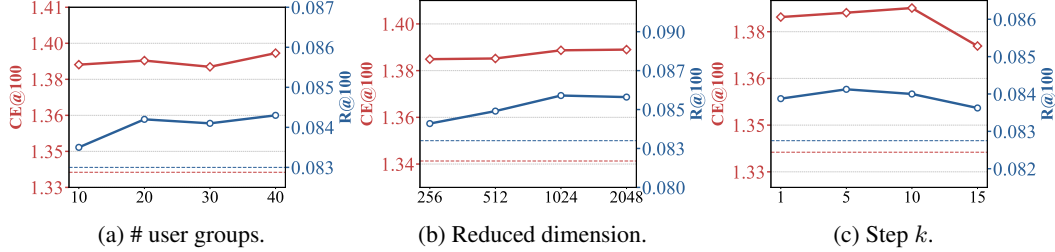


Figure 5: Impact of (a) number of user groups, (b) reduced dimension, and (c) step k on the performance of ToP-Rec. Dashed lines represent the performance of the backbone recommender.

Performance evaluation on more recommender backbones. We further evaluate the performance of ToP-Rec on two widely used recommenders, namely MF [14] and NGCF [15]. We set the hidden size to 32 and utilize the Adam optimizer with a learning rate of $5e-3$, keeping other settings the same as LightGCN. We compare ToP-Rec with representative baselines, all reporting relatively balanced results for diversity and relevance. Tables 4 and 5 show the performance comparison on Twitter, using MF and NGCF as backbones, respectively. * denotes the backbone model, and $+/ -$ indicates performance changes. The optimal performance is in bold, and the suboptimal is underlined. The results show that ToP-Rec achieves optimal performance in both cases, further demonstrating its generalizability.

Table 4: Performance comparison using MF as recommender backbone.

	R@50	R@100	CE@50	CE@100
MF*	0.0329	0.0501	1.2249	1.3189
MMR	0.0319 ⁻	0.0485 ⁻	1.2596 ⁺	1.3533 ⁺
LCD-UC	0.0327 ⁻	0.0503 ⁺	1.2558 ⁺	1.3504 ⁺
LLM4Re-AD	0.0323 ⁻	0.0498 ⁻	1.2620 ⁺	1.3444 ⁺
ToP-Rec	0.0345⁺	0.0510⁺	1.2980⁺	1.3813⁺

Table 5: Performance comparison using NGCF as recommender backbone.

	R@50	R@100	CE@50	CE@100
NGCF*	0.0468	0.0748	1.2478	1.2977
MMR	0.0435 ⁻	0.0708 ⁻	1.2800 ⁺	1.3261 ⁺
LCD-UC	0.0466 ⁻	0.0733 ⁻	1.2897 ⁺	1.3365 ⁺
LLM4Re-AD	0.0467 ⁻	0.0752 ⁺	1.2722 ⁺	1.3189 ⁺
ToP-Rec	0.0473⁺	0.0763⁺	1.3630⁺	1.4028⁺

Standard deviations. Table 6 presents the standard deviations for the recall and category entropy shown in Table 1.

Relevance-diversity trade-off. We provide a trade-off analysis on the Weibo and Amazon datasets, as shown in Figure 6. Our approach consistently achieves the best trade-off, demonstrating its outstanding performance.

Table 6: Standard deviation.

	Twitter				Weibo				Amazon			
	R@50	R@100	CE@50	CE@100	R@50	R@100	CE@50	CE@100	R@50	R@100	CE@50	CE@100
LightGCN*	0.0008	0.0007	0.0030	0.0038	0.0002	0.0006	0.0036	0.0026	0.0007	0.0006	0.0071	0.0057
Random	0.0014	0.0024	0.0131	0.0145	0.0003	0.0005	0.0039	0.0065	0.0015	0.0024	0.0108	0.0121
MMR	0.0004	0.0004	0.0027	0.0021	0.0002	0.0002	0.0084	0.0076	0.0014	0.0014	0.0030	0.0059
DPP	0.0017	0.0015	0.0108	0.0073	0.0006	0.0009	0.0054	0.0073	0.0006	0.0007	0.0053	0.0050
CDM	0.0012	0.0008	0.0046	0.0051	0.0002	0.0003	0.0057	0.0055	0.0021	0.0019	0.0161	0.0162
Box	0.0018	0.0028	0.0058	0.0044	0.0028	0.0016	0.0078	0.0121	0.0023	0.0020	0.0072	0.0089
LCD-UC	0.0016	0.0032	0.0093	0.0085	0.0016	0.0024	0.0081	0.0039	0.0021	0.0018	0.0082	0.0078
LLMRec-MMR	0.0011	0.0012	0.0044	0.0057	0.0006	0.0008	0.0062	0.0050	0.0017	0.0019	0.0101	0.0092
LLM4Re-A	0.0008	0.0010	0.0098	0.0083	0.0003	0.0008	0.0055	0.0079	0.0013	0.0015	0.0089	0.0091
LLM4Re-AD	0.0010	0.0012	0.0161	0.0147	0.0004	0.0005	0.0096	0.0065	0.0025	0.0028	0.0156	0.0114
ToP-Rec	0.0012	0.0013	0.0127	0.0141	0.0006	0.0008	0.0066	0.0059	0.0010	0.0006	0.0142	0.0136

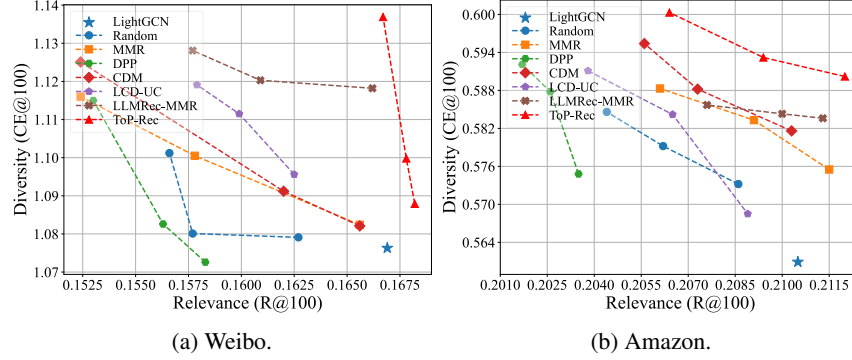


Figure 6: Diversity-relevance trade-off comparison. The upper-right represents the ideal.

Hyperparameter analysis. We examine the impact of the number of selected leaf nodes per user on ToP-Rec’s performance. Varying the number of selected leaves among $\{4, 5, 6, 7\}$, the results in Figure 7 show that as leaf nodes increase, diversity improves, while relevance rises initially and then slightly declines. This is likely due to more leaf nodes providing ToP-Rec with greater capacity to explore unobserved preferences. However, as the number of leaf nodes increases further, the risk of introducing noise also rises. Nevertheless, due to our systematic preference reasoning design, ToP-Rec’s performance remains robust, showing substantial improvement.

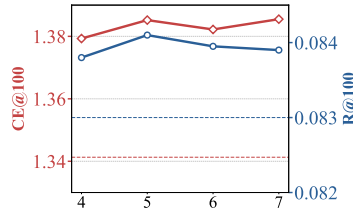


Figure 7: Impact of the number of selected leaf nodes per user. We use dashed lines to represent the performance of the backbone recommender.

136

Comparison with generative LLM-based recommendation. We also provide a comparison with the state-of-the-art LLM-based generative diversified recommender, DLCRec [16]. Since DLCRec generates recommendations by predicting item names rather than IDs, it struggles with scenarios like Twitter or Weibo, where items are posts with long text and lack explicit names. Therefore, we conducted evaluations on Amazon, where items are products with brief names. We followed DLCRec’s paradigm for LLM fine-tuning and generated 10 items per user during inference, calculating recall@10 and category-entropy@10. The results are shown in Table 7, indicating a significant improvement for our approach. We empirically found that DLCRec carries a high risk of generating homogeneous items, which can be attributed to its limited ability to fully capture

Table 7: Performance comparison with DLCRec.

	R@10	CE@10
DLCRec	0.0021	0.4714
ToP-Rec	0.0119	0.9639

users’ latent preferences, leading to poorer performance. Furthermore, due to the instability of LLM generation, DLCRec performs poorly under our original settings, where metrics are evaluated at Top 50 and 100.

A.5 Limitation and future work

While ToP-Rec shows promise in diversifying recommendations, two limitations need attention. First, the quality of user and item textual attributes affects its effectiveness. In cases of minimal or low-quality text, such as platforms with primarily video or image content, the current design may perform less effectively. Second, ToP-Rec does not focus on enhancing recommendation novelty. This limitation arises from prioritizing diversity in preference-aligned item matching without explicitly quantifying each item’s potential to surprise users. Future work will focus on integrating novelty-aware objectives, further enhancing the user experience.

A.6 Potential impacts

Our approach improves recommendation diversity by uncovering underexplored user preferences through LLMs, addressing two important societal challenges in recommendation systems. First, it reduces information redundancy by diversifying recommendations, helping users break free from repetitive suggestions and explore niche topics, which reduces the impact of algorithmic echo chambers. Second, by diversifying recommendations, users are given more opportunities to encounter content they may not have come across but are likely to be interested in, thereby fostering the diversity of societal culture. These improvements contribute to more balanced, inclusive recommendation systems that prioritize user-driven discovery over algorithmic determinism.

References

- [1] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*. 2019, pp. 4171–4186.
- [2] Guillaume Garrigos and Robert M Gower. “Handbook of convergence theorems for (stochastic) gradient methods”. In: *arXiv preprint arXiv:2301.11235* (2023).
- [3] Shangbin Feng et al. “Twibot-22: Towards graph-based twitter bot detection”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 35254–35269.
- [4] Jing Zhang et al. “Social influence locality for modeling retweeting behaviors.” In: *IJCAI*. Vol. 13. Citeseer. 2013, pp. 2761–2767.
- [5] Jianmo Ni, Jiacheng Li, and Julian McAuley. “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)*. 2019, pp. 188–197.
- [6] Jiacheng Li et al. “Text is all you need: Learning language representations for sequential recommendation”. In: *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2023, pp. 1258–1267.
- [7] Yu Zheng et al. “DGCN: Diversified recommendation with graph convolutional networks”. In: *Proceedings of the web conference 2021*. 2021, pp. 401–412.
- [8] Jaime Carbonell and Jade Goldstein. “The use of MMR, diversity-based reranking for reordering documents and producing summaries”. In: *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*. 1998, pp. 335–336.

- 190 [9] Laming Chen, Guoxin Zhang, and Eric Zhou. “Fast greedy map inference for determinantal
191 point process to improve recommendation diversity”. In: *Advances in Neural Information*
192 *Processing Systems* 31 (2018).
- 193 [10] Cheng Wu et al. “Enhancing Recommendation Accuracy and Diversity with Box Embedding:
194 A Universal Framework”. In: *Proceedings of the ACM Web Conference 2024*. 2024, pp. 3756–
195 3766.
- 196 [11] Fan Li et al. “Contextual Distillation Model for Diversified Recommendation”. In: *Proceedings*
197 *of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2024,
198 pp. 5307–5316.
- 199 [12] Jingtong Gao et al. “LLM4Rerank: LLM-based Auto-Reranking Framework for Recommen-
200 dations”. In: *Proceedings of the ACM on Web Conference 2025*. 2025, pp. 228–239.
- 201 [13] Wei Wei et al. “Llmrec: Large language models with graph augmentation for recommendation”.
202 In: *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*.
203 2024, pp. 806–815.
- 204 [14] Steffen Rendle et al. “BPR: Bayesian personalized ranking from implicit feedback”. In: *arXiv*
205 *preprint arXiv:1205.2618* (2012).
- 206 [15] Xiang Wang et al. “Neural graph collaborative filtering”. In: *Proceedings of the 42nd interna-*
207 *tional ACM SIGIR conference on Research and development in Information Retrieval*. 2019,
208 pp. 165–174.
- 209 [16] Jiaju Chen et al. “DLCRec: A Novel Approach for Managing Diversity in LLM-Based Recom-
210 mender Systems”. In: *Proceedings of the Eighteenth ACM International Conference on Web*
211 *Search and Data Mining*. 2025, pp. 857–865.