

Enhancing Graph-based Recommendations with Majority-Voting LLM-Rerank Augmentation

Minh-Anh Nguyen¹, Bao Nguyen², Ha Lan N.T.³, Tuan Anh Hoang⁴, Duc-Trong Le⁵, Dung D. Le¹,

¹VinUniversity, ²The Chinese University of Hong Kong, ³FPT University,

⁴RMIT University, ⁵VNU University of Engineering and Technology

Abstract

Recommendation systems often suffer from data sparsity, caused by limited user-item interactions, which degrades their performance and amplifies popularity bias in real-world scenarios. This paper proposes a novel data augmentation framework that leverages Large Language Models (LLMs) and item textual descriptions to enrich interaction data. By few-shot prompting LLMs multiple times to rerank items and aggregating the results via majority voting, we generate high-confidence synthetic user-item interactions, supported by theoretical guarantees based on the concentration of measure. To effectively leverage the augmented data in the context of a graph recommendation system, we integrate it into a graph contrastive learning framework to mitigate distributional shift and alleviate popularity bias. Extensive experiments show that our method improves accuracy and reduces popularity bias, outperforming strong baselines.

1 Introduction

Recommendation systems (RS), particularly graph-based recommendation models, often suffer from data sparsity, which significantly degrades performance and amplifies popularity bias (Idrissi and Zellou 2020). Traditional data augmentation techniques such as random walk sampling, node and edge dropout (Wu et al. 2021), and subgraph sampling (Fan et al. 2023a; Li et al. 2023) generate alternative views of the interaction graph for use in self-supervised learning. These methods are simple and can improve accuracy in low-interaction regimes. Still, they rely solely on interaction data and overlook the potential of other modalities, such as item textual descriptions.

Recent advances leverage the reasoning capabilities of Large Language Models (LLMs) and the rich semantic information in item descriptions to enhance user and item profiling for data augmentation (Wu et al. 2024b). While LLM-based methods show promise in enriching recommendation datasets using external knowledge, they suffer from two main limitations. First, the augmentation results generated by prompting LLMs are often inconsistent across different inference runs, leading to unstable performance. Specifically, using a commonly adopted prompt template from prior works to augment interaction data, we observe that both LLMRec (Hou et al. 2024b) and Llama4Rec (Luo et al.

Preprint. Under review.

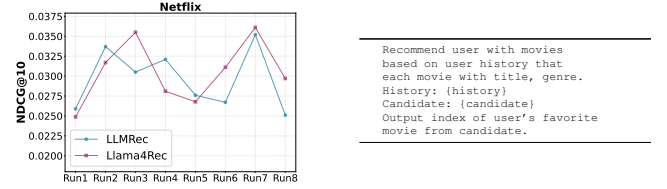


Figure 1: NDCG@10 variation across repeated LLM augmentation runs on 1,000 Netflix users using a casual prompt. Variability in augmented items across runs leads to unstable recommendation quality. Higher NDCG@10 indicates better alignment with user preferences.

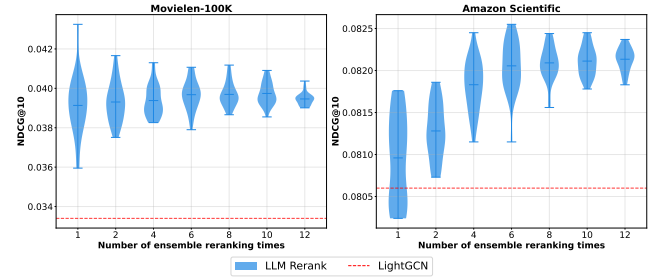


Figure 2: Distribution of NDCG@10 across different reranking times over 30 repeated runs. Increasing the ensemble size via more reranking times improves the mean and reduces variance, indicating greater stability and robustness. LLM-based reranking consistently outperforms LightGCN, and ensemble methods further improve accuracy.

2024) yield fluctuating NDCG@10 scores on the Netflix dataset, as shown in Figure 1. Second, embeddings derived from the textual content of LLM-embedded models tend to be out-of-distribution compared to collaborative embeddings, making it difficult to align them in a model-agnostic framework across various RS (Ren et al. 2024; Qiao et al. 2024).

To address the above challenges, we reformulate LLM-based data augmentation as an item reranking task, combining few-shot prompting with majority-vote ensembling via Reciprocal Rank Fusion (RRF) (Cormack, Clarke, and Buettcher 2009). This formulation enables the LLM to bet-

ter leverage its parametric knowledge of user preferences, assigning high positive scores to relevant items (Wu et al. 2024a; Hou et al. 2024c). Ensemble inference improves robustness and consistency by aggregating signals across reranking runs, as shown in Figures 2. To support this ensembling approach, we propose a theoretically grounded augmentation strategy, using concentration of measure theory to derive bounds on augmentation accuracy. Rather than relying on LLM-generated embeddings, we directly utilize the augmented data in a graph contrastive learning (CL) framework. This approach aligns representations between original and augmented graphs, mitigating distributional shift and popularity bias (Zimmermann et al. 2021), without modifying the base model architecture. We summarize our contributions as:

- We propose **VoteGCL**, a model-agnostic, end-to-end framework that employs LLMs with few-shot in-context reranking. VoteGCL uses majority-vote reranking over candidate items to perform data augmentation and further integrates graph contrastive learning to effectively leverage the augmented data.
- We provide a theoretical analysis based on the concentration of measure, demonstrating that majority-vote reranking effectively extracts the LLM’s most reliable knowledge, thereby enhancing the quality of augmented interactions.
- Extensive experiments show that VoteGCL consistently outperforms state-of-the-art graph-based and LLM-enhanced recommender models across multiple benchmarks, improving recommendation accuracy while mitigating popularity bias.

2 Related work

Graph-Augmented Recommendation. Graph-based RS represents user-item interactions as bipartite graphs, where nodes are users and items, and edges denote interactions. Early models used random walks to uncover latent relationships via transition probabilities (Nikolakopoulos and Karypis 2019). The introduction of Graph Convolutional Networks (GCNs) enabled embedding propagation better to capture collaborative signals (He et al. 2020; Wang et al. 2019). Various graph augmentation techniques have been developed to address data sparsity and over-smoothing issues in graph RS. Traditional graph augmentation methods (Ding et al. 2022) can be categorized into three approaches: (1) Edge/Node Dropout, which removes redundant connections to improve robustness (Wu et al. 2021; Zhou et al. 2023); (2) Graph Diffusion, which enriches connectivity by adding new edges based on structural properties (Fan et al. 2023b; Yang et al. 2021); and (3) Subgraph Sampling, which generates localized views for diverse training samples (Zhang et al. 2024; Xie et al. 2022). While effective, these methods focus solely on graph structure and overlook valuable side information such as item content.

LLMs-Enhanced Graph Recommendation. Recent advances explore integrating large language models (LLMs) into RS, typically along three lines: (1) Knowledge Enhancement, which uses LLMs’ reasoning capabilities to gen-

erate comprehensive user profiles and item descriptions by summarizing user behavior patterns and extracting rich semantic features from item content (Liu et al. 2024; Xi et al. 2024; Wang et al. 2024); (2) Model Enhancement, where LLM-generated textual embeddings of item descriptions and user profiles are integrated with collaborative signals through attention mechanisms or multi-modal alignment (Ren et al. 2024; Qiao et al. 2024); and (3) Interaction Enhancement employs LLMs to augment the user-item interaction space by generating synthetic interactions or directly scoring candidate items. This includes predicting missing ratings, generating counterfactual interactions to address data sparsity (Wei et al. 2024; Song, Chao, and Liu 2024). Despite their promise, LLM-based methods face challenges such as representation misalignment and inconsistent inference due to the stochastic nature of generation.

3 Preliminaries

Problem Formulation. In graph-based recommendation systems, user-item interactions are modeled as a bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \mathcal{U} \cup \mathcal{I}$ consists of user nodes $\mathcal{U}$ and item nodes $\mathcal{I}$, and $\mathcal{E}$ denotes the set of observed interactions. Each user $u \in \mathcal{U}$ and item $i \in \mathcal{I}$ is assigned a learnable embedding $\mathbf{e}_u, \mathbf{e}_i \in \mathbb{R}^d$, initialized in an embedding matrix $\mathbf{E}^{(0)} \in \mathbb{R}^{|\mathcal{V}| \times d}$ and updated during training. The model is optimized on mini-batches $\mathcal{B}$ sampled from $\mathcal{E}$. After training, we denote the resulting embedding matrix as $\mathbf{E}$, which is learned such that each user embedding $\mathbf{e}_u$ is close to the embeddings $\mathbf{e}_i$ of the items the user prefers.

Vanilla Graph-based Retrieval. We adopt LightGCN (He et al. 2020), a simplified yet effective GCN, as the primary retrieval model to propagate information over the user-item graph $\mathcal{G}$ by updating node representations through neighborhood aggregation and combination. At the l -th iteration, the node representations are updated as: $\mathbf{E}^{(l)} = H(\mathbf{E}^{(l-1)}, \mathcal{G})$, where $\mathbf{E}^{(l)}$ denotes the node representations at iteration l which corresponds to the l -th layer of graph convolution, and H is the neighborhood aggregation function. The embedding propagation process is mathematically formulated as a weighted combination of representations from different iterations. The final embeddings are computed as:

$$\mathbf{E} = \frac{1}{1+L} \left(\mathbf{E}^{(0)} + \tilde{\mathbf{A}}\mathbf{E}^{(0)} + \dots + \tilde{\mathbf{A}}^L\mathbf{E}^{(0)} \right), \quad (1)$$

where $\mathbf{E}$ denotes the final representations used for downstream prediction tasks, L corresponds to the number of propagation layers, and $\tilde{\mathbf{A}} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ is the normalized undirected adjacency matrix without self-connections. This iterative process allows the model to capture higher-order connectivity patterns in the graph. A common approach is to train the graph-based recommendation model using the Bayesian Personalized Ranking (BPR) (Rendle et al. 2012) loss. The model learns to assign higher scores to observed interactions than to unobserved ones. The predicted score for user u and item i is computed via inner product: $\hat{y}_{ui} = \mathbf{e}_u^\top \mathbf{e}_i$. The BPR loss is defined as:

$$\mathcal{L}_{\text{BPR}} = \sum_{(u,i,j) \in \mathcal{O}} -\log \sigma(\hat{y}_{ui} - \hat{y}_{uj}), \quad (2)$$

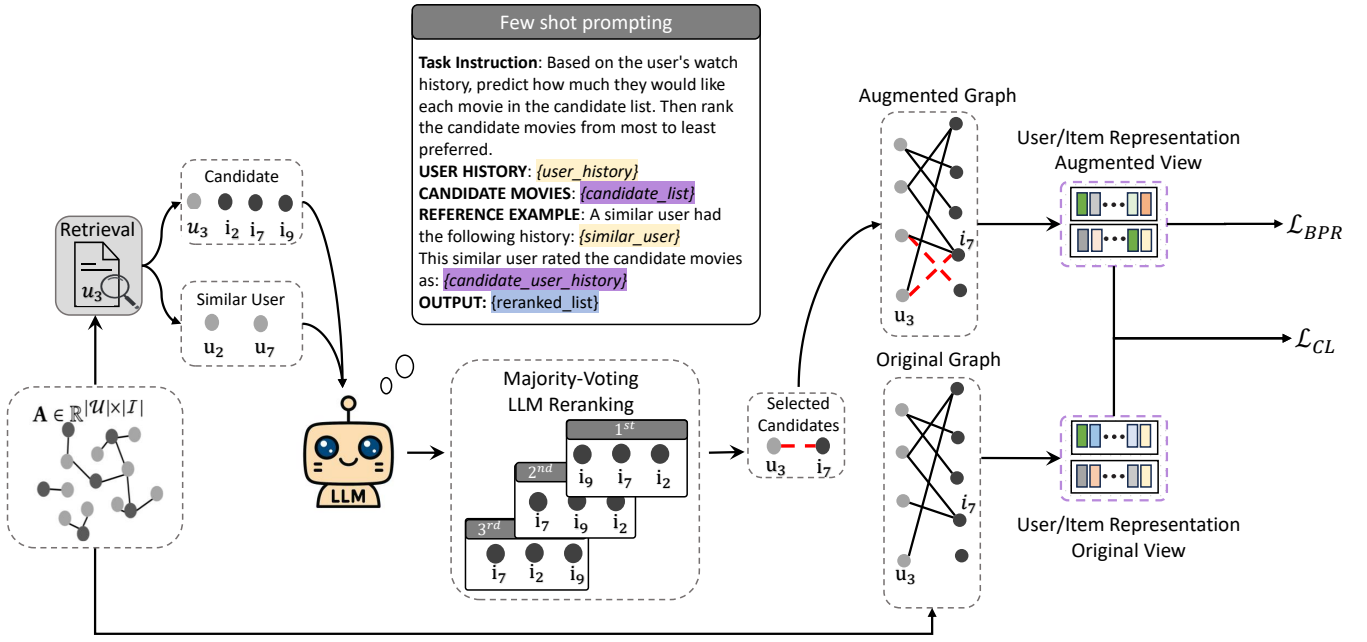


Figure 3: **Overview of the VoteGCL.** The framework starts with a retrieval model that generates candidate items and identifies similar users for low-degree users using collaborative signals. An LLM reranks these candidates, and majority voting selects top-K high-confidence interactions for augmentation. The augmented data is then integrated into a graph contrastive learning framework, where two contrastive views are created via separate graph convolutions on a shared embedding matrix.

where $O = \{(u, i, j) \mid (u, i) \in \mathcal{E}, (u, j) \in \mathcal{R}^-\}$ denotes the pairwise training data, $\mathcal{R}^-$ is the unobserved interactions and $\sigma(\cdot)$ is the sigmoid function. In inference, items closest to $\mathbf{e}_u$ in embedding space are recommended.

4 Methodology

We propose a framework **VoteGCL**, as illustrated in Figure 3, we use few-shot prompting LLM for inference via majority-voted reranking to construct the augmented dataset as detailed in Section 4.1. To effectively leverage the augmented data for graph recommendation, we employ the CL paradigm as detailed in Section 4.2.

4.1 Data Augmentation via Majority Reranking

This section presents a targeted data augmentation strategy aimed at enriching the training graph with high-quality synthetic interactions. Augmenting artificial interactions for high-degree users can introduce unnecessary cost and noisy interactions. Therefore, we apply the augmentation only to low-degree users, defined as those whose number of interactions falls below the 25th percentile of the user interaction distribution. This strategy ensures that edges are augmented only for users with insufficient interaction data. An ablation study analyzing the effect of this threshold is provided in Section 5.3.

For each user u in the selected set, we aim to select p new interactions that do not appear in the original dataset. While LLMs have shown promising capabilities in ranking and recommendation tasks, they are limited by context length and thus cannot consider all potential items and re-

lated users within a single prompt. Hence, we employ LightGCN as a conventional retrieval model to obtain an ordered shortlist of K non-interacted candidate items, denoted as $\mathbf{I}_{\text{can}} = \{i_k\}_{k=1}^K$, along with a set of similar users based on collaborative embeddings generated by the retrieval model, inspired by Wu et al.; Wei et al.. Based on this information, we employ an LLM to rerank the sequence $\mathbf{I}_{\text{can}}$, producing a sorted list denoted as $\mathbf{I}_{\text{sorted}}$. To perform the reranking, we design a generalizable few-shot prompting template consisting of four components: first, a task-specific instruction; second, the user’s watch history enriched with flattened metadata; third, a structured output format; and fourth, one or more few-shot examples drawn from similar users. Each example includes the user’s history along with a ground-truth, reranked list sorted by rating. This prompt is dataset-agnostic and can be easily adapted to new domains by reformatting the metadata. An example prompt for the movie dataset is provided in Figure 3, with further details available in Appendix C.

To improve the consistency of reranking outcomes and better leverage the LLM’s reasoning capabilities across multiple trials, we perform N independent rerankings of the candidate list $\mathbf{I}_{\text{can}}$. The resulting sequences are then aggregated using a simplified variant of RRF. For each item $i_k \in \mathbf{I}_{\text{can}}$, we compute an aggregated score as follows:

$$g(i_k) = \sum_{n=1}^N \frac{1}{\varsigma^{(n)}(i_k) + 1} \quad (3)$$

where $\varsigma^{(n)}(i_k)$ denotes the position of item i_k in the n -th reranked sequence. We select the top p items with the

highest scores to construct new artificial interactions for the user u . Following this data augmentation strategy, we theoretically prove that for any bounded, strictly decreasing function $g(\cdot)$, increasing the number of independent rerankings N asymptotically reveals the LLM’s optimal parametric knowledge for the reranking task. The formal statement is provided in Theorem 1, and a detailed proof can be found in Appendix D.

Theorem 1 (Concentration for Score Aggregation over Random Permutations). *Let $\mathcal{I} = \{i_1, i_2, \dots, i_K\}$ be a set of K items, and let $\mathcal{T}_K$ denote all permutations of $\{1, 2, \dots, K\}$. Let P be any probability distribution over $\mathcal{T}_K$, and let $\varsigma^{(1)}, \dots, \varsigma^{(N)}$ be independent random permutations sampled from P . Let $g : \mathbb{R} \rightarrow \mathbb{R}$ be a bounded, strictly decreasing function. Suppose $A, B \in \mathbb{R}$ are such that $A \leq g(x) \leq B$ for all x . For each item $i \in \mathcal{I}$, define the aggregated score*

$$S(i) = \sum_{n=1}^N g(\varsigma^{(n)}(i)),$$

where $\varsigma^{(n)}(i)$ is the rank assigned to item i by permutation $\varsigma^{(n)}$. Let consider two distinct items $i_j, i_k \in \mathcal{I}$, and $\mu = \mathbb{E}_{\varsigma \sim P} [\varsigma(i_j) - \varsigma(i_k)]$. If $\mu > 0$, then

$$\Pr(S(i_j) > S(i_k)) \leq \exp\left(-\frac{N\mu^2}{2(B-A)^2}\right).$$

We instantiate $g(\cdot)$ as in Equation 3 for its simplicity and prominence in prior work. More importantly, given a finite value of K , the $g(\cdot)$ function is strictly decreasing and bounded, satisfying the requirements of Theorem 1. The corollary below specializes in the result for this choice, and its proof is presented in Appendix D.

Corollary 1 (Reciprocal Rank Aggregation). *Let $g(x) = 1/(x+1)$ and define $S(i) = \sum_{n=1}^N 1/(\varsigma^{(n)}(i)+1)$ for each $i \in \mathcal{I}$. For any $i_j, i_k \in \mathcal{I}$, let $\mu = \mathbb{E}_{\varsigma \sim P} [\varsigma(i_j) - \varsigma(i_k)]$. If $\mu > 0$, then*

$$\Pr(S(i_j) > S(i_k)) \leq \exp\left(-\frac{N\mu^2}{2\left(1 - \frac{1}{K+1}\right)^2}\right).$$

The probability distribution P appearing in both the Theorem and Corollary is genuinely the LLM-induced distribution over permutations. Theorem 1 and Corollary 1 show that when item i_k is expected to be ranked higher than item i_j , meaning it has a lower average rank and $\mu > 0$, the probability that i_j receives a higher aggregated score than i_k decreases exponentially with the number of permutations N . As $N \rightarrow \infty$, this error probability vanishes, indicating that score aggregation becomes increasingly reliable. Therefore, selecting the top p items with the highest aggregated scores after enough rerankings recovers the optimal set of p items under the LLM’s ideal parametric ranking.

We repeat the process for all targeted users, leading to $Q_\alpha \times |\mathcal{V}| \times p$ newly augmented edges. After data augmentation, we obtain an enriched graph $\mathcal{G}_{\text{aug}} = (\mathcal{V}, \mathcal{E}^+)$, where the original edge set $\mathcal{E}$ is augmented with newly generated interactions, thereby expanding $\mathcal{E}^+ = \mathcal{E} \cup \mathcal{E}_{\text{new}}$ while preserving the bipartite structure between users $\mathcal{U}$ and items $\mathcal{I}$.

Method	Virtual Interaction	Few-shot Prompting	Textual Embed.	Consistent Aug. Data
VoteGCL	✓	✓	×	✓
LLMRec (Wei et al. 2024)	✓	×	✓	×
RLMRec (Ren et al. 2024)	×	×	✓	×
KAR (Xi et al. 2024)	×	×	✓	×

Table 1: Comparison of LLM augmented data methods

Unlike other LLM-augmented data approaches summarized in Table 1, VoteGCL does not incur the cost of generating textual embeddings for user/item profiling or adapting high-dimensional textual representations into recommendation models. Instead, VoteGCL focuses on generating high-quality interaction data and effectively leveraging it through a Graph CL paradigm, as detailed in Section 4.2. The key advantages of our LLM-augmented data are twofold: (1) a majority-vote reranking mechanism, which ensures more consistent and high-confidence user-item interactions, with detailed theoretical justification provided in Appendix D; and (2) a few-shot prompting strategy that incorporates collaborative signals from similar users, enabling better adaptation to user-specific preferences in the data augmentation task.

4.2 Graph Contrastive Learning Paradigm

We continue to employ LightGCN as our graph encoder due to its efficiency and effectiveness, though our framework is compatible with other graph backbones (e.g., NGCF (Wang et al. 2019), XSimGCL (Yu et al. 2023)). Inspired by SimCLR (Chen et al. 2020), we employ a CL paradigm between two representation views derived from the original and augmented data. This approach mitigates the distributional shift caused by potential noise introduced during graph augmentation. We share a common embedding matrix $\mathbf{E}^{(0)}$ to encode both the original graph $\mathcal{G}$ and its augmented version $\mathcal{G}_{\text{aug}}$. During each training epoch, node representations are generated as follows:

$$\mathbf{E}^{(l)} = H(\mathbf{E}^{(l-1)}, \mathcal{G}_{\text{aug}}), \quad \mathbf{E}_{\text{org}}^{(l)} = H(\mathbf{E}^{(l-1)}, \mathcal{G})$$

These embeddings serve as two distinct views of representation learning for contrastive supervision. The contrastive loss is then applied by treating embeddings of the same node across different views as positive pairs, and all others as negative:

$$\mathcal{L}_{\text{cl}} = \sum_{i \in \mathcal{B}} -\log \frac{\exp(\mathbf{z}_i^\top \mathbf{z}_{i, \text{org}} / \tau)}{\sum_{j \in \mathcal{B}} \exp(\mathbf{z}_i^\top \mathbf{z}_{j, \text{org}} / \tau)}, \quad (4)$$

Here, i, j represent user (item) nodes within a sampled mini-batch $\mathcal{B}$, and $\mathbf{z}, \mathbf{z}_{\text{org}}$ are the L_2 -normalized d -dimensional representations obtained from the augmented graph $\mathcal{G}_{\text{aug}}$ and original graph $\mathcal{G}$. Note that, $\mathbf{z}_i = \frac{\mathbf{e}_i}{\|\mathbf{e}_i\|_2}$. By replacing A with the adjacency matrices of graph $\mathcal{G}_{\text{aug}}$, $\mathbf{z}$ can be learned via Equation 1. The temperature parameter $\tau > 0$ (e.g., $\tau = 0.2$)

controls the sharpness of the similarity distribution. The CL loss can also be expressed as:

$$\mathcal{L}_{cl} = \sum_{i \in \mathcal{B}} \left(-\mathbf{z}_i^\top \mathbf{z}_{i,org} / \tau + \log \left(\sum_{j \in \mathcal{B} / \{i\}} \exp(\mathbf{z}_i^\top \mathbf{z}_{j,org} / \tau) \right) \right) \quad (5)$$

Thus, $\mathcal{L}_{cl}$ enhances uniformity in the embedding space through its first component and promotes diversity through the second component, thereby improving model accuracy and mitigating popularity bias (Zimmermann et al. 2021; Wang and Isola 2020). The final node embeddings used for recommendation are obtained from the last propagation layer of the augmented graph. Specifically, we adopt $\mathbf{E}^{(L)}$, where L corresponds to the total number of propagation layers, as the learned representation from the augmented graph $\mathcal{G}_{aug}$. This embedding captures the high-order structural information introduced by the augmentation and is subsequently employed to compute the BPR loss for optimizing recommendation performance. Finally, the overall training objective combines the recommendation loss with the CL loss:

$$\mathcal{L}_{main} = \mathcal{L}_{BPR} + \lambda \mathcal{L}_{cl} \quad (6)$$

where $\lambda \in (0, 1)$ is a hyperparameter to control the weight of CL loss. The main advantages of this CL paradigm are that it leverages LLMs purely for knowledge-guided augmentation without embedding integration, thereby avoiding high-dimensional representations and complex fusion with collaborative signals. Additionally, VoteGCL shares the same learnable embedding matrix for both the original and augmented graphs, enabling more efficient training with reasonable model complexity. A detailed analysis of model complexity is provided in Appendix B.

5 Experiments

5.1 Experimental Setup

Datasets. We evaluate VoteGCL on four real-world datasets from diverse domains: Amazon (Scientific and Book) (Hou et al. 2024a), MovieLens (Harper and Konstan 2015), Netflix (Wei et al. 2024), and Yelp2018 (Wang et al. 2019). Dataset statistics after preprocessing are summarized in Table 5. By default, we select the top-1 result from majority-vote LLM reranking as virtual edges for data augmentation.

Metrics. We use NDCG@K and Recall@K to evaluate alignment between recommended items and user preferences, and APLT@K (Abdollahpouri, Burke, and Mobasher 2019) to measure the average percentage of long-tail items among top-K recommendations as an indicator of popularity bias. Following the leave-one-out strategy, we use the latest interaction for testing, the second latest for validation, and the remaining for training. LightGCN generates item candidates for LLM reranking using *gpt-3.5-turbo* (Achiam et al. 2023) as prior work.

Baselines. We compare VoteGCL against three categories: (1) base graph-based recommendation systems including LightGCN (He et al. 2020), NGCF (Wang et al. 2019), and MF-BPR (Koren, Bell, and Volinsky 2009); (2) traditional augmented graph recommendation methods such

as SGL (Zhou et al. 2023), GraphDA (Fan et al. 2023b), GraphAug (Zhang et al. 2024), and XSimGCL (Yu et al. 2022); and (3) LLM-enhanced recommendation systems including KAR (Xi et al. 2024), RLMRec (Ren et al. 2024), and LLMRec (Wei et al. 2024). We provide details of baselines in Appendix A.

5.2 Main Results

Traditional graph recommendation. Table 2 demonstrates that the proposed VoteGCL framework exhibits strong model-agnostic capabilities, consistently improving performance across diverse graph-based recommendation architectures, including LightGCN, XSimGCL, and NGCF. VoteGCL achieves substantial improvements in recommendation accuracy, with LightGCN+VoteGCL showing the most significant gains of 13.6% on MovieLens 1M, and 14.6% on Yelp2018 for R@10 metrics. Notably, the framework demonstrates exceptional scalability, delivering consistent improvements across both large-scale datasets (Yelp2018 and MovieLens 1M) and smaller datasets (Amazon Scientific and MovieLens 100K), with LightGCN+VoteGCL achieving particularly strong performance gains of 43.6% on MovieLens 100K and 16.4% on Amazon Scientific for NDCG@10, demonstrating superior adaptation to user preferences in reranking scenarios. More importantly, the framework effectively addresses popularity bias by dramatically enhancing long-tail item recommendations, as evidenced by APLT improvements ranging from 25.5% to 818% across different dataset-model combinations. These results demonstrate that VoteGCL successfully mitigates popularity bias while maintaining recommendation accuracy, promoting diversity by enabling models to learn better representations for underrepresented items regardless of dataset scale. The consistent performance gains across four distinct datasets spanning different domains and varying sizes validate the robust generalizability and universal applicability of VoteGCL.

LLM-enhanced graph recommendation system. As shown in Figure 4, LightGCN + VoteGCL consistently outperforms all other methods across three benchmark datasets (Netflix, Amazon Book, and Yelp2018) under both @10 and @20 evaluation settings, achieving superior performance in terms of both accuracy and diversity. This includes outperforming traditional baselines (LightGCN, SGL) as well as LLM-enhanced models. While LLMRec, RLMRec, and KAR demonstrate notable improvements in ranking metrics such as NDCG, particularly when textual embeddings from LLMs are incorporated into the recommendation backbone, they also introduce a higher degree of popularity bias. This drawback is reflected in their lower APLT scores compared to SGL, which applies conventional data augmentation without relying on LLM-generated semantics. In contrast, VoteGCL alleviates this issue by generating high-confidence synthetic interactions through LLM prompting and integrating them effectively via graph contrastive learning, resulting in a more balanced trade-off between recommendation accuracy and fairness.

Dataset	Metric	MF-BPR	Light GCN	NGCF	Graph Aug	Graph DA	SGL	XSimGCL	NGCF+ VoteGCL	XSimGCL+ VoteGCL	LightGCN+ VoteGCL	Improv.
Movielen 100K	R@10	0.0750	0.0766	0.0681	0.0689	0.0528	0.0818	<u>0.0852</u>	0.0691 ^{0.3%}	0.0852 ^{0.0%}	0.0886 ^{15.7%}	4.0%
	N@10	0.0335	0.0334	0.0308	0.0334	0.0251	<u>0.0407</u>	0.0398	0.0372 ^{8.1%}	0.0411 ^{3.3%}	0.0494 ^{47.9%}	21.4%
	A@10	0.0042	0.0097	0.0019	0.0016	0.0463	0.0227	<u>0.0583</u>	0.0166 ^{733%}	0.1852 ^{217%}	0.0617 ^{536%}	217.7%
Movielen 1M	R@10	0.0712	0.0817	0.0628	0.0643	0.0799	0.0812	<u>0.0857</u>	0.0659 ^{4.9%}	0.0901 ^{5.1%}	0.0928 ^{13.6%}	8.3%
	N@10	0.0365	0.0403	0.0311	0.0268	0.0385	0.0402	<u>0.0416</u>	0.0375 ^{20.6%}	0.0428 ^{2.9%}	0.0462 ^{14.6%}	11.1%
	A@10	0.0015	0.0027	0.0011	0.0056	<u>0.0104</u>	0.0028	0.0098	0.0061 ^{454%}	0.0123 ^{25.5%}	0.0145 ^{437%}	39.4%
Yelp2018	R@10	0.0304	0.0425	0.0375	0.0352	0.0422	<u>0.0446</u>	0.0435	0.0401 ^{6.9%}	0.0442 ^{1.6%}	0.0487 ^{14.6%}	8.8%
	N@10	0.0147	0.0214	<u>0.0229</u>	0.0125	0.0230	0.0223	0.0221	0.0201 ^{0.0%}	0.0234 ^{10.9%}	0.0247 ^{15.4%}	7.9%
	A@10	0.0075	0.0038	0.0014	0.0012	0.0032	<u>0.0096</u>	0.0068	0.0018 ^{28.6%}	0.0210 ^{208%}	0.0349 ^{818%}	263.5%
Amazon Scientific	R@10	0.1173	0.1251	0.0941	0.0986	0.1204	<u>0.1297</u>	0.1291	0.1317 ^{39.9%}	0.1335 ^{3.4%}	0.1385 ^{10.7%}	6.8%
	N@10	0.0850	0.0806	0.0648	0.0646	0.0758	<u>0.0911</u>	0.0896	0.1008 ^{48.0%}	0.0936 ^{4.5%}	0.0993 ^{23.2%}	10.6%
	A@10	<u>0.2563</u>	0.2015	0.1839	0.0413	0.2034	0.2504	0.1779	0.2542 ^{38.2%}	0.1975 ^{11.0%}	0.2795 ^{38.7%}	9.1%

Table 2: Performance Comparison of Vanilla Graph Recommendation Methods on Multiple Datasets Before and After Data Augmentation. Each value in augmented data is shown as value^{improvement}, where the superscript indicates the percentage improvement over the baseline. Underlined values highlight the best baseline performance, while bold values highlight the best performance after data augmentation. "Improv" refers to the absolute improvement between these two.

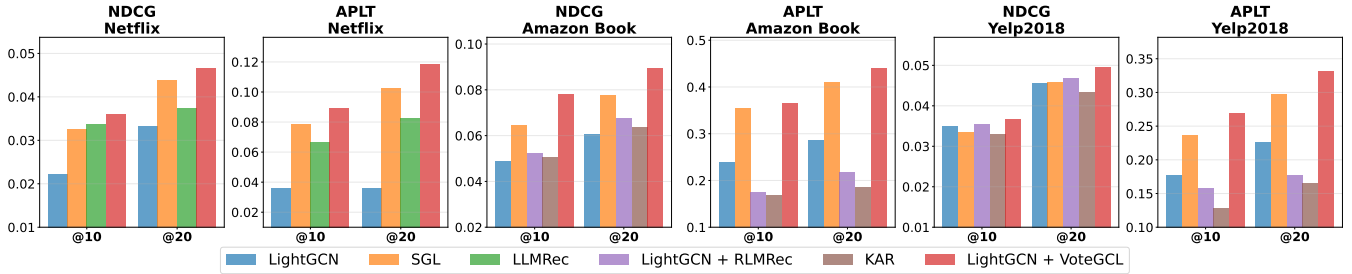


Figure 4: Performance of LLM-enhanced graph recommenders on three datasets under Top K settings (K = 10 and 20), using dataset-specific models and leveraging available LLM-augmented data for fair comparison.

Dataset	Metric	Quantile of users to be augmented			
		25%	50%	75%	100%
MovieLens-100K	N@10	0.0494	0.0482	0.0501	0.0503
	A@10	0.0617	0.0562	0.0574	0.0596
Amazon Scientific	N@10	0.0993	0.0983	0.0989	0.0991
	A@10	0.2795	0.2939	0.2893	0.2792
Netflix	N@10	0.0351	0.0336	0.0349	0.0352
	A@10	0.0892	0.1089	0.0995	0.0919

Table 3: Performance under different ratios of users selected for augmentation

5.3 Ablation Studies

Number of Virtual Edges for Augmentation. Table 3 examines the impact of augmenting different user quantiles on VoteGCL’s performance across three datasets. The results show that expanding augmentation to a larger portion of users does not consistently improve performance in either NDCG@10 or APLT@10. In some cases, performance even declines, suggesting that indiscriminate augmentation can introduce noise that offsets the benefits of additional interactions. These findings highlight the importance of targeted

augmentation, specifically, focusing on the bottom 25% of users with the fewest interactions. This approach improves generalization for cold-start users while maintaining overall model robustness and computational efficiency. Alternatively, the trade-off between recommendation accuracy and diversity can be more effectively managed by tuning the contrastive loss weight λ , allowing finer control over model behavior without increasing augmentation overhead.

Zero-shot vs Few-shot Prompting LLM with Different Number of Majority-voting.

Figure 5 illustrates the impact of the prompting strategy and the number of reranking rounds on the NDCG@10 performance of LLM-based item reranking. Across both datasets, few-shot prompting consistently outperforms zero-shot prompting, highlighting its effectiveness in capturing user preferences through collaborative signals. As the number of majority-voting rounds increases, performance improves and becomes more stable, particularly from 8 rounds onward, for both prompting strategies. Notably, the few-shot setting yields higher mean scores across 30 trials, indicating greater reliability and robustness in reranking outcomes. These findings suggest that

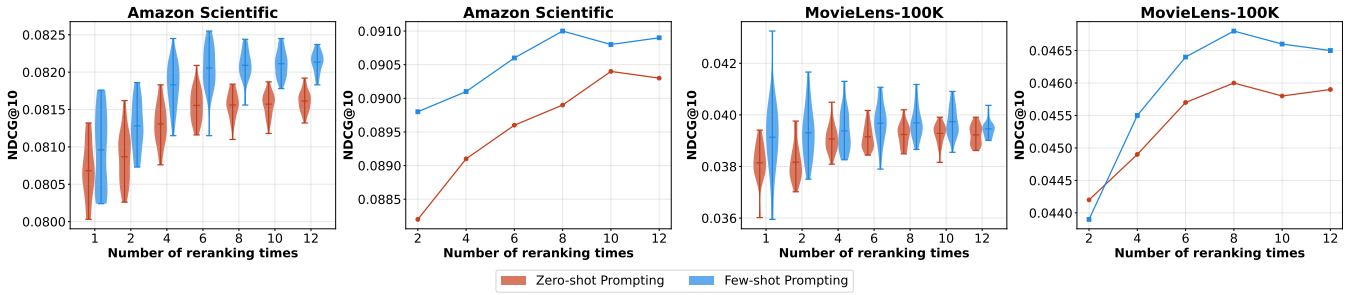


Figure 5: Comparison of reranking performance between zero-shot and few-shot prompting across different numbers of majority-voting rounds. Results are averaged over 30 runs on the Amazon Scientific and MovieLens-100K datasets.

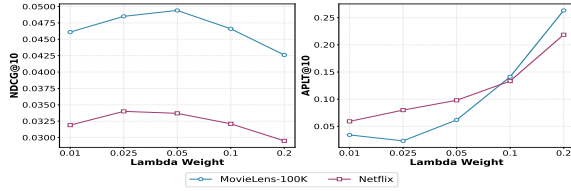


Figure 6: Performance of VoteGCL under different contrastive weights

effectively leveraging the LLM’s parametric knowledge via few-shot prompting leads to higher-quality augmented data and ultimately enhances the performance of VoteGCL. The performance at 6 voting rounds is competitive, making it a practical and cost-effective choice in resource-constrained scenarios. From these observations, we recommend combining few-shot prompting with a sufficiently large number of majority-voting rounds (e.g., ≥ 6) to fully realize the benefits of LLM-based reranking in data augmentation.

Controlling Hyperparam λ Contrastive Weight. As illustrated in Figure 6, the contrastive loss weight $\lambda \in (0, 1)$ plays a critical role in balancing recommendation accuracy and popularity bias mitigation. When λ increases, the APLT@10 metric improves significantly, indicating greater inclusion of long-tail items and enhanced diversity in recommendations. This effect is consistent across both MovieLens-100K and Netflix datasets. However, this gain in diversity comes with a trade-off: NDCG@10 gradually declines as λ grows, reflecting a reduction in ranking precision. The results suggest that overly high values of λ (e.g., 0.2) may overly prioritize long-tail exploration at the expense of accuracy. Based on these observations, we recommend tuning λ within the range of 0.01 to 0.1 to strike an effective balance between accuracy and fairness, depending on the characteristics of the target dataset.

Study of VoteGCL. Table 4 presents an ablation study evaluating the impact of key components in VoteGCL across three datasets. The full VoteGCL model consistently outperforms all baselines in both NDCG@10 and APLT@10. When the graph CL module is removed (“w/o GCL”), performance drops across all datasets, though it still outperforms LightGCN in certain metrics. This suggests that while

Methods	MovieLens-100K		Netflix		Amazon Book	
	N@10	A@10	N@10	A@10	N@10	A@10
VoteGCL	0.0494	0.0617	0.0351	0.0892	0.0781	0.3654
LightGCN	0.0344	0.0097	0.0221	0.0358	0.0486	0.2390
w/o GCL	0.0382	0.0102	0.0245	0.0341	0.0502	0.2511
w/o LLM	0.0201	0.0068	0.0115	0.0126	0.0274	0.1243

Table 4: Ablation Study on VoteGCL Components

the LLM-based augmentation is beneficial, without GCL, the model becomes more sensitive to noise from the synthetic data. Additionally, removing the LLM reranking with majority voting (“w/o LLM augment.”) and selecting top-1 candidates without using LLM to vote leads to a significant performance collapse. This highlights the importance of majority voting in stabilizing the augmentation and leveraging textual descriptions modalities, making both GCL and LLM-based reranking essential for the overall robustness and effectiveness of VoteGCL.

Additional Experimental Results. Additional ablation studies are provided in Appendix A. These include evaluations of different retrieval models for generating candidate items for LLM reranking, assessments of alternative LLMs for data augmentation (e.g., Qwen2.5 (Yang et al. 2024)), and an analysis of how varying the number of top-K items in the majority-voting process affects performance. Furthermore, we compare the cost of generating augmented data using VoteGCL in terms of price per accuracy gain and total generation cost, demonstrating that our approach provides a more cost-efficient alternative to other LLM-based methods.

6 Conclusion

We propose **VoteGCL**, a model-agnostic data augmentation framework that tackles inconsistency and distributional mismatch in LLM-enhanced recommendation. VoteGCL uses LLM-based reranking with majority voting to generate reliable synthetic interactions and applies contrastive learning to align original and augmented graphs without altering model architecture. Experiments across benchmarks show VoteGCL improves recommendation accuracy and reduces popularity bias, outperforming traditional and LLM-based data augmentation methods for a graph-based recommender.

References

- Abdollahpour, H.; Burke, R.; and Mobasher, B. 2019. Managing popularity bias in recommender systems with personalized re-ranking. *arXiv preprint arXiv:1901.07555*.
- Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.
- Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, 1597–1607. PmLR.
- Cormack, G. V.; Clarke, C. L.; and Buettcher, S. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, 758–759.
- Ding, K.; Xu, Z.; Tong, H.; and Liu, H. 2022. Data augmentation for deep graph learning: A survey. *ACM SIGKDD Explorations Newsletter*, 24(2): 61–77.
- Fan, Z.; Xu, K.; Dong, Z.; Peng, H.; Zhang, J.; and Yu, P. S. 2023a. Graph collaborative signals denoising and augmentation for recommendation. In *Proceedings of the 46th international ACM SIGIR conference on research and development in information retrieval*, 2037–2041.
- Fan, Z.; Xu, K.; Dong, Z.; Peng, H.; Zhang, J.; and Yu, P. S. 2023b. Graph Collaborative Signals Denoising and Augmentation for Recommendation. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR 2023.
- Harper, F. M.; and Konstan, J. A. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4): 1–19.
- He, X.; Deng, K.; Wang, X.; Li, Y.; Zhang, Y.; and Wang, M. 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.
- Hou, Y.; Li, J.; He, Z.; Yan, A.; Chen, X.; and McAuley, J. 2024a. Bridging Language and Items for Retrieval and Recommendation. *arXiv preprint arXiv:2403.03952*.
- Hou, Y.; Zhang, J.; Lin, Z.; Lu, H.; Xie, R.; McAuley, J.; and Zhao, W. X. 2024b. Large language models are zero-shot rankers for recommender systems. In *European Conference on Information Retrieval*, 364–381. Springer.
- Hou, Y.; Zhang, J.; Lin, Z.; Lu, H.; Xie, R.; McAuley, J.; and Zhao, W. X. 2024c. Large language models are zero-shot rankers for recommender systems. In *European Conference on Information Retrieval*, 364–381. Springer.
- Idrissi, N.; and Zellou, A. 2020. A systematic literature review of sparsity issues in recommender systems. *Social Network Analysis and Mining*, 10(1): 15.
- Koren, Y.; Bell, R.; and Volinsky, C. 2009. Matrix factorization techniques for recommender systems. *Computer*, 42(8): 30–37.
- Li, C.; Xia, L.; Ren, X.; Ye, Y.; Xu, Y.; and Huang, C. 2023. Graph transformer for recommendation. In *Proceedings of the 46th international ACM SIGIR conference on research and development in information retrieval*, 1680–1689.
- Liu, Q.; Chen, N.; Sakai, T.; and Wu, X.-M. 2024. Once: Boosting content-based recommendation with both open- and closed-source large language models. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, 452–461.
- Luo, S.; Yao, Y.; He, B.; Huang, Y.; Zhou, A.; Zhang, X.; Xiao, Y.; Zhan, M.; and Song, L. 2024. Integrating large language models into recommendation via mutual augmentation and adaptive aggregation. *arXiv preprint arXiv:2401.13870*.
- Nikolakopoulos, A. N.; and Karypis, G. 2019. Recwalk: Nearly uncoupled random walks for top-n recommendation. In *Proceedings of the twelfth ACM international conference on web search and data mining*, 150–158.
- Qiao, S.; Gao, C.; Wen, J.; Zhou, W.; Luo, Q.; Chen, P.; and Li, Y. 2024. LLM4SBR: A lightweight and effective framework for integrating large language models in session-based recommendation. *arXiv preprint arXiv:2402.13840*.
- Ren, X.; Wei, W.; Xia, L.; Su, L.; Cheng, S.; Wang, J.; Yin, D.; and Huang, C. 2024. Representation learning with large language models for recommendation. In *Proceedings of the ACM Web Conference 2024*, 3464–3475.
- Rendle, S.; Freudenthaler, C.; Gantner, Z.; and Schmidt-Thieme, L. 2012. BPR: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*.
- Song, T.; Chao, W.; and Liu, H. 2024. Large language model enhanced hard sample identification for denoising recommendation. *arXiv preprint arXiv:2409.10343*.
- Wang, T.; and Isola, P. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International conference on machine learning*, 9929–9939. PMLR.
- Wang, X.; He, X.; Wang, M.; Feng, F.; and Chua, T.-S. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*, 165–174.
- Wang, Y.; Chu, Z.; Ouyang, X.; Wang, S.; Hao, H.; Shen, Y.; Gu, J.; Xue, S.; Zhang, J.; Cui, Q.; et al. 2024. Llmrg: Improving recommendations through large language model reasoning graphs. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, 19189–19196.
- Wei, W.; Ren, X.; Tang, J.; Wang, Q.; Su, L.; Cheng, S.; Wang, J.; Yin, D.; and Huang, C. 2024. Llmrec: Large language models with graph augmentation for recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, 806–815.
- Wu, J.; Chang, C.-C.; Yu, T.; He, Z.; Wang, J.; Hou, Y.; and McAuley, J. 2024a. Coral: collaborative retrieval-augmented large language models improve long-tail recommendation. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 3391–3401.

Wu, J.; Wang, X.; Feng, F.; He, X.; Chen, L.; Lian, J.; and Xie, X. 2021. Self-supervised graph learning for recommendation. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval*, 726–735.

Wu, L.; Zheng, Z.; Qiu, Z.; Wang, H.; Gu, H.; Shen, T.; Qin, C.; Zhu, C.; Zhu, H.; Liu, Q.; et al. 2024b. A survey on large language models for recommendation. *World Wide Web*, 27(5): 60.

Xi, Y.; Liu, W.; Lin, J.; Cai, X.; Zhu, H.; Zhu, J.; Chen, B.; Tang, R.; Zhang, W.; and Yu, Y. 2024. Towards open-world recommendation with knowledge augmentation from large language models. In *Proceedings of the 18th ACM Conference on Recommender Systems*, 12–22.

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

Yang, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; Yu, B.; Li, C.; Liu, D.; Huang, F.; Wei, H.; Lin, H.; Yang, J.; Tu, J.; Zhang, J.; Yang, J.; Yang, J.; Zhou, J.; Lin, J.; Dang, K.; Lu, K.; Bao, K.; Yang, K.; Yu, L.; Li, M.; Xue, M.; Zhang, P.; Zhu, Q.; Men, R.; Lin, R.; Li, T.; Xia, T.; Ren, X.; Ren, X.; Fan, Y.; Su, Y.; Zhang, Y.; Wan, Y.; Liu, Y.; Cui, Z.; Zhang, Z.; and Qiu, Z. 2024. Qwen2.5 Technical Report. *arXiv preprint arXiv:2412.15115*.

Yang, Y.; Wu, L.; Hong, R.; Zhang, K.; and Wang, M. 2021. Enhanced Graph Learning for Collaborative Filtering via Mutual Information Maximization. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 71–80.

Yu, J.; Xia, X.; Chen, T.; Cui, L.; Hung, N. Q. V.; and Yin, H. 2023. XSimGCL: Towards extremely simple graph contrastive learning for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 36(2): 913–926.

Yu, J.; Yin, H.; Xia, X.; Chen, T.; Cui, L.; and Nguyen, Q. V. H. 2022. Are graph augmentations necessary? simple graph contrastive learning for recommendation. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*, 1294–1303.

Zhang, Q.; Xia, L.; Cai, X.; Yiu, S.; Huang, C.; and Jensen, C. S. 2024. Graph Augmentation for Recommendation. *arXiv preprint arXiv:2403.16656*.

Zhou, X.; Sun, A.; Liu, Y.; Zhang, J.; and Miao, C. 2023. Selfcf: A simple framework for self-supervised collaborative filtering. *ACM Transactions on Recommender Systems*, 1(2): 1–25.

Zimmermann, R. S.; Sharma, Y.; Schneider, S.; Bethge, M.; and Brendel, W. 2021. Contrastive learning inverts the data generating process. In *International conference on machine learning*, 12979–12990. PMLR.

A More experimental results

A.1 Experiment Setting

We conduct experiments on several real-world datasets, with dataset statistics summarized in Table 5. All models are trained for 100 epochs using 256-dimensional embeddings and a learning rate of $1e^{-3}$. Baseline hyperparameters are adopted from their original implementations. We employ LightGCN as the retrieval model to generate item candidates for LLM reranking. For augmentation, we follow prior work and use *gpt-3.5-turbo* (Achiam et al. 2023). Each graph-based recommendation model is configured with two layers. For VoteGCL, we evaluate contrastive weights in the range (0.05, 0.1), with values specifically tuned for each dataset. All experiments are executed on a single NVIDIA A5000 GPU.

Datasets	Users	Items	Inters.	Density
Amazon Scientific	10,995	5,314	76,091	1.3×10^{-3}
Amazon Book	11,000	9,332	212,484	2.1×10^{-4}
Movielens-100K	610	9,701	100,836	1.7×10^{-2}
Movielens-1M	6000	25,317	914,108	6.2×10^{-3}
Yelp2018	11,091	86,646	544,672	5.3×10^{-4}
Netflix	13,187	17,336	68,933	3.2×10^{-4}

Table 5: Dataset statistics

Baselines. We compare VoteGCL against three categories of baseline models:

- **Vanilla graph-based recommendation baselines.** LightGCN (He et al. 2020) simplifies GCNs by using only neighborhood aggregation. NGCF (Wang et al. 2019) propagates embeddings over a bipartite graph to capture collaborative signals. MF-BPR (Koren, Bell, and Volinsky 2009) applies matrix factorization with BPR loss to model user-item interactions.
- **Traditional augmented graph recommendation baselines.** SGL (Zhou et al. 2023) applies contrastive learning to LightGCN using view augmentations to reduce bias and noise. GraphDA (Fan et al. 2023b) refines graph structure with user-user/item-item correlations and top-K sampling. GraphAug (Zhang et al. 2024) employs a graph information bottleneck for denoised, adaptive self-supervision. XSimGCL (Yu et al. 2022) replaces graph augmentations with noise-based perturbations for simplified and effective contrastive learning.
- **LLM-enhanced recommendation baselines.** KAR (Xi et al. 2024) augments recommendations with LLM-extracted reasoning and factual knowledge via a hybrid-expert adaptor. RLMRec (Ren et al. 2024) aligns LLM-based semantic representations with collaborative signals through cross-view learning. LLMRec (Wei et al. 2024) enriches user-item graphs using LLM-guided edge reinforcement, item attribute enhancement, and user profiling with denoising.

Detail Metrics. We provide definitions of all evaluation metrics used in our experiments as follows:

Dataset	Metric	Number of selected top-K		
		1	2	3
MovieLens-100K	N@10	0.0494	0.0448	0.0439
	A@10	0.0617	0.0593	0.0607
Amazon Scientific	N@10	0.0993	0.0911	0.0887
	A@10	0.2795	0.2041	0.1855

Table 6: Performance comparison across the number of majority-voting

- **Average Percentage of Long Tail Items (APLT):** measures the average percentage of long tail items in the recommended lists

$$\text{APLT@K} = \frac{1}{|U_t|} \sum_{u \in U_t} \frac{|\{i, i \in (L_u @ K \cap \Gamma)\}|}{|L_u @ K|}$$

Where item i has been rated in the training set. $L_u @ K$ is the recommended list of top K relevant items for user u and $|U_t|$ is the number of users in the test set, Γ represents for long tail item set.

- **Recall@K** measures the proportion of relevant items that appear in the top- K recommended list for each user:

$$\text{Recall@K} = \frac{1}{|U_t|} \sum_{u \in U_t} \frac{|L_u @ K \cap R_u|}{|R_u|}$$

Where R_u is the set of ground-truth relevant items for user u .

- **NDCG@K** evaluates the ranking quality of the top- K recommended items, emphasizing higher ranks:

$$\text{NDCG@K} = \frac{1}{|U_t|} \sum_{u \in U_t} \frac{\text{DCG@K}_u}{\text{IDCG@K}_u},$$

$$\text{DCG@K}_u = \sum_{i=1}^K \frac{\mathbb{I}(L_u[i] \in R_u)}{\log_2(i+1)},$$

$$\text{IDCG@K}_u = \sum_{i=1}^{\min(K, |R_u|)} \frac{1}{\log_2(i+1)}$$

where $L_u[i]$ denotes the i -th item in the recommended list for user u , and $\mathbb{I}(\cdot)$ is the indicator function.

A.2 Number of Augmented Interactions

Table 6 shows that selecting only the top-1 item after majority-voting LLM reranking consistently yields the best performance across both datasets, in terms of both NDCG@10 and APLT@10. As K increases, the additional items introduced into the augmented data may contribute noise, potentially degrading recommendation quality. This effect is more pronounced in the Amazon Scientific dataset, where both metrics decline significantly when $K > 1$. We therefore recommend fixing $K = 1$ and adjusting the contrastive weight to better navigate the accuracy-diversity trade-off.

Retrieval	MovieLens-100K			Amazon Scientific		
	Base N@10	N@10	A@10	Base N@10	N@10	A@10
XSimGCL	0.0398	0.0495	0.0662	0.0896	0.0993	0.2795
SGL	0.0407	0.0512	0.0608	0.0911	0.0950	0.2850
LightGCN	0.0334	0.0494	0.0617	0.0853	0.0920	0.2720

Table 7: Performance comparison of different retrieval methods and VoteGCL on retrieval-augmented data. "Base N@10" denotes the top-10 recommendation performance of the retrieval models themselves, while "N@10" and "A@10" reflect the performance of VoteGCL using the corresponding retrieved candidates.

A.3 Different retrieval models

Table 7 presents a comparison of different retrieval models and their impact on the performance of the VoteGCL reranking framework across two datasets. The results show that stronger retrieval models, such as XSimGCL and SGL, generally yield higher quality candidate items, which in turn lead to improved reranking performance by VoteGCL, as reflected in higher N@10 and A@10 scores compared to when using LightGCN. For instance, on the MovieLens-100K dataset, using XSimGCL as the retrieval model enables VoteGCL to achieve an A@10 of 0.0662, outperforming the LightGCN-based pipeline. Similarly, in the Amazon Scientific dataset, SGL provides the best A@10 of 0.2850. These findings suggest that the choice of retrieval model has a significant influence on the overall effectiveness of the recommendation system. While we recommend deploying stronger retrieval models in industrial settings to maximize performance, LightGCN remains a practical option due to its simplicity, fast inference, and reasonably competitive accuracy.

A.4 Different LLMs to augment data

Table 8 compares the ranking performance of different large language models (LLMs) and their impact on the VoteGCL reranking framework across two datasets. Overall, GPT-3.5-turbo consistently outperforms Qwen2.5-14B-Instruct across all metrics, likely due to its stronger parametric knowledge and more robust instruction-following capabilities. This advantage is reflected in both the base recommendation performance and the downstream gains achieved by VoteGCL using the LLM-augmented inputs. Nevertheless, Qwen2.5-14B-Instruct still delivers competitive results, making it a viable free alternative for scenarios where cost or deployment constraints prohibit the use of proprietary APIs like GPT. This highlights a practical trade-off between performance and accessibility when selecting LLMs for recommendation tasks.

A.5 Cost Efficiency

Cost and accuracy trade off. Figure 7 illustrates the cost-accuracy trade-offs associated with two complementary strategies for improving recommendation performance: increasing the number of LLM-based majority-vote rerankings and augmenting data for a subset of users. On the left, we observe that increasing the number of votes leads to consistent improvements in NDCG@10 across both datasets, albeit with rising inference cost. On the right, augmenting a greater proportion of users also yields notable accu-

racy gains with incremental cost. Notably, both evaluations show that substantial improvements in NDCG@10 can be achieved with relatively modest cost increases, for instance, from just 6 votes onward or augmenting only 10% of users. Based on these trends, we adopt a configuration using 8-way majority voting and data augmentation for 25% of users, which provides a strong balance between accuracy and cost.

Cost Comparison with Different Numbers of Candidates

Table 9 presents the performance and cost trade-offs when varying the number of retrieved candidates injected into the prompt. Increasing the number of candidates retrieved from the LightGCN model enhances the likelihood of including relevant items, leading to improved recall and NDCG@10 scores. However, this comes at the cost of longer input prompts, resulting in increased token usage and inference cost. While the best performance is achieved at $k = 20$, we observe that $k = 10$ offers a more favorable balance between accuracy gains and cost, making it our default choice for downstream prompting.

Cost Comparison with other methods. Table 10 compares the performance of our method with and without reasoning prompts against the LLMRec (Wei et al. 2024) baseline on the Netflix dataset. By leveraging the reasoning capabilities of LLMs in our prompt design, we guide the model to produce a reranked list explicitly. When disabling reasoning in the prompt, the output is limited to a simple format (e.g., "A-B-C-...") without any natural language explanation, resulting in minimal output tokens and significantly reduced cost. Notably, even with this no-reasoning prompt, the method still achieves substantial improvements in NDCG@10 relative to cost, outperforming the LLMRec baseline in both effectiveness and cost-efficiency. Under the default reasoning setting, VoteGCL demonstrates strong overall performance while maintaining a favorable cost-performance trade-off compared to LLMRec.

LLM	MovieLens-100K			Amazon Scientific		
	Base N@10	N@10	A@10	Base N@10	N@10	A@10
gpt-3.5-turbo	0.0395	0.0494	0.0617	0.0821	0.0993	0.2795
Qwen2.5-14B-Instruct	0.0342	0.421	0.0598	0.0784	0.0926	0.2612

Table 8: Ranking performance comparison of different LLMs on two datasets, along with the performance of VoteGCL on the corresponding LLM-augmented data. "Base N@10" indicates the LLM’s recommendation performances, while "N@10" and "A@10" represent VoteGCL performance with LLM-augmented inputs.

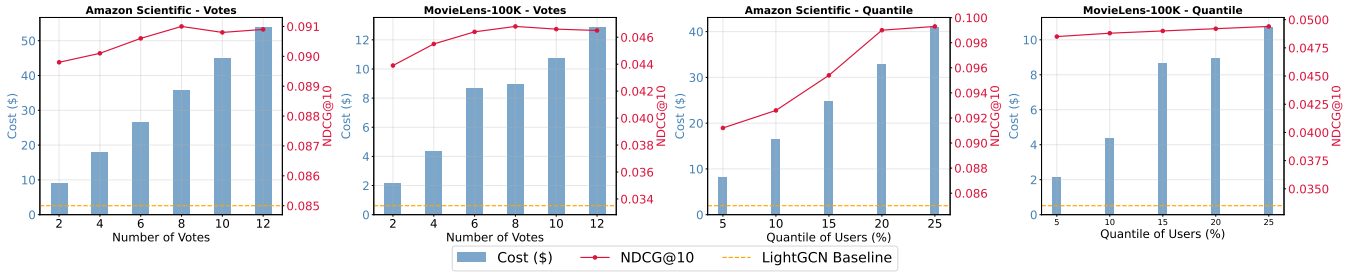


Figure 7: Cost and NDCG@10 performance trade-offs across two datasets (Amazon Scientific and MovieLens-100K). Left: Increasing the number of majority-vote LLM rerankings improves accuracy but raises inference cost. Right: Augmenting a larger quantile of user data yields accuracy gains with corresponding increases in cost.

Number of candidates	Amazon Scientific			MovieLens-100K		
	k=5	k=10	k=20	k=5	k=10	k=20
Cost (\$)	35.73	41.05	48.02	8.07	8.97	10.14
Recall@10	0.1378	0.1386	0.1437	0.0801	0.0886	0.0858
NDCG@10	0.0984	0.0994	0.1043	0.0414	0.0494	0.0463
Rate of Recall improvement to cost (%)	75.37	67.46	67.44	56.61	174.65	118.40
Rate of NDCG improvement to cost (%)	96.86	87.92	89.09	296.76	534.07	380.74

Table 9: Performance comparison across different numbers of retrieved candidates injected into the prompt. We report the total cost (in USD). The rates of improvement relative to cost are computed as the percentage gain over the LightGCN baseline divided by the corresponding cost

Dataset	Method	Cost(\$)	N@10	A@10	Imp N(%)	CIR N(%)	Imp A(%)	CIR A(%)
Netflix	Without reasoning	9.47	0.0330	0.0639	49.32	5.21	134.36	14.19
	With reasoning	21.43	0.0351	0.0892	58.83	2.75	149.16	6.96
LLMRec		21.14	0.0336	0.0667	52.04	2.46	86.31	4.08

Table 10: Performance comparison on the Netflix dataset and the LLMRec baseline. We report the cost (in USD), NDCG@10 (N@10), and Accuracy@10 (A@10), along with the percentage improvements over the LightGCN baseline (Imp N and Imp A) and their corresponding cost-improvement ratios (CIR N and CIR A), calculated as improvement per dollar.

B Model Complexity

We compare the time complexity of VoteGCL with LightGCN and SGL-ED under a single-batch training setup, where in-batch negative sampling is adopted. Let $|\mathcal{E}|$ be the number of edges in the user-item bipartite graph, d the embedding dimension, B the batch size, M the number of nodes in a batch, and L the number of GCN layers.

Component	LightGCN	SGL-ED	VoteGCL
Adjacency Matrix	$O(2 \mathcal{E})$	$O(2 \mathcal{E} + 4\rho \mathcal{E})$	$O(2(\mathcal{E} + \mathcal{E}^+))$
Graph Encoding	$O(2 \mathcal{E} Ld)$	$O((2 + 4\rho) \mathcal{E} Ld)$	$O(2Ld(\mathcal{E} + \mathcal{E}^+))$
Prediction	$O(2Bd)$	$O(2Bd)$	$O(2Bd)$
Contrast	-	$O(Bd + BMd)$	$O(Bd + BMd)$

Table 11: The comparison of time complexity

Table 11 summarizes the time complexity comparison between LightGCN, SGL-ED, and our proposed VoteGCL. We observe that the main computational overhead of VoteGCL lies not in graph processing itself but in the data augmentation phase, where we apply Majority LLM Reranking to generate high-quality virtual interactions $\mathcal{E}^+$. This step occurs offline and is performed once before training, introducing no runtime burden during model optimization. In contrast, SGL-ED incurs significantly higher complexity during training due to explicit graph augmentations. It constructs two additional perturbed adjacency matrices, resulting in a higher cost for both adjacency preparation and graph encoding. Specifically, the encoding complexity of SGL-ED is roughly $(2 + 4\rho)$ times that of LightGCN, whereas VoteGCL only adds a modest $2|\mathcal{E}^+|Ld$ term. Furthermore, all models share the same prediction cost under BPR loss with in-batch negatives. For contrastive learning, VoteGCL and SGL-ED

have similar complexity $O(Bd + BMd)$, while LightGCN does not employ contrastive learning. Overall, VoteGCL achieves a favorable trade-off between performance and efficiency. By shifting the augmentation cost to a one-time LLM-based process, it maintains low training complexity comparable to LightGCN while benefiting from richer supervision akin to SGL-ED.

C Detail LLM Prompt

Listing 1: Detail Prompt Template for Movie Reranking

You are a movie recommendation system. Given the user history in chronological order, recommend an item from the candidate pool with its index letter.

USER HISTORY:
{user_history}
The user history shows the movies the user has watched in chronological order. The first movie is the oldest one the user watched, and the last movie is the most recent one. Each entry follows this format: [Position]. [Movie Title] ([Release Year]) [Genres] - Rating: [User Rating]
The user rating ranges from 1.0 to 5.0, with 5.0 being the highest level of enjoyment.

TOP 10 CANDIDATE MOVIE LIST:
{candidate_list}
Each candidate movie is listed with an index letter (A, B, etc.) followed by the movie title, release year, and genres.

REFERENCE EXAMPLE: A similar user had the following history:
{similar_user_history}
This user rated the candidate product as:
{candidate_user_history}

Your task is to reorder these candidates based on the user's preferences, where the first movie should be the one the user would most likely enjoy, and subsequent movies represent decreasing levels of preference, and based on the reference example, too. Please analyze and summarize the user's preferences in paragraph form, and write it in <think></think> tags at the beginning of your response. Then, explain your reasoning for the ordering of the candidate movies and write it in <reasoning></reasoning> tags. Finally, provide your recommended ordering of ALL candidates movies as a hyphen-separated list of indices (e.g., A-B-C-D-E-F-G-H-I-J) and place it in <output></output> tags.

Make sure to include ALL 10 movie indices in your response. The first index should represent the movie you believe the user would enjoy most, with subsequent indices representing decreasing levels of preference.

Template 1 illustrates the prompt template we designed for the movie recommendation task, using the MovieLens dataset as a primary example. However, the template is designed to be generalizable and can be easily adapted to other datasets with minimal modification. One key design decision is the flattening of user history metadata into a structured natural language format: [Position]. [Movie Title] ([Release Year]) [Genres] - Rating: [User Rating]. This flattening not only makes the prompt more readable and coherent to the language model but also facilitates compatibility with diverse datasets by allowing simple reformatting of metadata fields. Furthermore, we cast the recommendation task as a reranking problem using a verbalized format. Rather than requiring the model to regenerate full movie titles, the prompt provides an indexed list of candidate movies and asks the model to output a ranked sequence of index letters. This design significantly reduces generation cost and latency, while maintaining high interpretability. To encourage transparency and structured reasoning, the prompt explicitly

instructs the model to include preference analysis and justification using $\langle \textit{think} \rangle$ and $\langle \textit{reasoning} \rangle$ tags, followed by the final ordered list in an $\langle \textit{output} \rangle$ tag.

Additionally, the prompt incorporates collaborative filtering by injecting a reference example from a similar user. We identify the most similar user by computing cosine similarity between user embeddings, then include their history in the prompt. The last three rated items from a similar user serve as ground truth to further guide the model. This structured response format improves reproducibility and effectively leverages collaborative signals in recommendations. In practical deployments, the reasoning component can be made optional to reduce inference cost while preserving the core reranking functionality.

D Theoretical results

Proof of Theorem 1. For convenience, we recall our notations in the main paper, followed by our formal proof.

We consider a set of K items retrieved by a retrieval system, denoted by $\mathcal{I} = \{i_1, i_2, \dots, i_K\}$. Let $\mathcal{T}_K$ denote the set of all permutations over K elements:

$$\mathcal{T}_K = \{\sigma : \{1, 2, \dots, K\} \rightarrow \{1, 2, \dots, K\} \mid \sigma \text{ is bijective}\}.$$

Each permutation $\sigma \in \mathcal{T}_K$ represents a ranking assignment over the items in $\mathcal{I}$. Specifically, $\sigma(i_k)$ denotes the **rank** assigned to item i_k . In other words, lower values of $\sigma(i_k)$ indicate higher relevance or priority in the ranked list.

We define a random variable ς taking values in $\mathcal{T}_K$, with probability distribution

$$P : \mathcal{T}_K \rightarrow [0, 1], \text{ such that } \sum_{\sigma \in \mathcal{T}_K} P(\varsigma = \sigma) = 1.$$

In our work, all probabilities and distributions are conditioned on the prompt information, denoted as $\mathbf{c}$, which includes user and item information. We focus on one user u , meaning that $\mathbf{c}$ is fixed. Hence, we eliminate the appearance of $\mathbf{c}$ -dependence in our proof for better presentation. Throughout this work, we use σ to represent a permutation in $\mathcal{T}_K$, and write $P(\sigma)$ as shorthand for $P(\varsigma = \sigma)$, the probability assigned to permutation σ .

We draw N independent permutations $\varsigma^{(1)}, \varsigma^{(2)}, \dots, \varsigma^{(N)}$ from the distribution P . For each item $i_k \in \mathcal{I}$, where $k = 1, \dots, K$, we define $\varsigma^{(n)}(i_k)$ as the rank assigned to item i_k in the n -th permutation. Fix two items i_j and i_k , with $j, k \in \{1, \dots, K\}$. Define

$$S_j = \sum_{n=1}^N g(\varsigma^{(n)}(i_j)), \quad S_k = \sum_{n=1}^N g(\varsigma^{(n)}(i_k)),$$

where $g : \mathbb{R} \rightarrow \mathbb{R}$ denotes any bounded strictly decreasing function. This assumption is natural and intuitive because items with a lower rank should have a higher score.

Let us define the difference random variables:

$$D_n = g(\varsigma^{(n)}(i_j)) - g(\varsigma^{(n)}(i_k)), \quad \text{for } n = 1, \dots, N.$$

Then we can write:

$$S_j - S_k = \sum_{n=1}^N D_n.$$

Because the function $g(\cdot)$ is bounded, there exist real numbers $A < B$ such that for all x , $A \leq g(x) \leq B$. Therefore, for any permutation ς , the difference $D_n = g(\varsigma^{(n)}(i_j)) - g(\varsigma^{(n)}(i_k))$ satisfies $-(B-A) \leq D_n \leq B-A$.

Let $\mathbb{E}[D_n]$ be the expected score difference between i_j and i_k under the distribution P .

Applying Hoeffding's inequality: if $D_1, \dots, D_N$ are independent with $D_n \in [a, b]$, then for any $t > 0$,

$$\mathbb{P}\left(\sum_{n=1}^N (D_n - \mathbb{E}[D_n]) \geq t\right) \leq \exp\left(-\frac{2t^2}{N(b-a)^2}\right).$$

In our setup, $a = -(B-A)$ and $b = B-A$, so $b-a = 2(B-A)$. Thus,

$$\begin{aligned} \mathbb{P}(S_j > S_k) &= \mathbb{P}\left(\sum_{n=1}^N D_n > 0\right) \\ &= \mathbb{P}\left(\sum_{n=1}^N (D_n - \mathbb{E}[D_n]) > -N\mathbb{E}[D_n]\right). \end{aligned}$$

If $\mathbb{E}[D_n] < 0$, then $-N\mathbb{E}[D_n] > 0$ and Hoeffding's inequality gives

$$\begin{aligned} \mathbb{P}(S_j > S_k) &\leq \exp\left(-\frac{2(-N\mu)^2}{N[2(B-A)]^2}\right) \\ &= \exp\left(-\frac{N\mu^2}{2(B-A)^2}\right). \end{aligned}$$

Let us define $\mu = \mathbb{E}[\varsigma^{(n)}(i_j) - \varsigma^{(n)}(i_k)]$. Because $g(\cdot)$ is bounded and strictly decreasing, $\mathbb{E}[D_n] > 0$ is equivalent that $\mu > 0$ $\square$

This formally characterizes the concentration of the aggregated comparison as N increases. The derived concentration result provides a rigorous guarantee for comparing the aggregated scores for pairs of items under repeated sampling from a stochastic permutation model. The bound demonstrates that, as long as the expected difference in scores $\mu = \mathbb{E}[D_n]$ is strictly negative, the probability that the "worse" item i_j outperforms the "better" item i_k after N samples decays exponentially in N . Notably, the exponent depends quadratically on the absolute value of the mean difference $|\mu|$ and inversely on the squared range of the score differences $(B-A)^2$. Thus, for a reasonably well-separated pair of items and a bounded scoring function, even a modest number of samples N suffices to make the ordering highly reliable.

Before proceeding with the proof for Corollary 1, we provide its self-contained formal statement in Corollary 2.

Corollary 2 (Reciprocal Rank Aggregation). *Let $\mathcal{I} = \{i_1, i_2, \dots, i_K\}$ be a set of K items, and let $\mathcal{T}_K$ denote the set of all permutations over $\{1, 2, \dots, K\}$. Let P be any*

probability distribution over $\mathcal{T}_K$, and let $\varsigma^{(1)}, \dots, \varsigma^{(N)}$ be independent random permutations drawn from P .

Define the score function $g(x) = 1/(x+1)$, and for each item $i \in \mathcal{I}$, define its aggregated score as

$$S(i) = \sum_{n=1}^N \frac{1}{\varsigma^{(n)}(i) + 1},$$

where $\varsigma^{(n)}(i)$ denotes the rank assigned to item i by permutation $\varsigma^{(n)}$.

Fix two distinct items $i_j, i_k \in \mathcal{I}$, and define the expected rank gap $\mu = \mathbb{E}[\varsigma^{(n)}(i_j) - \varsigma^{(n)}(i_k)]$. If $\mu > 0$, then the probability that item i_j receives a higher aggregated score than item i_k satisfies the concentration bound

$$\Pr(S(i_j) > S(i_k)) \leq \exp\left(-\frac{N\mu^2}{2\left(1 - \frac{1}{K+1}\right)^2}\right).$$

Proof of Corollary 1. For $x \in \{1, 2, \dots, K\}$, $g(x)$ is maximized when $x = 1$, so $B = 1$. It is minimized when $x = K$, so $A = 1/(K+1)$. Therefore, for all $i_j, i_k \in \mathcal{I}$ and for all permutations, the values of g satisfy $A \leq g(x) \leq B$, and $B - A = 1 - \frac{1}{K+1} = \frac{K}{K+1}$. Substituting A and B into Theorem 1 gives the stated bound. $\square$