Process-Supervised LLM Recommenders via Flow-guided Tuning

Chongming Gao*
chongminggao@ustc.edu.cn
University of Science and Technology of China
Hefei, China

Mengyao Gao*
mengyao0301@mail.ustc.edu.cn
University of Science and Technology of China
Hefei, China

Chenxiao Fan
simonfan@mail.ustc.edu.cn
University of Science and Technology of China
Hefei, China

Shuai Yuan
syuanaf@connect.ust.hk
Hong Kong University of Science and Technology
Hong Kong, China

Wentao Shi
shiwentao123@mail.ustc.edu.cn
University of Science and Technology of China
Hefei, China

Xiangnan He†
xiangnanhe@gmail.com
MoE Key Lab of BIPC, University of Science and Technology of China
Hefei, China

Abstract

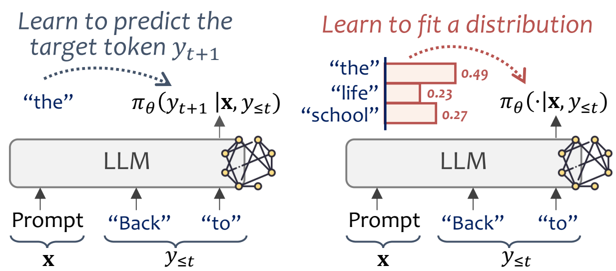
While large language models (LLMs) are increasingly adapted for recommendation systems via supervised fine-tuning (SFT), this approach amplifies popularity bias due to its likelihood maximization objective, compromising recommendation diversity and fairness. To address this, we present **Flow**-guided fin**e**-tuning **r**ecommender (Flower), which replaces SFT with a Generative Flow Network (GFlowNet) [6] framework that enacts process supervision through token-level reward propagation. Flower's key innovation lies in decomposing item-level rewards into constituent token rewards, enabling direct alignment between token generation probabilities and their reward signals. This mechanism achieves three critical advancements: (1) popularity bias mitigation and fairness enhancement through empirical distribution matching, (2) preservation of diversity through GFlowNet's proportional sampling, and (3) flexible integration of personalized preferences via adaptable token rewards. Experiments demonstrate Flower's superior distribution-fitting capability and its significant advantages over traditional SFT in terms of accuracy, fairness, and diversity, highlighting its potential to improve LLM-based recommendation systems. The implementation is available via https://github.com/Mr-Peach0301/Flower.

CCS Concepts

• **Information systems → Recommender systems**.

Keywords

Large Language Model Recommenders,Generative Flow Networks, Diversity-aware Fine-tuning, Process Supervision



(a) Supervised fine-tuning (b) Flow-guided fine-tuning

Figure 1: Illustration of two tuning paradigms in LLM-based next-item recommendation tasks

ACM Reference Format:
Chongming Gao, Mengyao Gao, Chenxiao Fan, Shuai Yuan, Wentao Shi, and Xiangnan He. 2025. Process-Supervised LLM Recommenders via Flow-guided Tuning. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '25), July 13–18, 2025, Padua, Italy.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3726302.3729981

1 Introduction

In recent years, recommendation systems powered by Large Language Models (LLMs) have emerged as a rapidly advancing field of research [23, 41]. Leveraging LLMs' vast repository of world knowledge and advanced learning capabilities, LLMs have demonstrated remarkable potential to enhance recommendation accuracy [4, 24], improve fairness [9, 15], and deliver more explainable [28, 42] and controllable [7, 27, 38] recommendation results. These advancements contribute significantly to improving the overall user experience in recommendation systems.

While effective in general-purpose tasks, pretrained LLMs require supervised fine-tuning (SFT) to acquire the knowledge specific to downstream tasks. In recommendation tasks, preference modeling is a key objective. A commonly used SFT paradigm involves constructing instruction data from historical user behavior sequences, enabling the LLM to predict the next item based on prior interactions [3, 4, 10]. Items are typically represented by their titles, which consist of multiple tokens generated sequentially by the LLM. SFT leverages the cross-entropy (CE) loss to maximize

*Both authors contributed equally to this research.
†Corresponding Author.

SIGIR '25, Padua, Italy

ACM ISBN 979-8-4007-1592-1/2025/07
https://doi.org/10.1145/3726302.3729981

the likelihood of the target labels. For example, in a next-movie recommendation task (Fig. 1(a)), the movie title "Back to the Future" is generated token by token. Given the prompt and the first two tokens, "Back" and "to", SFT optimizes the prediction of the next ground-truth token, "the". While effective, SFT introduces two critical challenges for recommendation systems:

- **Limited diversity.** SFT often drives models to produce high-probability tokens, leading to overfitting on dominant patterns in the training data [20]. As a result, the model generates homogeneous, less personalized recommendations–undermining the core goal of recommender systems.
- **Popularity bias amplification.** SFT reinforces biases present in both fine-tuning data and pretraining corpora [9], causing models to over-recommend popular items while underexposing niche content, which harms fairness and user experience.

To address these limitations, existing solutions in LLM-based recommendation systems (LRSs) can be broadly categorized into two approaches. The first approach involves modifying the SFT learning process. For example, assigning different weights to samples to adjust the learning loss [15] can mitigate category-specific popularity bias but fails to enhance the diversity of LLM-generated outputs. Another strategy is multi-stage SFT, where expert priors are integrated into successive stages to promote diversity [7]. However, this method relies heavily on manually designed, multi-stage workflows, which are prone to error propagation and may exacerbate bias. The second approach emphasizes post-SFT policy optimization to better align LLMs with human preferences, aiming to alleviate popularity bias and enhance diversity. For example, reinforcement learning with human feedback (RLHF) leverages reward signals to guide the recommender toward diverse outputs [27]. However, RLHF's reward maximization objective often leads to a collapse toward high-reward outcomes, which can further reduce diversity [17, 43]. More recently, Direct Preference Optimization (DPO) [32] has been applied to LRSs, using contrastive positive and negative samples to reduce popularity bias [10, 21]. Despite its promise, DPO tends to induce distributional collapse by driving the probabilities of negative samples to zero [1], ultimately limiting recommendation diversity [10].

The shortcomings of these multi-stage pipelines or post-hoc corrections motivate our fundamental rethinking of alignment paradigms for LLM-based recommenders. Rather than patching SFT's limitations through auxiliary mechanisms, we propose **Flow**-guided fin**e**-tuning **r**ecommender (Flower), which replaces conventional SFT with the generative flow network (GFlowNet)-based fine tuning. GFlowNet is a diversity-seeking reinforcement learning algorithm that trains policies to sample items with probabilities proportional to a given reward function rather than simply maximizing the reward [6].

By leveraging GFlowNet's mechanism, we conceptualize the prefix tree of all feasible items as an irreversible flow network, where "flow" refers to the unnormalized probabilities moving from the root node to the leaf nodes. The reward assigned to each item corresponds to the flow at the leaf nodes. This framework enables us to compute the flow values at each branching point of the network, which serve as token-level rewards for next-token prediction. With

Table 1: Comparison of finetuning and alignment methods.

Paradigm	Method	Reward	Objective
Finetuning	SFT	/	Maximize the probability for each ground-truth token.
	Flower	Flow-guided process reward	Align the next-token probability distribution with token-level reward distribution.
Preference alignment	RLHF	Outcome reward	Maximize the reward of the generated item or item list.
	DPO	/	Maximize the scores of the chosen item over the rejected.

these token-level rewards, Flower supervises the generation probabilities of each token during fine-tuning. As shown in Fig. 1(b), unlike SFT, which focuses on optimizing the prediction of a single ground truth token at a time, the objective of Flower is to align the policy (i.e., the predicted probability distribution over tokens) with the token-level reward distribution across all feasible tokens. Beyond fairness and diversity considerations, we further incorporate personalized user preferences into the derived token-level rewards using an auxiliary model, thereby enhancing both the accuracy and personalization of the recommendations.

Flower is a new fine-tuning paradigm superior to conventional SFT in enhancing diversity while maintaining accuracy. After tuning with Flower, additional alignment methods such as RLHF and DPO can still be applied. A comparison of different methods is provided in Table 1. The main contributions are as follows:

- We identify two key issues caused by the SFT mechanism in LRSs: limited diversity and popularity bias, both of which degrade recommendation performance.
- We propose Flower, a fine-tuning paradigm that introduces the concept of flow to assign token-level rewards to all feasible next tokens, providing process-level supervision for LLMs.
- Our token-level rewards are heuristically assigned, requiring no additional learning. This approach is simple yet effective and can be easily modified to incorporate personalized preferences.
- Experiments validate the superiority of Flower over SFT in enhancing accuracy, diversity, and fairness. Moreover, these advantages are preserved even when further alignment methods such as PPO and DPO are applied.

2 Preliminary

This section introduces SFT for next-item recommendation with LLMs, followed by an overview of the basics of GFlowNets.

2.1 SFT for Next-item Recommendation

Next-item prediction is a fundamental task in recommender systems, where the goal is to predict the next item a user is likely to interact with, based on their historical behavior sequence. Leveraging the generative capabilities of LLMs, this task can be cast as generating the next recommended item directly.

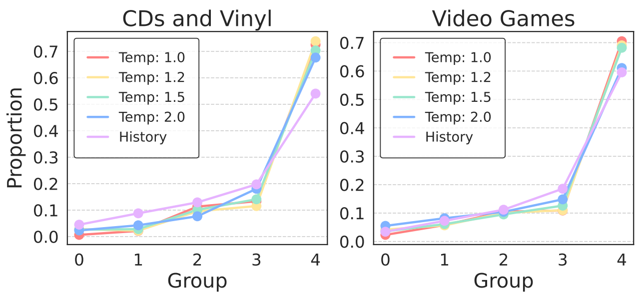


Figure 2: Distribution of the top-5 recommendations across five groups divided by their frequency of occurrence.

Following the instruction-tuning paradigm of BIGRec [3], we define the task as: given a prompt $\mathbf{x}$—typically describing the task and listing the user's previously interacted items—the model policy π_θ is trained to generate the next item $\mathbf{y}$, represented as a token sequence $\mathbf{y} = [y_1, y_2, \cdots, y_T]$.

The model is fine-tuned using standard cross-entropy loss with teacher forcing:

$$\mathcal{L}_{\text{SFT}} = -\frac{1}{T}\sum_{t=1}^{T} \log \pi_\theta(y_t \mid \mathbf{x}, y_{1:t-1}), \tag{1}$$

where π_θ denotes the model's predicted probability for token y_t given prior tokens and the input prompt.

After fine-tuning, the LLM is used to generate recommendations during inference. However, it may produce invalid or nonexistent items. To mitigate this, Bao et al. [3] propose a matching mechanism to align outputs with real items, while Bao et al. [4] introduce constrained decoding to restrict token sampling to valid continuations. We adopt the latter to ensure recommendation validity.

2.2 Problems in SFT-based Recommendations

SFT introduces bias and fairness issues in LRSs [9, 10, 15]. To illustrate this, we utilize the CDs and Video datasets from the Amazon Review datasets, dividing all items into five groups based on their frequency of occurrence in the historical sequences of the training data. After fine-tuning the LLM with BIGRec[1], we analyze the distribution of the top-5 recommended items across these five groups. Furthermore, to investigate the influence of the temperature parameter during the inference stage, we conduct experiments with temperatures set to 1.0, 1.2, 1.5, and 2.0.

As shown in Figure 2, Group 0 corresponds to the least popular items, while Group 4 represents the most popular ones. The top-5 recommendation results reveal a noticeable tendency to recommend items from the most popular group, disproportionately favoring when compared to the historical sequences. This highlights a clear presence of popularity bias, making the recommender system unfair to less popular items. Moreover, as the temperature increases, the unfairness is partially alleviated; however, both popularity bias and unfairness persist to varying degrees.

[1]Detailed experimental settings are provided in Section 4

2.3 Generative Flow Networks (GFlowNets)

Generative Flow Networks (GFlowNets) [5] are a class of generative models that learn stochastic policies for sequential decision-making. Inspired by physical flow systems, GFlowNets define unnormalized probabilistic flows to model diverse outcomes, allocating probability mass proportional to outcome rewards.

GFlowNets operate over a directed acyclic graph (DAG), where each path represents a sequence of actions from a root state to a terminal state. The flow through each path determines the probability of generating a particular outcome, enabling diverse and reward-aligned generation.

Formally, let $G = (\mathcal{S}, \mathcal{A})$ denote the DAG, where nodes $s \in \mathcal{S}$ are *states* and directed edges $(s_1 \to s_2) \in \mathcal{A}$ are *actions*. If $(s_1 \to s_2)$ exists, then s_2 is a *child* of s_1, and s_1 is its *parent*. The graph has a unique *initial state* s_0 with no parents, and a set of *terminal states* $\mathcal{Y}$ with no children. A *trajectory* $\tau \in \mathcal{T}$ is a sequence of transitions $\tau = (s_m \to s_{m+1} \to \cdots \to s_n)$.

A *forward policy* is a set of distributions $P_F(-|s)$ over the children of each nonterminal state $s \in \mathcal{S}$, inducing a trajectory distribution:

$$P_F(\tau = (s_0 \to \cdots \to s_n)) = \prod_{i=0}^{n-1} P_F(s_{i+1} \mid s_i). \tag{2}$$

Given a nonnegative reward function $R : \mathcal{Y} \to \mathbb{R}_{\geq 0}$, the objective of GFlowNets is to estimate a policy P_F such that the likelihood of sampling $y \in \mathcal{Y}$ is proportional to $R(y)$:

$$R(y) = Z \sum_{\tau=(s_0 \to \cdots \to s_n = y)} P_F(\tau) \quad \forall y \in \mathcal{Y}, \tag{3}$$

where Z is a normalization constant satisfying $Z = \sum_{y \in \mathcal{Y}} R(y)$.

2.4 Training Objective for GFlowNets

The sum over all $\tau \in \mathcal{T}$ in Eq. (3) is generally intractable, prompting the use of auxiliary variables to facilitate learning. We adopt the Subtrajectory Balance (SubTB) objective [29], which introduces the *state flow* $F(s)$ — a nonnegative scalar representing the total flow through state s:

$$F(s) := F(\{\tau \in \mathcal{T} : s \in \tau\}) = \sum_{\tau \in \mathcal{T} : s \in \tau} F(\tau). \tag{4}$$

Subtrajectory Balance. For any trajectory segment $\tau_{m:n} = (s_m \to \cdots \to s_n)$, the following constraint holds:

$$F(s_m) \prod_{i=m}^{n-1} P_F(s_{i+1}|s_i; \theta) = F(s_n) \prod_{i=m}^{n-1} P_B(s_i|s_{i+1}; \theta), \tag{5}$$

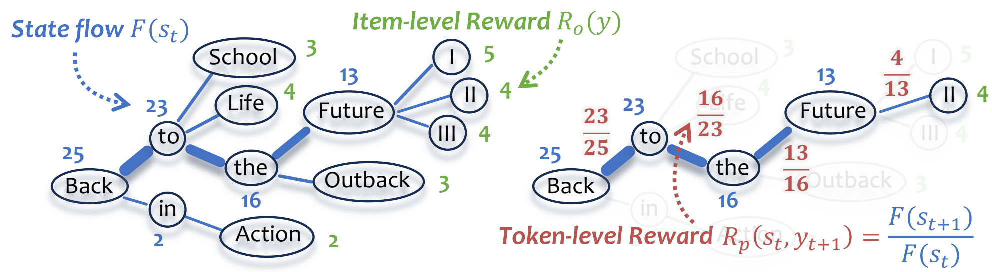
where $P_F(s_{i+1}|s_i; \theta)$ is the *forward policy*, parameterized by θ, representing the probability of taking action $(s_i \to s_{i+1})$ conditioned on the state s_i. Similarly, $P_B(s_i|s_{i+1}; \theta)$ is the *backward policy*, modeling the reverse transitions. Here, $F(y) = R(y)$ if y is terminal, and $F(s_0) = Z$ if s_0 is the initial state.

The Subtrajectory Balance constraint leads to the following *subtrajectory balance objective*:

$$\ell_{\text{SubTB}}(\tau_{m:n}) = \left(\log \frac{F(s_m) \prod_{i=m}^{n-1} P_F(s_{i+1}|s_i; \theta)}{F(s_n) \prod_{i=m}^{n-1} P_B(s_i|s_{i+1}; \theta)} \right)^2. \tag{6}$$

It is straightforward to observe that the objective in Eq. (6) satisfies the desired condition of GFlowNets outlined in Eq. (3).

Movie title	Reward
Back to School	3
Back to Life	4
Back to the Future I	5
Back to the Future II	4
Back to the Future III	4
Back to the Outback	3
Back in Action	2



(a) Item-level outcome rewards (b) "Flow" in prefix tree of all items (c) Token-level rewards for process supervision

Figure 3: Illustration of the prefix tree, state flow, item-level rewards, and flow-guided token-level rewards in Flower.

3 Method: Flower

In this section, we introduce the problem formulation and present how we fine-tune an LLM using our proposed method, Flower, to achieve process supervision.

3.1 Problem Formulation

Let $\mathcal{I}$ denote the set of valid items in the dataset, where each item $i \in \mathcal{I}$ is represented by its title—a token sequence $\mathbf{y}^{(i)} = [y_1^{(i)}, y_2^{(i)}, \cdots, y_T^{(i)}]$, with each token $y_t^{(i)}$ belonging to the LLM's vocabulary. The titles collectively form a prefix tree, where shared prefixes define common paths. For instance, "Back to the Future II" and "Back to Life" share the prefix "Back to" before diverging. Note that in this example, words are used as tokens for illustration purposes. In practice, an item is automatically tokenized by an open-source LLM, such as LLaMA or Qwen.

This prefix tree is a special DAG: nodes represent tokens, and edges denote valid transitions to the next token. The root corresponds to the empty sequence, and each path to a terminal node forms a complete item title. We define:

- *State* s_t as a prefix sequence $s_t = y_{\le t} = [y_0, \cdots, y_t]$, representing a node's path from the root.
- *Action* a_t as appending a valid token y_{t+1}, transitioning from the current state s_t to a child state s_{t+1}.

Fig. 3(b) shows an example prefix tree for the 7 movie titles in Fig. 3(a). At the node "to", the state is "Back to", and valid actions lead to "Back to School", "Back to Life", or "Back to the".

The traversal aligns with the GFlowNets framework, where the probability of generating an item is the product of transition probabilities along its path. For example, the probability of "Back to Life" is computed as the product over "Back" → "to" → "Life".

Remark: This formulation casts generation as traversal through a prefix tree, ensuring only valid tokens are sampled at each step. It aligns model outputs with the dataset while leveraging GFlowNets to promote diversity and reward-proportional generation.

3.2 LLM as The Policy for GFlowNets

We employ decoder-based LLMs to implement the policy in GFlowNets. Given the prompt $\mathbf{x}$, the model generates the tokens of an item y sequentially: $y_1 \to \cdots \to y_T$. The probability of generating the item y is defined as the product of the conditional probabilities of its tokens. Specifically, the forward policy Eq. (2) is expressed as:

$$P_F(y) = \prod_{t=1}^{T} \pi_\theta(y_t | \mathbf{x}, y_{1:t-1}), \tag{7}$$

where $y_{1:t-1}$ represents the sequence of tokens generated prior to step t, and π_θ is the decoder-based model parameterized by θ. For example, the probability of generating the movie title "Back to Life" is calculated as:

$$\begin{aligned} P_F(\text{“Back to Life”}) = \pi_\theta(\text{“Back”}|\mathbf{x}) \cdot \pi_\theta(\text{“to”}|\mathbf{x}, \text{“Back”}) \\ \cdot \pi_\theta(\text{“Life”}|\mathbf{x}, \text{“Back to”}). \end{aligned} \tag{8}$$

In the GFlowNets framework, the forward policy $P_F(s_{i+1}|s_i;\theta)$ in Eq. (5) is implemented as $\pi_\theta(y_t|\mathbf{x}, y_{1:t-1})$. Meanwhile, the backward policy $P_B(s_i|s_{i+1};\theta)$ is always equal to 1, since each node in the prefix tree has only one parent.

3.3 Flow-guided Token-level Reward

Denote $R_o(y)$ as the *outcome reward*, i.e., the item-level reward for generating an item y, we will derive the state flow $F(y)$ as below:

3.3.1 State Flow on The Prefix Tree. By setting the state flow at a terminal state equal to the outcome reward, i.e., $F(y) = R(y)$, and the initial flow equal to the total reward, $F(s_0) = Z = \sum_{y \in \mathcal{Y}} R_o(y)$, the flow for an intermediate state s is recursively defined as the sum of the flows of its child states:

$$F(s) = \sum_{s' \in \text{Child}(s)} F(s'). \tag{9}$$

Remark: Generally, estimating a proper flow in GFlowNets requires learning a flow function over the DAG [6]. However, benefiting from the prefix tree structure used in our approach, the flow can be directly computed without additional parameter estimation. This structure ensures computational simplicity and eliminates the need for a dedicated flow model.

Using the state flow, we can derive the objective of subtrajectory balance in Eq. (6) for any trajectory $\tau_{m,n} = (s_m \to \cdots \to s_n)$ as:

$$\mathcal{L}(\tau_{m,n}) = \left(\log \frac{F(s_m) \prod_{t=m}^{n-1} \pi_\theta(y_{t+1}|\mathbf{x}, y_{\le t})}{F(s_n)} \right)^2. \tag{10}$$

3.3.2 Process Reward. To better illustrate how the policy is optimized through the flow mechanism, we define the *process reward*, a token-level reward for generating the token y_{t+1} given the prompt

$\mathbf{x}$ and the previously generated tokens $y_{\leq t}$. As shown in Fig. 3(c), the process reward is defined as:

$$R_p(s_t, a_t) = R_p(y_{\leq t}, y_{t+1}) = \frac{F(s_{t+1})}{F(s_t)}. \tag{11}$$

Using this definition, the objective Eq. (10) can be rewritten as:

$$\mathcal{L}_R(\tau_{m,n}) = \left(\sum_{t=m}^{n-1} \log \pi_\theta(y_{t+1}|\mathbf{x}, y_{\leq t}) - \sum_{t=m}^{n-1} \log R_p(y_{\leq t}, y_{t+1}) \right)^2. \tag{12}$$

When the length of the subtrajectory is reduced to 2, the objective simplifies to directly fitting the policy $\pi_\theta(y_{t+1}|\mathbf{x}, y_{\leq t})$ to the token-level reward $R_p(y_{\leq t}, y_{t+1})$, thereby achieving process supervision.

3.3.3 Reward Setting. To evaluate bias and fairness issues in LRSs, a common approach is to analyze the mismatch between the distribution of ground-truth user preferences and the distribution of model-predicted results [9]. Based on this perspective, we define the outcome reward $R_o(y)$ of an item y as its frequency of occurrence in the historical sequences of the training data. The objective in Eq. (12) encourages the policy to generate items with a distribution aligned with the empirical data distribution, thereby addressing popularity bias and mitigating fairness issues.

However, this reward remains static across all users and does not account for personalized preferences. To address this limitation, we introduce a *preference score* p_{ui}, which predicts the likelihood of user u liking item i. This score can be obtained from any auxiliary model, such as a traditional recommendation system. Given p_{ui}, we incorporate personalization by modifying the process reward term $\log R_p(y_{\leq t}, y_{t+1})$ as: (1) $\frac{\log R_p(y_{\leq t}, y_{t+1})}{p_{ui}}$, or (2) $\log(p_{ui} \cdot R_p(y_{\leq t}, y_{t+1}))$. This adjustment effectively introduces user-specific information into the process rewards without altering the original flow derivation. In Section 4.5.1, we will compare the performance of these reward variants.

Remark: Many policy optimization methods employ complex process reward models (PRMs) [22, 39] to enhance reasoning in large language models [35, 45], requiring significant computational resources for learning and verification. In contrast, our approach adopts heuristically assigned rewards, which are simple yet effective. This design avoids additional parameter learning and ensures efficient process supervision.

3.4 Fine-tuning LLMs through Process Rewards

To fine-tune the policy π_θ, we integrate the original SFT loss $\mathcal{L}_{\text{SFT}}$ from Eq. (1) with the subtrajectory balance objective $\mathcal{L}_R$ in Eq. (12). SFT is trained on offline datasets, while the subtrajectory balance objective is optimized on-policy by generating a batch of items, traversing their title set $\mathcal{T}$, and evaluating all feasible subtrajectories. The combined loss function of Flower is formulated as:

$$\mathcal{L}_{\text{Flower}} = \mathcal{L}_{\text{SFT}} + \lambda \sum_{\tau \in \mathcal{T}} \sum_{0 \leq m < n \leq T} \mathcal{L}_R(\tau_{m,n}), \tag{13}$$

where $\tau_{m,n}$ represents a token subsequence from the title of a specific item. The hyperparameter λ controls the trade-off between the SFT loss and the subtrajectory balance objective.

Table 2: Dataset statistics before and after processing.

Dataset		#User	#Item	#Interaction
CDs and Vinyl 2015.10-2018.10	Before	440490	179277	815053
	After	7685	5841	69249
Video Games 2015.10-2018.10	Before	588656	45295	849496
	After	9066	3858	70483
Movies and TV 2017.10-2018.10	Before	204439	53855	349292
	After	3663	2653	31085

This combined loss preserves the supervised performance of SFT while leveraging GFlowNets to promote diversity and reward-proportionality, enabling the policy to generate fairer and more representative items.

4 Experiments

In this section, we conduct experiments to address the following research questions:

- **RQ1**: How effectively does Flower fit a specific distribution?
- **RQ2**: How do LLM-based methods perform in terms of accuracy, fairness, and diversity in the next-item recommendation task?
- **RQ3**: What is the impact of using Flower as a reference policy on RL and DPO-based methods?
- **RQ4**: What are the effects of the key factors in Flower?

4.1 Experimental Setup

4.1.1 Dataset. We conduct experiments on three real-world datasets from Amazon review data[2], including CDs, Video Games, and Movies. These datasets contain user review data from May 1996 to October 2018. Following the preprocessing strategy in [3], we truncate the dataset based on time information, remove unpopular users and items with fewer than five interactions, and limit the maximum item sequence length to 10. The datasets are chronologically split into training, validation, and test sets in an 8:1:1 ratio, detailed statistical information is provided in Table 2.

4.1.2 Evaluation Protocal. We evaluate top-k recommendation performance across three dimensions: Accuracy, Fairness, and Diversity. Accuracy and Fairness are assessed at the item level, while Diversity is evaluated at the word level.

For **Accuracy**, following prior work [3, 4, 15], we adopt two widely used metrics: Hit Ratio (HR@K) and Normalized Discounted Cumulative Gain (NDCG@K). **Fairness** is measured using DGU@K and MGU@K, which quantify the discrepancy between the group distribution of recommended titles in the top-k results and their distribution in the training set's historical sequences [15]. Titles are grouped by popularity: first, the frequency of each title in the training set is computed and sorted in descending order, then partitioned into eight equal-sized groups. Titles absent from the training set's historical sequences are assigned to the least popular group. To assess **Diversity**, we use two metrics: (1) Entropy of Words (H), which calculates the entropy of all English words in the recommended

[2]https://cseweb.ucsd.edu/~jmcauley/datasets.html#amazon_reviews

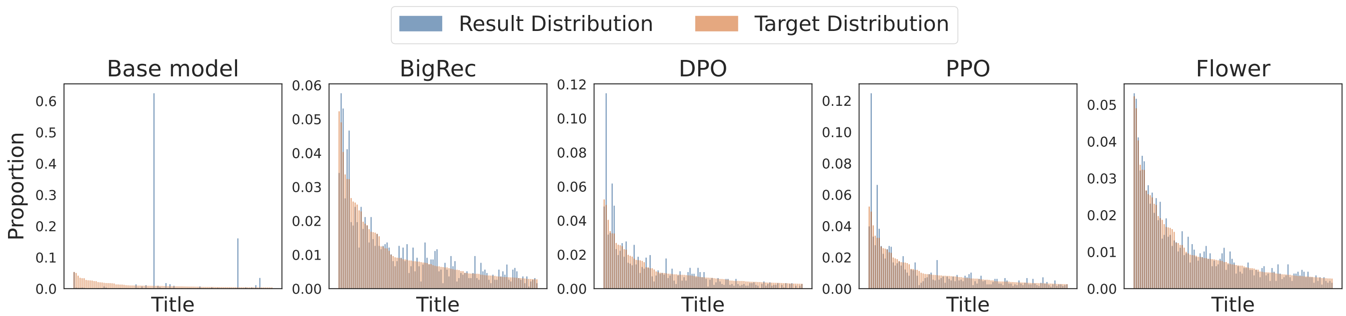


Figure 4: Comparison of the distributions between the target set and the recommended results across 100 movie titles.

titles, and (2) Type-Token Ratio (TTR), defined as the ratio of unique words to the total number of words in the recommendations.

In the comparison results, we report NDCG@5, HR@5, DGU@10, MGU@10, H, and TTR as evaluation metrics.

4.1.3 Baseline. We select one traditional sequential recommendation model and several SFT-based LRSs as baselines:

- **SASRec** [16] is a widely used sequential recommendation baseline employing a self-attention mechanism.
- **BIGRec** [3] is one of the earliest and most classic supervised fine-tuning (SFT) methods for directly generating item titles.
- **Temp** [4] adjusts the temperature coefficient for inference based on the trained BIGRec model.
- $\boldsymbol{D^3}$ [4] enhances BIGRec by removing length normalization to address amplification bias and incorporating a text-free assistant model, SASRec, in the inference stage to mitigate the homogeneity issue and improve recommendation diversity.
- **IFairLRS** [15] improves fairness in BIGRec by balancing recommendations across categories through weighting the SFT loss.

4.1.4 Implementation Details. For SASRec, we optimize using binary cross-entropy loss and the Adam optimizer, with a learning rate in [1e-2, 1e-3, 1e-4], a batch size of 1024, and weight decay in [1e-3, 1e-4, 1e-5, 1e-6]. For LLM-based methods, we use Qwen2.5-1.5B-Instruct as the base model, with the learning rate set to 3e-4, batch size to 128, maximum training epochs to 7, early stopping patience to 2, and optimize models using the AdamW optimizer. For experiments related to the temperature coefficient, we adjust it within the range of [1.2, 1.5, 2.0]. Considering the scale difference between the flow-guided loss and the SFT loss, λ is in the range of [0.01, 0.005, 0.001, 0.0005, 0.0001]. Unless otherwise specified, we use the process reward $\frac{\log R_p}{p_{ui}}$ described in Section 3.3.3 for Flower, where the preference score p_{ui} is obtained from SASRec.

4.2 Distribution Fitting Capability (RQ1)

Before applying our method to personalized recommendation problems, we first evaluate the distribution fitting capabilities of Flower compared to SFT, DPO, and PPO in a history-free recommendation scenario using the Movies and TV dataset. In this setup, the LLM is prompted to recommend a movie without providing any history or examples, aiming to assess how well each method aligns with the distribution of the training set, i.e., the target item set. We use the same target item set for all methods. After tuning, each method generates recommendations equal in size to the tuning dataset. We evaluate the mismatch between the distribution of items in the target set and the generated recommendations.

Table 3: Quantitative results of the distribution mismatch between the target set (T) and the recommended results (R) across 1500 movie titles. KL divergence and JS divergence are computed at both the title and token levels.

	Base Model	BIGRec	DPO	PPO	Flower
Title KL(T‖R)	20.235	5.114	17.765	9.985	**0.961**
Title KL(R‖T)	4.117	0.788	2.703	1.466	**0.190**
Title JS	0.513	0.184	0.449	0.341	**0.047**
Token KL(T‖R)	16.999	3.646	13.653	6.549	**1.982**
Token KL(R‖T)	5.291	1.307	2.940	2.174	**0.838**
Token JS	0.565	0.291	0.458	0.429	**0.217**

4.2.1 Qualitative Visualization. For illustration, we create the target set by sampling 100 items from the Movie dataset, preserving their interaction frequencies as shown in Table 2. We employed Qwen2.5-1.5B-Instruct as the base model. BIGRec uses all interactions in the target set as training data. For DPO, we randomly select one item as the chosen response and another less-interacted item as the rejected response. For PPO, we assign normalized interaction counts as item-level rewards. Additionally, we report the recommendation results of the base model (pre-trained LLM without tuning) as a reference.

The item-level distributions are illustrated in Fig. 4. The base model's recommendations concentrate on a few specific titles, while BIGRec skews heavily toward popular titles, a bias further amplified in DPO and PPO. In contrast, Flower effectively learns the target distribution, capturing titles with varying popularity and mitigating the unfairness observed in other methods.

4.2.2 Quantitative Analysis. To quantify these observations, we increase the target set to 1500 items and employ Qwen2.5-3B-Instruct as the base model. We compute the Kullback-Leibler (KL) divergence and Jensen-Shannon (JS) divergence between the generated and target distributions at both the token and item levels. As shown in Table 3, Flower achieves superior distribution fitting compared

Table 4: Performance of all methods evaluated in terms of accuracy, fairness, and diversity. The best results are bolded.

	CDs and Vinyl						Video Games						Movies and TV					
	NDCG↑	**HR↑**	**DGU↓**	**MGU↓**	**H↑**	**TTR↑**	**NDCG↑**	**HR↑**	**DGU↓**	**MGU↓**	**H↑**	**TTR↑**	**NDCG↑**	**HR↑**	**DGU↓**	**MGU↓**	**H↑**	**TTR↑**
SASRec	0.0641	0.0851	0.184	0.038	9.188	0.124	0.0369	0.0544	0.167	0.033	8.229	0.050	0.0902	0.1072	0.138	0.032	8.892	0.167
BIGRec	0.0573	0.0715	0.217	0.045	5.900	0.006	0.0326	0.0466	0.151	0.029	7.504	0.004	0.0930	0.1134	0.123	0.028	8.297	0.018
Temp	0.0503	0.0627	0.222	0.044	6.202	0.006	0.0306	0.0444	0.129	0.026	7.307	0.004	0.0852	0.1061	0.139	0.027	8.145	0.018
D3	**0.0812**	**0.0999**	0.355	0.072	7.635	0.013	0.0413	0.0607	0.220	0.041	7.645	0.005	**0.1007**	**0.1225**	0.147	0.033	8.348	0.020
IFairLRS	0.0621	0.0762	0.217	0.045	6.420	0.007	0.0396	0.0568	0.144	0.030	7.699	0.005	0.0957	0.1170	0.159	0.043	8.048	0.015
Flower	0.0700	0.0885	**0.075**	**0.021**	**7.919**	**0.013**	**0.0543**	**0.0799**	**0.108**	**0.023**	**7.750**	**0.005**	0.0959	0.1199	**0.076**	**0.026**	**8.808**	**0.023**

Table 5: Performance comparison of Flower (F) and BIGRec (B) as reference policies of RL and DPO-based methods.

	CDs and Vinyl						Video Games						Movies and TV					
	NDCG↑	**HR↑**	**DGU↓**	**MGU↓**	**H↑**	**TTR↑**	**NDCG↑**	**HR↑**	**DGU↓**	**MGU↓**	**H↑**	**TTR↑**	**NDCG↑**	**HR↑**	**DGU↓**	**MGU↓**	**H↑**	**TTR↑**
B_PPO	0.0519	0.0640	0.246	0.049	5.670	0.005	0.0282	0.0401	0.191	0.035	7.204	0.004	0.0871	0.1075	0.175	0.033	8.114	0.016
B_S-DPO	0.0712	0.0908	0.104	0.025	8.539	0.016	**0.0671**	**0.0900**	0.083	0.020	8.223	0.008	0.1037	0.1232	**0.070**	0.022	9.068	0.025
B_RosePO	0.0641	0.0810	0.105	0.023	**8.627**	0.017	0.0599	0.0786	0.286	0.057	**8.546**	0.008	0.1012	0.1178	0.145	0.030	9.347	0.027
B_DMPO	0.0718	0.0890	0.083	0.016	8.275	0.015	0.0424	0.0622	0.056	0.015	8.254	0.007	0.0960	0.1199	0.076	0.026	8.807	0.023
F_PPO	0.0620	0.0788	0.085	0.023	7.574	0.011	0.0565	0.0757	0.124	0.024	7.561	0.005	0.0963	0.1196	0.083	0.028	8.751	0.022
F_S-DPO	**0.0772**	**0.0944**	0.085	0.019	8.326	0.016	0.0636	0.0834	0.075	0.016	8.393	0.007	**0.1042**	**0.1269**	0.073	**0.017**	9.159	0.026
F_RosePO	0.0701	0.0872	0.127	0.028	8.608	0.017	0.0608	0.0799	0.305	0.059	8.501	**0.008**	0.1012	0.1214	0.188	0.037	**9.361**	**0.028**
F_DMPO	0.0731	0.0913	**0.063**	**0.012**	8.545	**0.017**	0.0644	0.0869	**0.043**	**0.013**	8.233	0.007	0.0974	0.1211	0.072	0.022	8.721	0.024

to other methods, validating its ability to enhance diversity and align with the target distribution.

4.3 Next-item Recommendation Results (RQ2)

We evaluate the recommendation performance of Flower and baseline methods on the next-item recommendation task using three open-world datasets. Unlike traditional recommendation models that rely on item IDs to represent titles, LLM-based models are inherently constrained by the text generation nature of LLMs, leading to a diversity gap compared to traditional methods. As a result, we exclude SASRec from diversity comparisons. The overall experimental results are summarized in Table 4, and the key observations are as follows:

- Compared to baseline methods, Flower achieves optimal fairness and diversity across all datasets. While Flower's accuracy is second only to D^3 on the Video Games dataset, it demonstrates consistent advantages in fairness and diversity. Notably, D^3 improves diversity by integrating SASRec collaborative information during inference but suffers from the poorest fairness among all methods, with a significant gap compared to Flower. This highlights Flower's balanced performance across all metrics.
- For methods specifically aimed at enhancing fairness, Flower outperforms IFairLRS across all metrics and datasets. This indicates that, compared to IFairLRS's approach of reweighting the entire title, Flower's use of probabilistic process supervision at the token level through GFlowNets achieves better results.
- Temp improves fairness and diversity over BIGRec by introducing higher randomness during inference. However, this improvement comes at the expense of accuracy. In contrast, Flower's approach of decomposing title rewards into token-level conditional probabilities allows for strict token-wise supervision. This mechanism ensures a balance between randomness and control, leading to simultaneous improvements in accuracy, fairness, and diversity without sacrificing any single dimension.
- Both Flower and D^3 leverage collaborative information from SASRec. However, D^3 focuses solely on accuracy optimization, neglecting fairness and diversity by not imposing additional constraints. As a result, D^3 achieves higher accuracy but suffers from significant fairness degradation. Conversely, Flower integrates fairness considerations into the collaborative information, enabling simultaneous improvements in accuracy, fairness, and diversity. This showcases Flower's ability to balance multiple objectives effectively.

4.4 Flower as a Reference Policy (RQ3)

Existing RLHF and DPO-based methods are typically fine-tuned on top of SFT. In contrast, Flower is a fine-tuning framework designed to address the diversity and unfairness issues inherent in SFT. We investigate the performance of Flower and BIGRec (an SFT-based method) as reference policies for preference alignment methods, such as RL and DPO-based approaches.

We select three DPO-based recommendation methods for comparison: **DMPO** [2] is framework that bridges the gap between LLMs and recommendation tasks by sampling multiple negative items as rejected responses. **S-DPO** [8] is method that incorporates multiple negative samples in user preference data and generalizes pairwise DPO loss to a softmax ranking loss. **RosePO** [21] is a general framework that combines negative sampling strategies with personalized uncertainty to improve fairness, unbiasedness, and robustness. In our experiments, we sample rejected items based on item popularity distributions for comparison with RosePO. For RL-based methods, we implement PPO [34], using the frequency of each title in the training set as the reward.

As shown in Table 5, similar to how Flower outperforms BIGRec, preference alignment methods based on Flower generally

Table 6: Recommendation performance under different reward settings. The best results are highlighted in bold.

	CDs and Vinyl						Video Games						Movies and TV					
	NDCG↑	**HR↑**	**DGU↓**	**MGU↓**	**H↑**	**TTR↑**	**NDCG↑**	**HR↑**	**DGU↓**	**MGU↓**	**H↑**	**TTR↑**	**NDCG↑**	**HR↑**	**DGU↓**	**MGU↓**	**H↑**	**TTR↑**
$\log R_p$	**0.0712**	0.0880	0.071	0.019	7.885	0.013	0.0372	0.0539	0.102	0.020	7.651	0.005	0.0930	0.1123	0.084	**0.022**	**8.944**	**0.025**
$\log(R_p \cdot p_{ui})$	0.0700	**0.0898**	**0.060**	**0.018**	7.902	0.013	0.0366	0.0532	**0.092**	**0.018**	7.595	0.005	0.0905	0.1156	0.115	0.026	8.540	0.019
$\frac{\log R_p}{p_{ui}}$	0.0700	0.0885	0.075	0.021	**7.919**	**0.013**	**0.0543**	**0.0799**	0.108	0.023	**7.750**	**0.005**	**0.0953**	**0.1192**	**0.076**	0.027	8.808	0.023

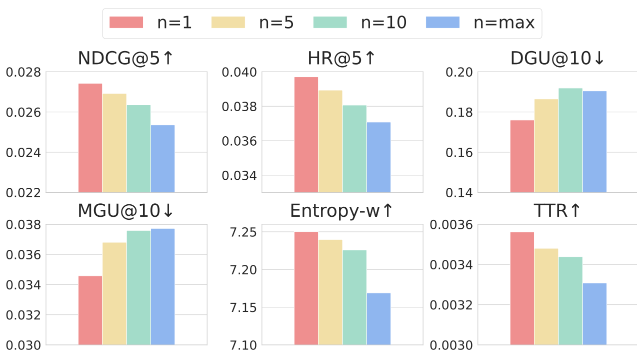


Figure 5: Impact of title partitioning granularities on the Video Games dataset.

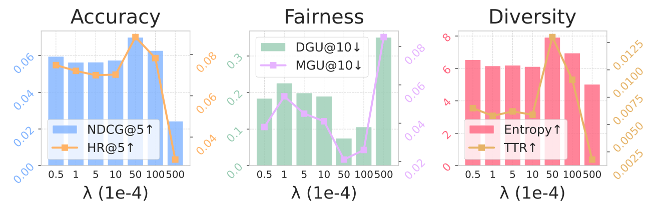


Figure 6: Performance with varying λ on the CDs dataset.

achieve better performance compared to their BIGRec-based counterparts. This demonstrates the potential of Flower-tuned methods for downstream applications.

Notably, RosePO excessively suppresses popular items, leading Flower's results—initially closer to the target distribution than BIGRec's—to deviate further from the target distribution compared to B_RosePO, resulting in poorer fairness for F_RosePO. Furthermore, due to the simplicity of its reward design, PPO fails to effectively optimize performance. However, Flower-based F_PPO still outperforms its BIGRec-based counterpart, B_PPO, highlighting the robustness of Flower as a reference policy.

4.5 Analysis of Key Factors in Flower (RQ4)

To comprehensively evaluate the contributions of Flower, we conduct ablation studies on three critical factors: reward formulation, the granularity of title segmentation, and the hyperparameter λ in Eq. (13). Each factor plays a distinct and essential role in shaping the model's performance: (1) reward formulation determines the target distribution for supervision, (2) partitioning titles into subtrajectories of varying granularity enables supervision at different intensities, and (3) λ controls the balance between the flow-guided loss and the SFT loss. By systematically varying these factors, we aim to understand their individual impacts and optimize their configurations to achieve the best trade-off among accuracy, fairness, and diversity.

4.5.1 Effects of Reward Setting. We investigate the effects of different process reward formulations. Specifically, we evaluate the results using the following three reward definitions:

- $\log R_p$: The original process reward term, as defined in Eq. (11).
- $\frac{\log R_p}{p_{ui}}$: A modified process reward described in Section 3.3.3, where the preference score p_{ui} is obtained from SASRec.
- $\log(R_p \cdot p_{ui})$: Another modified process reward, with the preference score p_{ui} also derived from SASRec.

The comparison of recommendation performance is shown in Table 6. For **accuracy**, $\frac{\log R_p}{p_{ui}}$ achieves the best performance on the Video Games and Movies datasets. Except for NDCG on the CDs dataset, all optimal accuracy is attained by methods incorporating SASRec scores, which validates the positive impact of personalized preferences on accuracy. For **fairness**, $\log(R_p \cdot p_{ui})$ demonstrates superior performance on the CDs and Video Games datasets compared to other methods, while $\log R_p$ performs well on only one metric of one dataset. This indicates that the effective integration of fairness and personalized preferences can simultaneously improve both accuracy and fairness. For **diversity**, $\frac{\log R_p}{p_{ui}}$ outperforms others on two datasets and ranks second only to $\log R_p$ on one dataset. Considering all three aspects, $\frac{\log R_p}{p_{ui}}$ exhibits the best comprehensive performance.

4.5.2 Impact of Supervision Granularity. Theoretically, convergence guarantees identical distributions regardless of how the states are partitioned. However, in practical scenarios, optimization is constrained to a finite number of steps, making it difficult to achieve the theoretical optimum. Therefore, ensuring sample efficiency becomes critical. To address this, we investigate how partitioning titles with varying granularities affects the effectiveness of fine-tuning.

We experiment with four granularities on the Video Games dataset: partitioning titles every 1 token (the original Flower method), 5 tokens, 10 tokens, or treating the entire title as a single state. During training, evaluations are performed every 20 steps, and the average values for each metric are calculated. As illustrated in Fig. 5, accuracy, fairness, and diversity improve progressively as the granularity becomes finer. The best performance is achieved when each token is treated as an individual partition, indicating that finer-grained constraints provide stronger supervision and lead to better training outcomes.

4.5.3 Performance Varying λ. The flow-guided loss and SFT loss can differ by 3 to 4 orders of magnitude, prompting us to explore the impact of λ across different magnitudes. We report the results on the CDs dataset. As shown in Fig. 6, as λ increases from left to right, the influence of the SFT loss diminishes while that of the flow-guided loss increases. Accuracy, fairness, and diversity generally exhibit a trend of first improving and then declining, with the best performance observed around $\lambda = 0.005$. When the weight of the flow-guided loss becomes excessively large, the lack of SFT loss constraints leads to a collapse in performance. Conversely, when $\lambda \leq 0.001$ (x-axis value of 10 or lower), the influence of the flow-guided loss becomes negligible, and the performance gradually converges to that of the SFT-only method.

5 Related Work

5.1 LLMs for Recommendation

Large Language Models (LLMs) have shown strong capabilities in text generation, reasoning, and generalization, motivating their use in personalized recommendation tasks. Supervised fine-tuning (SFT) has become a core approach for adapting LLMs to domain-specific recommendation data, significantly boosting performance [3, 7, 46]. To further align model outputs with user preferences and reduce bias, post-SFT training methods such as Direct Preference Optimization (DPO) have been proposed [8, 10].

Despite these advances, SFT-based models often suffer from popularity bias, leading to filter bubbles and degraded user experience [11, 12]. This is largely due to overfitting introduced by the cross-entropy loss used in fine-tuning, which biases the model toward frequently occurring items in the training set. While recent efforts represent items using unique identifier sequences [33, 40], they remain within the CE loss framework and inherit its limitations.

In this work, we propose a novel fine-tuning paradigm that addresses the shortcomings of SFT and promotes more balanced, personalized recommendations.

5.2 Process Supervised

To better align LLMs with human preferences or enhance reasoning ability, a post-SFT alignment step is often introduced. Techniques like RLHF and DPO [32] apply outcome-level supervision by evaluating model responses holistically. However, such supervision is coarse-grained, making it hard to interpret or guide intermediate reasoning steps [36].

Recently, process supervision has gained attention for its ability to provide step-level feedback using Process Reward Models (PRMs) [19, 47]. Compared to outcome-level methods, process supervision offers more interpretable and direct optimization signals, leading to improved reasoning quality and alignment. Most existing approaches require learning parametric PRMs—e.g., modeling step-wise correctness probabilities [37, 39]—which is often impractical in recommendation tasks due to sparse user feedback. In this work, we adopt simple yet effective heuristic rewards to approximate step-level quality. This design avoids additional parameter learning and enables efficient process supervision in data-scarce settings.

5.3 Applications of GFlowNets

A key advantage of GFlowNets lies in their ability to sample diverse solutions while maintaining proportionality to the reward. Furthermore, GFlowNets demonstrate strong generalization capabilities, allowing them to handle states not encountered during training [5, 6, 30, 44]. These properties make GFlowNets particularly well-suited for tasks that require exploring a wide solution space, such as molecular design [14, 48] and structured prediction [31]. In recommendation systems, GFlowNets have also shown significant potential. For example, Liu et al. [25] use GFlowNets to introduce diversity while maintaining quality in listwise recommendations, and Liu et al. [26] apply GFlowNets to enhance user retention while fostering exploration.

Recently, GFlowNets have been employed in fine-tuning LLMs for specific tasks. For example, Hu et al. [13] fine-tune LLMs using GFlowNets to achieve diversity in tasks such as sentence infilling, chain-of-thought reasoning, and problem-solving with external tools. Lee et al. [18] adopt GFlowNets to fine-tune LLM-based attacker models, enabling the generation of diverse and effective attack prompts for Red-teaming. Similarly, Yu et al. [43] apply GFlowNets to train LLMs for puzzle-solving tasks, including BlocksWorld and Game24. In this work, we are the first to use GFlowNets to address the limitations of SFT in LLM-based next-item recommendation tasks.

6 Conclusion

This work addresses key limitations of supervised fine-tuning (SFT) in LLM-based recommendation systems, notably limited diversity and amplified popularity bias, which hinder accuracy, fairness, and personalization. These issues largely arise from the overfitting nature of cross-entropy loss used in SFT. To overcome these challenges, we propose **Flower**, a novel fine-tuning paradigm based on generative flow networks (GFlowNets) and process-level supervision. Flower frames recommendation as a flow network, propagating item-level rewards—derived from item frequencies—to token-level supervision. This aligns token generation with reward distributions, promoting balanced, diverse, and fair recommendations. Experiments on three real-world sequential recommendation datasets show that Flower outperforms SFT in accuracy, diversity, and fairness. Furthermore, applying alignment methods after Flower fine-tuning yields better results than applying them on top of standard SFT models.

This study emphasizes the transformative potential of integrating diversity-seeking mechanisms into LLM-based recommendation systems. By introducing the Flow-guided fine-tuning paradigm, we address core limitations of conventional approaches, bridging the gap between accuracy, fairness, and personalization. Beyond its immediate applications, this framework lays the foundation for future innovations, including extending its applicability to diverse recommendation contexts and refining reward design to better capture nuanced user preferences and domain-specific goals.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (62402470, U21B2026, 62121002, U24B20180), Anhui Provincial Natural Science Foundation (2408085QF189), the Fundamental Research Funds for the Central Universities of China (WK2100000053), and the advanced computing resources provided by the Supercomputing Center of the USTC.

References

[1] Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. 2024. A general theoretical paradigm to understand learning from human preferences *(AISTATS '24)*.

[2] Zhuoxi Bai, Ning Wu, Fengyu Cai, Xinyi Zhu, and Yun Xiong. 2024. Aligning Large Language Model with Direct Multi-Preference Optimization for Recommendation *(CIKM '24)*. 76–86.

[3] Keqin Bao, Jizhi Zhang, Wenjie Wang, Yang Zhang, Zhengyi Yang, Yanchen Luo, Chong Chen, Fuli Feng, and Qi Tian. 2025. A Bi-Step Grounding Paradigm for Large Language Models in Recommendation Systems. *ACM Transactions on Recommender Systems (TORS)* (2025).

[4] Keqin Bao, Jizhi Zhang, Yang Zhang, Xinyue Huo, Chong Chen, and Fuli Feng. 2024. Decoding Matters: Addressing Amplification Bias and Homogeneity Issue for LLM-based Recommendation. *EMNLP '24* (2024).

[5] Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. 2024. Flow network based generative models for non-iterative diverse candidate generation *(NIPS '21)*.

[6] Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J. Hu, Mo Tiwari, and Emmanuel Bengio. 2024. GFlowNet Foundations. *Journal of Machine Learning Research (JMLR)* 24, 1 (2024).

[7] Jiaju Chen, Chongming Gao, Shuai Yuan, Shuchang Liu, Qingpeng Cai, and Peng Jiang. 2025. DLCRec: A Novel Approach for Managing Diversity in LLM-Based Recommender Systems. *WSDM* (2025).

[8] Yuxin Chen, Junfei Tan, An Zhang, Zhengyi Yang, Leheng Sheng, Enzhi Zhang, Xiang Wang, and Tat-Seng Chua. 2024. On Softmax Direct Preference Optimization for Recommendation. *NeurIPS '24* 37 (2024), 27463–27489.

[9] Sunhao Dai, Chen Xu, Shicheng Xu, Liang Pang, Zhenhua Dong, and Jun Xu. 2024. Bias and Unfairness in Information Retrieval Systems: New Challenges in the LLM Era *(KDD '24)*. 6437–6447.

[10] Chongming Gao, Ruijun Chen, Shuai Yuan, Kexin Huang, Yuanqing Yu, and Xiangnan He. 2025. SPRec: Self-Play to Debias LLM-based Recommendation *(WWW '25)*. 5075–5084.

[11] Chongming Gao, Kexin Huang, Jiawei Chen, Yuan Zhang, Biao Li, Peng Jiang, Shiqi Wang, Zhong Zhang, and Xiangnan He. 2023. Alleviating Matthew Effect of Offline Reinforcement Learning in Interactive Recommendation *(SIGIR '23)*.

[12] Chongming Gao, Shiqi Wang, Shijun Li, Jiawei Chen, Xiangnan He, Wenqiang Lei, Biao Li, Yuan Zhang, and Peng Jiang. 2023. CIRS: Bursting Filter Bubbles by Counterfactual Interactive Recommender System. *ACM Transactions on Information Systems (TOIS)* 42, 1 (2023).

[13] Edward J Hu, Moksh Jain, Eric Elmoznino, Younesse Kaddar, Guillaume Lajoie, Yoshua Bengio, and Nikolay Malkin. 2024. Amortizing intractable inference in large language models *(ICLR '24)*.

[14] Moksh Jain, Emmanuel Bengio, Alex Hernandez-Garcia, Jarrid Rector-Brooks, Bonaventure FP Dossou, Chanakya Ajit Ekbote, Jie Fu, Tianyu Zhang, Michael Kilgour, Dinghuai Zhang, et al. 2022. Biological Sequence Design with GFlowNets *(ICML '22)*. 9786–9801.

[15] Meng Jiang, Keqin Bao, Jizhi Zhang, Wenjie Wang, Zhengyi Yang, Fuli Feng, and Xiangnan He. 2024. Item-side Fairness of Large Language Model-based Recommendation System *(WWW '24)*. 4717–4726.

[16] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation *(ICDM '18)*. 197–206.

[17] Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. 2024. Understanding the Effects of RLHF on LLM Generalisation and Diversity.

[18] Seanie Lee, Minsu Kim, Lynn Cherif, David Dobre, Juho Lee, Sung Ju Hwang, Kenji Kawaguchi, Gauthier Gidel, Yoshua Bengio, Nikolay Malkin, et al. 2025. Learning Diverse Attacks on Large Language Models for Robust Red-Teaming and Safety Tuning *(ICLR '25)*.

[19] Wendi Li and Yixuan Li. 2025. Process Reward Model with Q-value Rankings. *ICLR '25* (2025).

[20] Ziniu Li, Congliang Chen, Tian Xu, Zeyu Qin, Jiancong Xiao, Ruoyu Sun, and Zhi-Quan Luo. 2024. Entropic Distribution Matching in Supervised Fine-tuning of LLMs: Less Overfitting and Better Diversity *(FITML '24)*.

[21] Jiayi Liao, Xiangnan He, Ruobing Xie, Jiancan Wu, Yancheng Yuan, Xingwu Sun, Zhanhui Kang, and Xiang Wang. 2024. RosePO: Aligning LLM-based Recommenders with Human Values. *arXiv preprint arXiv:2410.12519* (2024).

[22] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's Verify Step by Step *(ICLR '24)*.

[23] Jianghao Lin, Xinyi Dai, Yunjia Xi, Weiwen Liu, Bo Chen, Hao Zhang, Yong Liu, Chuhan Wu, Xiangyang Li, Chenxu Zhu, Huifeng Guo, Yong Yu, Ruiming Tang, and Weinan Zhang. 2024. How Can Recommender Systems Benefit from Large Language Models: A Survey. *ACM Trans. Inf. Syst.* (2024).

[24] Xinyu Lin, Wenjie Wang, Yongqi Li, Shuo Yang, Fuli Feng, Yinwei Wei, and Tat-Seng Chua. 2024. Data-efficient Fine-tuning for LLM-based Recommendation *(SIGIR '24)*. 365–374.

[25] Shuchang Liu, Qingpeng Cai, Zhankui He, Bowen Sun, Julian McAuley, Dong Zheng, Peng Jiang, and Kun Gai. 2023. Generative Flow Network for Listwise Recommendation *(KDD '23)*. 1524–1534.

[26] Ziru Liu, Shuchang Liu, Bin Yang, Zhenghai Xue, Qingpeng Cai, Xiangyu Zhao, Zijian Zhang, Lantao Hu, Han Li, and Peng Jiang. 2024. Modeling User Retention through Generative Flow Networks *(KDD '24)*. 5497–5508.

[27] Wensheng Lu, Jianxun Lian, Wei Zhang, Guanghua Li, Mingyang Zhou, Hao Liao, and Xing Xie. 2024. Aligning Large Language Models for Controllable Recommendations *(ACL '24)*. 8159–8172.

[28] Qiyao Ma, Xubin Ren, and Chao Huang. 2024. XRec: Large Language Models for Explainable Recommendation *(EMNLP 24 Findings)*. 391–402.

[29] Kanika Madan, Jarrid Rector-Brooks, Maksym Korablyov, Emmanuel Bengio, Moksh Jain, Andrei Cristian Nica, Tom Bosc, Yoshua Bengio, and Nikolay Malkin. 2023. Learning gflownets from partial episodes for improved convergence and stability *(ICML '23)*. 23467–23483.

[30] Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. 2022. Trajectory balance: improved credit assignment in GFlowNets *(NeurIPS '22)*.

[31] Nikolay Malkin, Salem Lahlou, Tristan Deleu, Xu Ji, Edward J Hu, Katie E Everett, Dinghuai Zhang, and Yoshua Bengio. 2023. GFlowNets and variational inference *(ICLR '23)*.

[32] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. *NeurIPS '23* 36 (2023).

[33] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Keshavan, Trung Vu, Lukasz Heidt, Lichan Hong, Yi Tay, Vinh Q. Tran, Jonah Samost, Maciej Kula, Ed H. Chi, and Maheswaran Sathiamoorthy. 2024. Recommender systems with generative retrieval *(NeurIPS '23)*.

[34] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[35] Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. 2025. Rewarding Progress: Scaling Automated Process Verifiers for LLM Reasoning *(ICLR '25)*.

[36] Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. 2025. Rewarding progress: Scaling automated process verifiers for LLM reasoning. *ICLR '25* (2025).

[37] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300* (2024).

[38] Chenglei Shen, Xiao Zhang, Teng Shi, Changshuo Zhang, Guofu Xie, and Jun Xu. 2024. A survey of controllable learning: Methods and applications in information retrieval. *arXiv preprint arXiv:2407.06083* (2024).

[39] Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024. Math-shepherd: Verify and reinforce llms step-by-step without human annotations *(ACL '24)*. 9426–9439.

[40] Wenjie Wang, Honghui Bao, Xinyu Lin, Jizhi Zhang, Yongqi Li, Fuli Feng, See-Kiong Ng, and Tat-Seng Chua. 2024. Learnable Item Tokenization for Generative Recommendation *(CIKM '24)*. 2400–2409.

[41] Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, Hui Xiong, and Enhong Chen. 2024. A Survey on Large Language Models for Recommendation. *World Wide Web* 27, 5 (2024).

[42] Mengyuan Yang, Mengying Zhu, Yan Wang, Linxun Chen, Yilei Zhao, Xiuyuan Wang, Bing Han, Xiaolin Zheng, and Jianwei Yin. 2024. Fine-Tuning Large Language Model Based Explainable Recommendation with Explainable Quality Reward *(AAAI '24, 8)*. 9250–9259.

[43] Fangxu Yu, Lai Jiang, Haoqiang Kang, Shibo Hao, and Lianhui Qin. 2024. Flow of reasoning: Efficient training of llm policy with divergent thinking. *arXiv preprint arXiv:2406.05673* (2024).

[44] Dinghuai Zhang, Nikolay Malkin, Zhen Liu, Alexandra Volokhova, Aaron Courville, and Yoshua Bengio. 2022. Generative flow networks for discrete probabilistic modeling *(ICML '22)*. 26412–26428.

[45] Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024. ReST-MCTS*: LLM Self-Training via Process Reward Guided Tree Search *(NeurIPS '24)*.

[46] Yang Zhang, Fuli Feng, Jizhi Zhang, Keqin Bao, Qifan Wang, and Xiangnan He. 2025. Collm: Integrating Collaborative Embeddings into Large Language Models for Recommendation. *TKDE* (2025).

[47] Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. The lessons of developing process reward models in mathematical reasoning. *arXiv preprint arXiv:2501.07301* (2025).

[48] Yiheng Zhu, Jialu Wu, Chaowen Hu, Jiahuan Yan, Chang-Yu Hsieh, Tingjun Hou, and Jian Wu. 2024. Sample-efficient multi-objective molecular optimization with GFlowNets *(NeurIPS '23)*.