# RLHF-based Optimization of Recommendations Using Large Language Models

**Fanyu You** fyou@usc.edu

**Rishabh Agrawal** rishabha@usc.edu

**Mingxi Wang** mingxiwa@usc.edu

**Rutik Rajendra Yerunkar** ryerunka@usc.edu

**Yash Gawankar** gawankar@usc.edu

## Abstract

This study explores the optimization of recommendation systems with Large Language Models (LLMs) using two distinct methodologies: Reinforcement Learning from Human Feedback (RLHF) and Direct Preference Optimization (DPO). Separate pipelines were developed to fine-tune LLMs for inferred user preferences. Using MovieLens datasets, we show that supervised fine-tuning (SFT) and preference tuning improve LLM-based recommendations. These results highlight the potential and limitations of LLM-based systems for recommendation tasks. Our code is available at https://github.com/fanyuy2/LLM-RL4Rec/.

## 1 Introduction

Recommendation systems are essential to deliver personalized content and address user preferences effectively. Traditional methods like collaborative filtering and content-based recommendations often struggle with sparse data and dynamic user behaviors, limiting their scalability and adaptability. Advances in artificial intelligence, particularly reinforcement learning and large language models (LLMs), offer promising solutions to these challenges.

LLMs excel in understanding and generating complex information, making them well-suited for recommendation tasks. Reinforcement Learning from Human Feedback (RLHF) enhances these models by iteratively refining recommendations based on user feedback. However, RLHF alone may falter with noisy or sparse feedback signals.

To address these limitations, we developed two separate optimization pipelines: one using RLHF and another with Direct Preference Optimization (DPO). The RLHF pipeline iteratively fine-tunes LLMs using reward feedback, while the DPO pipeline optimizes preference-based ranking for inferred user preferences. These independent approaches enable a comprehensive evaluation of their strengths and weaknesses.

Using MovieLens datasets (Harper and Konstan, 2015) as benchmarks, our results show that supervised fine-tuning (SFT) and preference tuning significantly improve LLM-based recommendations over initial baselines.

Key contributions of this study include: 1. Development of RLHF and DPO pipelines for LLM-based recommendation systems. 2. Evaluation of the impact of SFT and preference tuning on model performance using various metrics.

This study explores the adaptation of LLMs for recommendation tasks. Figure 1 presents an overview of our RLHF-based approach, while the other approach employed in this work is based on DPO.

## 2 Related Works

Traditional recommendation models such as collaborative filtering (Sarwar et al., 2001) and content-based methods (Chu and Park, 2009) have been the foundation for personalized recommendations. However, these approaches often struggle to generalize to new or sparse data and have limited ability to capture dynamic user preferences. Recently, large language models (LLMs) have demonstrated significant potential to address these challenges, thanks to their advanced contextual understanding and natural language processing capabilities (Bao et al., 2023). At the same time, reinforcement learning (RL) has gained traction as a powerful tool to optimize long-term user satisfaction by continuously adapting to evolving user behaviors (Zheng et al., 2018). While both LLMs and RL have individually proven effective in enhancing recommendation systems, the synergy between them remains under-explored. Existing efforts largely focus on using RL to automate prompt engineering (Wang et al., 2024b) or leveraging LLMs as the

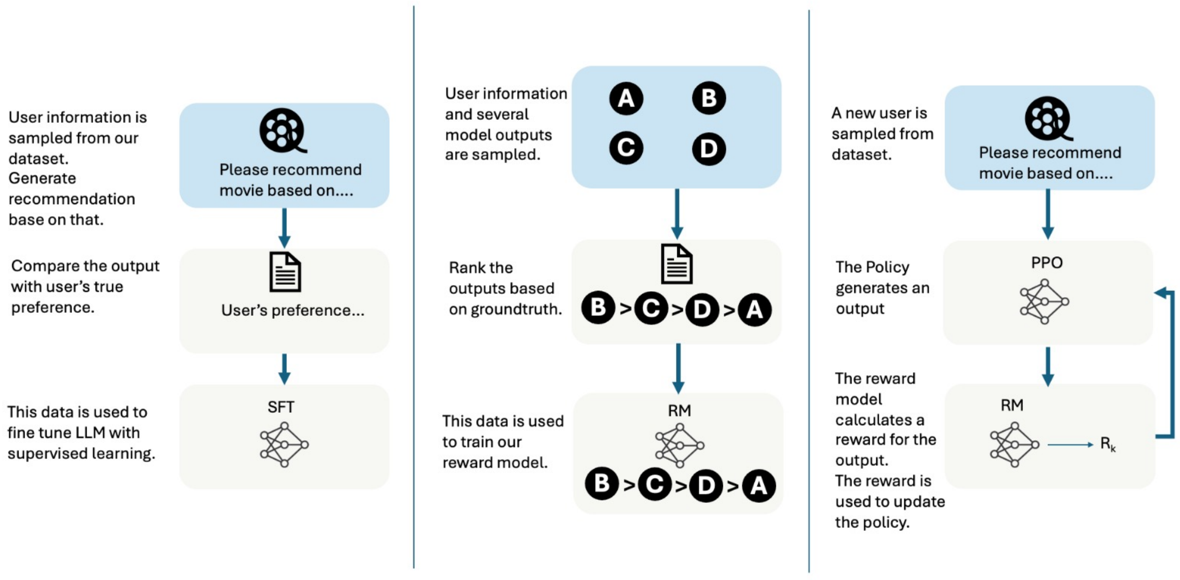
User information is sampled from our dataset. Generate recommendation base on that.
Please recommend movie based on....
Compare the output with user's true preference.
User's preference...
This data is used to fine tune LLM with supervised learning.
SFT
User information and several model outputs are sampled.
A
B
C
D
Rank the outputs based on groundtruth.
B > C > D > A
This data is used to train our reward model.
RM
B > C > D > A
A new user is sampled from dataset.
Please recommend movie based on....
The Policy generates an output
PPO
The reward model calculates a reward for the output. The reward is used to update the policy.
RM
R_k

Figure 1: An overview of our RLHF approach.

environment to improve state representation and reward modeling in RL-based recommender systems (Wang et al., 2024a). In contrast, our model introduces a Reinforcement Learning from Human Feedback (RLHF) framework specifically designed to optimize large language models for recommendation tasks.

## 3 Methodologies

### 3.1 Data Preparation

The data preparation pipeline processes and enriches user, movie, and interaction data from the MovieLens dataset to support the recommendation task. User profiles are constructed by incorporating demographic details such as age, gender, occupation, and geographic location, with the latter inferred from ZIP codes (Gheem and Verkhovskiy, 2020). The dataset includes detailed user information, such as demographics and a comprehensive history of each user's rated movies, along with corresponding movie ratings. However, timestamps are provided only for when the user rated a movie, not when they watched it. For our setup, we assume the rating timestamp as the time the user watched the movie.

A custom method identifies valid interaction windows by balancing positive and negative user ratings based on predefined thresholds. These interaction windows define the user's historical context, categorizing movies into "liked" and "disliked" groups based on ratings above or below the thresholds, respectively.

To define the recommendation task, we iterate over all users and identify valid timestamps where the context window contains at least a minimum number ($k$) of liked or disliked movies. For each valid user, a random timestamp is sampled, serving as the endpoint of the fixed-length context window. Movies watched within this window are categorized into liked or disliked groups based on predefined thresholds and used to generate prompts for the user at the sampled timestamp. A predefined prompt template incorporates the user's historical data, which is passed to the LLM to produce a list of top-$k$ recommended movies, where $k$ is a configurable parameter. Figure 2 illustrates an example of the prompt.

The preprocessed data and generated prompts facilitate downstream tasks such as reward model training, policy optimization, and recommendation evaluation. To handle cases where the LLM recommends movies not present in the dataset, a final mapping step is performed, which we discuss in the following section.

### 3.2 Item Space Mapping

A key challenge in using LLMs for recommendation systems is aligning their non-deterministic output with the structured item space of the dataset. LLM-generated recommendations often vary in formatting, capitalization, token ordering, or phrasing. For instance, LLMs might output "The Shawshank

I am a male, aged 25, from Los Angeles, CA, working as executive.I have previously watched and liked the movies: ['Nightmare Before Christmas, The (1993) (childrens, comedy, musical)', '2001: A Space Odyssey (1968) (drama, mystery, sci_fi, thriller)', 'Deer Hunter, The (1978) (drama, war)', 'Bridge on the River Kwai, The (1957) (drama, war)', 'North by Northwest (1959) (comedy, thriller)'].\nI have watched and HATED the movies: ['Mother (1996) (comedy)', 'Starship Troopers (1997) (action, adventure, sci_fi, war)', 'Air Force One (1997) (action, thriller)']Please provide recommendations for movies released before April 22nd, 1998, based on my history.\Based on my profile, recommend the top 5 movies I am most likely to watch next.\Please provide the output in a list of strings format, containing only the movie titles.\Make sure to strictly adhere to the output format given below. Strictly do not generate any additional information other than the movie names.\Format: ['movie_name', 'movie_name', ... 'movie_name'] Make sure to limit the recommendations to movies available in the MovieLens dataset.

Figure 2: An example of prompt.

Redemption" instead of the exact movie title as in the dataset - "Shawshank Redemption, The (1994)" or omit details like the release year entirely, leading to misalignment with ground truth entries.

To address this, we use a fuzzy matching technique based on the *fuzz.WRatio* (Leite and Vaconcelos, 2024) scoring function, which incorporates Levenshtein distance—a measure of the minimum character edits needed to match two strings. The method combines multiple scoring techniques to resolve discrepancies:

- **Simple Ratio:** Direct comparison using Levenshtein distance, e.g., matching "The Shawshank Redemption" to "Shawshank Redemption, The (1994)."
- **Partial Ratio:** Matches the best substring, e.g., "Star Wars: A New Hope" with "Star Wars (1977)."
- **Token Sort Ratio:** Matches strings after sorting tokens alphabetically, e.g., "Dead Man's Chest Pirates of the Caribbean" with "Pirates of the Caribbean: Dead Man's Chest (2006)."
- **Token Set Ratio:** Matches unique tokens, ignoring redundancies, e.g., "Fast and Furious" with "The Fast and the Furious (2001)."
- **Weighted Adjustments:** Adjusts scores for string length and complexity, e.g., "Avatar" compared to "Avatar: The Way of Water (2022)."

This approach ensures robust handling of formatting variations and token mismatches. For each LLM-generated title, similarity scores are computed against all dataset entries, and the best match is selected to align recommendations with the dataset effectively.

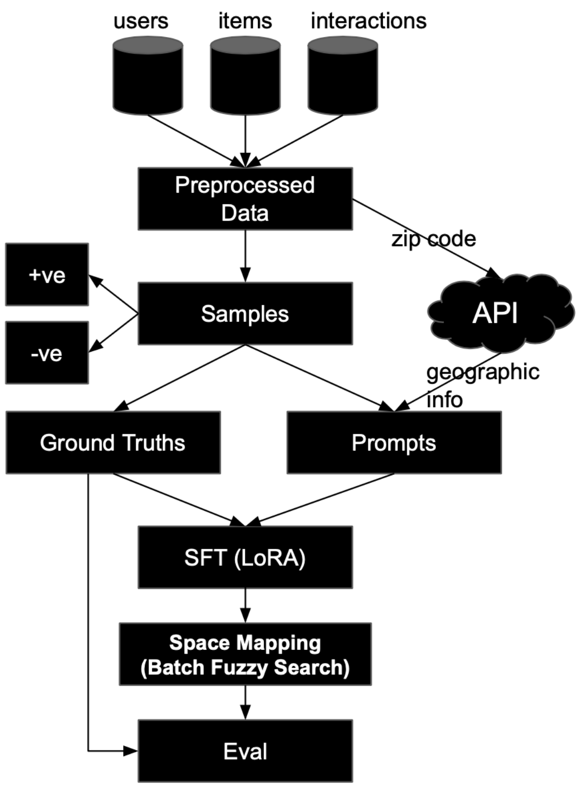

Figure 3: A flowchart of SFT pipeline.

## 3.3 Supervised Fine-Tuning (SFT)

SFT is a critical initial step in our project to optimize the recommendation system. In this stage, we adopt a pre-trained LLM to generate recommendations tailored to user preferences by fine-tuning it with labeled data.

The primary goal of SFT is to familiarize the pre-trained LLM with the specific task structure of generating top-k movie recommendations given a user's context. By training in prompt-response pairs, the model learns to produce outputs aligned with user preferences based on historical interactions and predefined criteria as well as domain knowledge. This significantly improves the baseline performance, forming the foundation for further refinement using preference tuning.

For labeled data, we used prompt and ground truth pairs obtained after data preparation, where the prompt provides the context and the ground truth serves as the corresponding response. Figure 3 illustrates the flowchart of our SFT pipeline.

For SFT, we used the "meta-llama/Llama-3.2-1B-Instruct" model as our pretrained base model, chosen for its strong contextual understanding and ability to handle natural language prompts effectively.

The training process focused on maximizing the probability of generating target recommendations that align with labeled responses. This employs cross-entropy loss function to encourage the model to generate outputs closely matching the labeled data while penalizing incorrect predictions.

While SFT established a strong baseline by enabling the pretrained LLM to generate recommendations based on historical data, it revealed limitations in adapting to structural responses and nuanced preferences. This underscored the need for further refinement through preference tuning, which builds on the foundation laid during SFT.

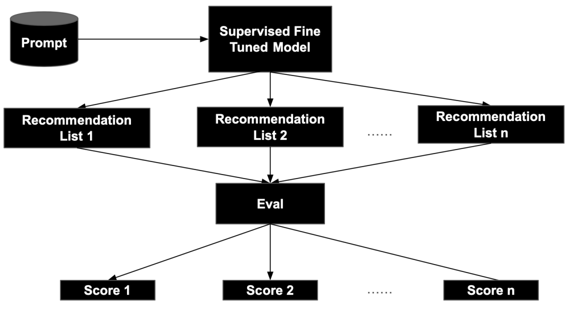

Figure 4: A flowchart for preference data creation.

### 3.4 Preference Data

To enable preference-based optimization, a pairwise preference dataset was constructed by generating $n$ distinct responses for each prompt and evaluating their quality against the ground truth using NDCG@k. The NDCG metric was chosen to prioritize the ranking of recommendations. The model outputs were mapped to the item space using fuzzy matching, and pairwise comparisons $\binom{n}{2}$ were conducted based on their NDCG@k scores. Preferences were assigned to the response with the higher score, forming a structured dataset of ranked preferences for downstream optimization through Direct Preference Optimization (DPO) and Proximal Policy Optimization (PPO). A visual representation for this process is shown in figure 4.

### 3.5 RLHF with PPO

We use Proximal Policy Optimization (PPO) to fine-tune the model with human feedback. Let $\pi_\theta(a|s)$ represent the policy that produces an action $a$ given a state $s$. Let $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ represent the probability ratio between the current policy and the old policy. Let $A_t$ represent the advantage. Let $\epsilon$ represent a small threshold, for example $0.1$. PPO updates the parameters $\theta$ by maximizing the following objectives:

$$L^{PPO}(\theta) = \mathbb{E}_t\Big[\min\big(r_t(\theta)A_t, \text{clip}\big(r_t(\theta), 1-\epsilon, 1+\epsilon\big)A_t\big)\Big]. \quad (1)$$

This objective function tries to improve the policy while avoiding large changes. It uses a clipped probability ratio to keep updates stable. The advantage $A_t$ represents how much better an action is compared to an average action. We estimate $A_t$ with standard methods like Generalized Advantage Estimation. The reward model provides a scalar score for each recommended item. This score reflects user feedback. The model updates its parameters to increase expected reward over many iterations. PPO steps allow the model to produce recommendations that align with observed user preferences. The process repeats until it reaches a stable policy that gives better recommendations than the initial supervised baseline.

### 3.6 Direct Preference Optimization (DPO)

Direct Preference Optimization (DPO) is a method for fine-tuning language models to align with preferences through a direct, supervised learning framework. Unlike Reinforcement Learning with Human Feedback (RLHF), which relies on a reward model and reinforcement learning algorithms, DPO directly optimizes a language model's parameters using preference data.

Given a dataset of response pairs $(x, y_{\text{pref}}, y_{\text{npref}})$, where $x$ is the input and $y_{\text{pref}}$ and $y_{\text{npref}}$ are the responses annotated with preference, where $y_{\text{pref}}$ is the preferred one whereas $y_{\text{npref}}$ is the dispreferred one, DPO aims to optimize the model $\pi_\theta$ by maximizing the likelihood of the preferred responses.

The optimization is based on the following objective derived from a probabilistic preference model:

$$L^{DPO}(\theta) = \mathbb{E}_{(x,y_{\text{pref}},y_{\text{npref}})}\Big[\log\sigma\Big(\beta\big(\log\pi_\theta(y_{\text{pref}}|x) - \log\pi_\theta(y_{\text{npref}}|x)\big)\Big)\Big],$$

where $\sigma(z) = \frac{1}{1+e^{-z}}$ is the sigmoid function, and $\beta > 0$ is a temperature-like scaling parameter controlling the sharpness of the preference decision.

The core idea is to directly fine-tune the model $\pi_\theta$ such that it assigns higher probabilities to preferred responses compared to dispreferred ones. This approach circumvents the complexities of reward modeling and reinforcement learning, leading to a more stable and computationally efficient training process.

## 4 Experiments

### 4.1 Datasets

We used MovieLens 100K and 1M datasets for training, validation, and testing in our experiments. These datasets contain detailed user profiles, including demographic information, viewing history, and movie ratings. To ensure consistency, the data was split into training, validation, and testing sets with a 70%-15%-15% ratio. The split was performed randomly at the user level.

The datasets were preprocessed by categorizing movie ratings into positive (liked) or negative (disliked) classes using predefined thresholds: ratings less or equal to 2 were labeled as disliked, while ratings more than or equal to 3 were labeled as liked. User interactions were subsequently transformed into natural language prompts, which served as input for the recommendation model.

### 4.2 Baseline Model

We compared our proposed model to the “meta-llama/Llama-3.2-1B-Instruct” model, a pretrained LLM known for its strong natural language processing capabilities and ability to handle recommendation tasks with minimal fine-tuning. This model serves as a benchmark to highlight the improvements introduced by incorporating SFT and preference tuning.

### 4.3 Experimental Setups

The experiments were conducted in the following environment:

- **Hardware**: 16GB NVIDIA GPU, and 128GB RAM.
- **Software**: Hugging Face Transformers, and Python 3.12.
- **Framework**: The Hugging Face TRL API (von Werra et al., 2020) was used for SFT, DPO, and RLHF, while the PEFT API (Mangrulkar et al., 2022) was utilized for LoRA.

We trained the SFT model using the following configuration: the optimizer was AdamW (Loshchilov and Hutter, 2019), with a learning rate of $5 \times 10^{-5}$, a batch size of 8, and 3 training epochs. For the LoRA configuration, we set the LoRA alpha to 128, the LoRA dropout to 0.05, the rank $(r)$ to 256, and the bias parameter to "none," while applying LoRA to all linear layers (`target_modules="all-linear"`).

We trained DPO with the following configuration: AdamW optimizer, a batch size of 1, 2 training epochs, 1 gradient accumulation step, and fp16 enabled $True$. Logging was performed every 10 steps. The LoRA-specific configuration included r equals to 16, LoRA alpha set to 16, target modules as $all - linear$, and a LoRA dropout rate of 0.05.

We trained the RLHF pipeline using the following configuration. For the LoRA setup, we used a rank $(r)$ of 8, a LoRA alpha of 32, and a dropout rate of 0.1. The reward model was trained with a learning rate of $5 \times 10^{-5}$, a batch size of 1 per device, and 1 training epoch. For the PPO configuration, we set the learning rate to $1 \times 10^{-5}$ and used a batch size of 16.

Further details of our configuration can be found on our GitHub repository.

### 4.4 Evaluation Metrics

The performance of our recommendation system was evaluated using the following metrics:

**Normalized Discounted Cumulative Gain (NDCG@k)** evaluates the ranking quality of recommendations, emphasizing higher-ranked items:

$$\text{NDCG@k} = \frac{\text{DCG@k}}{\text{IDCG@k}}.$$

The formula for **DCG@k** is given as:

$$\text{DCG@k} = \sum_{i=1}^{k} \frac{\text{rel}_i}{\log_2(i+1)},$$

Here:

- $k$ is the rank position until which the gain is considered.
- $\text{rel}_i$ is the relevance score of the item at position $i$.

**IDCG@k** is the ideal DCG when documents are perfectly ranked.

**Precision@k** is the proportion of recommended movies within the top-k that are relevant. It is calculated by the number of relevant documents retrieved at rank $k$ divided by the total number of documents retrieved at rank $k$.

$$\text{Precision@k} = \frac{1}{k}\sum_{i=1}^{k}\text{Rel}_i$$

Here:

- $k$ is the rank position.
- $\text{Rel}_i$ is an indicator function that equals 1 if the document at position $i$ is relevant, and 0 otherwise.

**Recall@k** is the proportion of relevant movies retrieved within the top-k recommendations. It is calculated by the number of relevant documents retrieved at rank $k$ divided by the total number of relevant documents.

$$\text{Recall@k} = \frac{\sum_{i=1}^{k}\text{Rel}_i}{R}$$

Here:

- $k$ is the rank position.
- $\text{Rel}_i$ is an indicator function that equals 1 if the document at position $i$ is relevant, and 0 otherwise.
- $R$ is the total number of relevant documents in the dataset.

The evaluation was performed on the test set, and metrics were computed for $k = 5$.

## 5 Results

The performance of our recommendation pipelines—Base Model, SFT Model, RLHF Model, and DPO Model—using NDCG@k, Precision@k, and Recall@k metrics. The results are summarized in Table 1 for Movielens $100k$ dataset.

| **Metric** | **Base** | **SFT** | **DPO** | **RLHF** |
|---|---|---|---|---|
| NDCG@k | 0.624 | 1.687 | 1.726 | 1.821 |
| Precision@k | 0.606 | 1.818 | 1.890 | 1.818 |
| Recall@k | 0.512 | 1.441 | 1.497 | 1.489 |

Table 1: Performance comparison across Base Model, SFT, DPO, and RLHF pipelines (in %) over Movielens 100k dataset.

| **Stage** | **Sample Recommendations** |
|---|---|
| Ground Truth | [“Devil’s Advocate, The (1997)”, ’Jaws (1975)’, ’Alien (1979)’, ’Raiders of the Lost Ark (1981)’, ’Psycho (1960)’] |
| Pre-SFT | [’For Richer or Poorer (comedy)’, ’Half Baked (comedy)’, ’Everyone Says I Love You (comedy, musical, romance)’, ’The Saint (action, romance, thriller)’, ’Rocket Man (comedy)’] |
| Post-SFT | [’Sleeper (1973)’, ’Bridge on the River Kwai, The (1957)’, ’English Patient, The (1996)’, ’Raiders of the Lost Ark’, “Dead Poets Society”] |
| Post-DPO | [’Bridge on the River Kwai, The (1957)’, ’Alien (1979)’, “Devil’s Advocate, The (1997)”, ’Raiders of the Lost Ark (1981)’, ’Jaws (1975)’] |
| Post-RLHF | [“Devil’s Advocate, The (1997)”, ’Jaws (1975)’, ’Bridge on the River Kwai, The (1957)’, ’Alien (1979)’„ ’Raiders of the Lost Ark (1981)’] |

Table 2: Sample recommendations after different stages of training.

Sample recommendations after each training stage are shown in Table 2 to highlight qualitative changes.

Next, we expanded our experiments to the larger Movielens $1M$ dataset. The results are summarized in Table 3. Note that this table does not include results for DPO or RLHF, as resource constraints prevented us from training these models on this dataset.

| **Metric** | **Base** | **SFT** |
|---|---|---|
| NDCG@k | 0.213 | 2.341 |
| Precision@k | 0.254 | 2.199 |
| Recall@k | 0.228 | 1.812 |

Table 3: Performance comparison across Base Model and SFT over Movielens 1M dataset.

## 6 Discussion

The experimental results provide valuable insights into the performance and behavior of LLM-based recommendation systems. Key observations are as follows:

SFT establishes a strong foundation by aligning the pretrained LLM's outputs with instructions through task-specific prompt-response pairs. The substantial 170%-200% improvements in NDCG@k, Precision@k, and Recall@k over the base model highlights the effectiveness of this approach in adapting LLMs to recommendation tasks. SFT enables the model to produce relevant and context-aware recommendations, addressing the limitations of the base model.

Building on this, Direct Preference Optimization (DPO) further refines the recommendation quality by directly optimizing on pairwise user preferences. Unlike RLHF with PPO, DPO avoids reliance on a separate reward model, simplifying the optimization process. While the 2.3%-3.9% incremental gains over SFT appear modest, they demonstrate the importance of fine-grained preference alignment in improving ranking performance. Together, SFT and DPO showcase a complementary approach: SFT provides large-scale improvements by adapting the model to structured data, while DPO fine-tunes the model to prioritize user-preferred outputs.

For RLHF, although training requires substantial computational resources and time, it achieves the best NDCG@k scores, which is the most critical metric in recommendation tasks. Since the preference dataset was generated using the same metric, we believe this alignment contributes to the observed improvements in NDCG@k for both DPO and RLHF compared to the SFT model. Overall, we observe 0.5%–8% improvements in NDCG@k, Precision@k, and Recall@k over the SFT baseline, highlighting RLHF's ability to enhance the recommendation capabilities of LLMs.

We hypothesize that RLHF slightly outperforms DPO due to the learning process. In RLHF, the preference dataset is first used to train a reward model, which then guides the policy model by providing feedback signals during exploration of its output space. This allows the policy model to iteratively generate outputs that maximize the learned reward, effectively aligning its predictions with user preferences. In contrast, in our framework, the DPO model relies solely on a fixed preference dataset and lacks real-time preference feedback during training. Due to resource constraints, we were unable to generate human-annotated preferences on the fly, which limits the DPO model's ability to explore the output space as effectively as RLHF in our setting.

Table 2 illustrates the progression of model-generated recommendation lists. The Pre-SFT model often produces irrelevant or loosely related outputs, whereas the Post-SFT model demonstrates significant improvements in relevance and alignment with user preferences. Both DPO and RLHF further enhance the ranking quality, delivering outputs that are more closely aligned with the ground truth.

## 7 Conclusion

In this work, we explored the optimization of Large Language Model (LLM)-based recommendation systems using Reinforcement Learning from Human Feedback (RLHF) with Proximal Policy Optimization (PPO) and Direct Preference Optimization (DPO). Starting from a pretrained LLM baseline, we introduced a systematic fine-tuning pipeline consisting of Supervised Fine-Tuning (SFT) followed by preference-based optimization.

Our approach demonstrates the feasibility of adapting LLMs for recommendation tasks by combining task-specific fine-tuning with preference alignment. The results show significant improvements over the baseline, indicating that LLMs can successfully learn to generate structured recommendations with high relevance and ranking quality. Additionally, we provided a qualitative analysis that reveals the progressive improvements in model outputs, bridging the gap between unstructured LLM responses and structured item recommendations.

Overall, this study highlights the strengths and limitations of LLM-based recommendation systems. While our approach demonstrates promising results, it also underscores the need for further work to address scalability, ranking precision, and comparison with traditional collaborative filtering methods. Our findings provide a solid foundation for advancing LLM-driven recommendation systems and exploring more robust optimization techniques.

## 8 Future Works

Future work will explore advanced personalization techniques inspired by Reinforcement Learning with Personalized Human Feedback (RLPHF), particularly the **personalized soups** (Jang et al., 2023) framework. This approach introduces a scalable parameter-merging mechanism to align independently trained policy models with diverse user preferences, eliminating the need for retraining and enabling linear scalability as preferences evolve.

Incorporating personalized soups into our project would facilitate handling conflicting or multidimensional user objectives, such as varying genre preferences, by dynamically merging policies at inference. This could significantly improve personalization and adaptability, supporting the addition of new preferences without retraining the model.

Further directions include contextual personalization through temporal dynamics, session-based interactions, and user mood, as well as hybrid models combining collaborative filtering, content-based methods, and large language models. Improving interpretability and explainability will remain a priority to build user trust and satisfaction.

## References

Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 1007–1014.

Wei Chu and Seung-Taek Park. 2009. Personalized recommendation on dynamic content using predictive bilinear models. In *Proceedings of the 18th international conference on World wide web*, pages 691–700.

Nathan Van Gheem and Boris Verkhovskiy. 2020. pyzipcode. https://github.com/vangheem/pyzipcode.

F. Maxwell Harper and Joseph A. Konstan. 2015. The movielens datasets: History and context. *ACM Transactions on Interactive Intelligent Systems (TiiS)*.

Joel Jang, Seungone Kim, Bill Yuchen Lin, Yizhong Wang, Jack Hessel, Luke Zettlemoyer, Hannaneh Hajishirzi, Yejin Choi, and Prithviraj Ammanabrolu. 2023. Personalized soups: Personalized large language model alignment via post-hoc parameter merging. *Preprint*, arXiv:2310.11564.

Andre Leite and Hugo Vaconcelos. 2024. Rapidfuzz: String similarity computation using 'rapidfuzz'. https://CRAN.R-project.org/package=RapidFuzz.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft.

Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. 2020. Trl: Transformer reinforcement learning. https://github.com/huggingface/trl.

Jie Wang, Alexandros Karatzoglou, Ioannis Arapakis, and Joemon M Jose. 2024a. Reinforcement learning-based recommender systems with large language models for state reward and action modeling. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 375–385.

Ziyan Wang, Yingpeng Du, Zhu Sun, Haoyan Chua, Kaidong Feng, Wenya Wang, and Jie Zhang. 2024b. Re2llm: Reflective reinforcement large language model for session-based recommendation. *arXiv preprint arXiv:2403.16427*.

Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. 2018. Drn: A deep reinforcement learning framework for news recommendation. In *Proceedings of the 2018 world wide web conference*, pages 167–176.

## A Appendix

### A.1 Individual Contributions

The project was a collaborative effort, with each team member contributing to distinct components to ensure its successful completion.

- **Fanyu You**: Developed the SFT pipeline, performed hyperparameter tuning, contributed to the overall framework design, and conducted the literature survey.
- **Rishabh Agrawal**: Developed the DPO pipeline, designed the preference data creation methodology, performed hyperparameter tuning, contributed to the overall framework design, and conducted the literature survey.

- **Mingxi Wang**: Developed the RLHF pipeline, performed hyperparameter tuning, contributed to the overall framework design, and conducted the literature survey.
- **Rutik Yerunkar**: Conducted the literature survey, performed model testing to identify models compatible with the available hardware, and carried out exploratory data analysis (EDA).
- **Yash Gawankar**: Handled data preprocessing, implemented item-space mapping using fuzzy search, developed the preference data creation pipeline, developed the evaluation pipeline, participated in the overall framework design, and conducted the literature survey.