

You are a helpful assistant. Your task is to analyze a user's purchase history, summarize their preferences, analyze a new target item, and then generate a recommendation rationale and predict a rating for that target item. Please follow these steps precisely:

- User Interest Extraction** : ** For *each* item in the provided `user_purchase_history`, analyze the user's `rating` and `review`. Identify and state the points the user liked (`[like]`) and disliked (`[dislike]`). Then based on *all* the points summarized, consolidate the user's overall interest. Categorize them into positive (`[pos]`) and negative (`[neg]`) aspects. Adhere strictly to the following format:

```
''' <analyze user>...</analyze user> '''
```
- Item Attributes Extraction** : ** Extract the features of the target item provided in `{target_item}`. Predict potential points the user might like (`[like]`) or dislike (`[dislike]`) about this specific item, considering general product attributes. Adhere strictly to the following format:

```
''' <analyze item>... </analyze item> '''
```
- Match Analysis** : ** First, internally reason (`<match>`) about *why* the target item would (or would not) be a good recommendation for this user. This reasoning should explicitly connect the user's overall interest (from Step 1) with the target item's potential likes/dislikes (from Step 2). Provide detailed justifications. Adhere strictly to the following format:

```
''' <match>... </match> '''
```
- Rating Prediction** : ** Then, provide the final predicted rating for the target item within the `<rate>` tags. The answer should *only* contain the predicted numerical rating (e.g., on a 1-5 scale). Adhere strictly to the following format:

```
''' <rate>... </rate> '''
```

Your inputs will be `user_purchase_history` and `target_item`. Ensure your output follows the specified formats and uses the `<analyze user>`, `<analyze item>`, `<match>` and `<rate>` tags correctly.

```
<user> user_purchase_history: {history}
target_item: {target_item}
</user>
```

Figure 1: System Prompt

A System Prompt

As illustrated in Fig 1, we guide the early outputs of the LLM through a structured process in the system prompt to achieve faster training convergence and superior model performance. Specifically, within the `<analyze user>` and `</analyze user>` tags, we directed the LLM to first list the features [like] and disliked features [dislike] of each product based on the user's historical interaction records, and then summarize the user's complete preferences using [pos] and [neg] tags. Subsequently, for the target item, we encourage the LLM to use [like] and [dislike] tags within the `<analyze item>` and `</analyze item>` tags to summarize the features that the user might like and dislike about the target item. Following this, the LLM engaged in a thoughtful analysis of the match between the user and the target item within the tags `<match></match>`, and finally provided the predicted user rating within the tags `<rate></rate>`.

B Limitation

While RecZero and RecOne exhibit promising results, this study is not without its limitations. First, due to computational constraints, we were unable to fully assess the potential performance gains that could be achieved by leveraging larger base models as the foundation for RL. This limitation may hinder a thorough evaluation of the models' true performance potential. Second, the study did not explore whether RecZero and RecOne could serve as viable replacements for existing Teacher models in generating cold-start data for multi-round self-iterative optimization. These limitations highlight the need for future work to investigate the scalability and efficacy of RecZero and RecOne in more resource-intensive and complex iterative settings, thereby providing a more comprehensive assessment of their practical applicability.

C Experimental Design and Evaluation

Datasets

To comprehensively validate the effectiveness of our proposed RecZero and RecOne, we conduct systematic experiments on four representative real-world recommendation datasets. Through comparative analysis with various baseline models, including traditional recommendation methods and state-of-the-art (SOTA) LLM-based approaches, we thoroughly demonstrate the superiority of our proposed methods. These datasets span across different domains and scenarios, specifically including: Book, Music, Yelp and IMDb.

- 30 • **Book:**
- 31 This dataset is derived from the "Book" subset of the widely used Amazon dataset for recommen-
- 32 dation scenario evaluation. It records a large number of ratings, reviews, and rich book product
- 33 metadata from users on the Amazon platform in the book scenario.
- 34 • **Music:**
- 35 Similarly, this dataset is from the "Music" subset of the widely used Amazon dataset for recom-
- 36 mendation scenario evaluation, containing user ratings and reviews in the music domain.
- 37 • **Yelp:**
- 38 The Yelp Open dataset contains a large number of ratings and reviews from consumers for local
- 39 restaurants and stores, and is widely used for performance evaluation in recommendation scenarios.
- 40 • **IMDb:**
- 41 This dataset, collected from the IMDb website, includes ratings and reviews in the movie domain
- 42 and is a widely used benchmark in recommendation systems.

43 **Baselines.**

44 We compared RecZero and RecOne with multiple baseline models, including traditional collaborative
 45 filtering (CF) methods, review-enhanced CF methods, and LLM-based approaches, such as MF,
 46 WDL, RGCL, P5, LLMRec, Rec-SAVER, EXP3RT, and Reason4Rec. Among them, MF and WDL
 47 are classic CF methods, while RGCL leverages review information through graph contrastive learning
 48 to enhance recommendation performance. P5 unifies multiple recommendation tasks within an
 49 LLM-based framework, enabling text-based explainable recommendations. We adopted T5-base
 50 as the base LLM model, consistent with the original paper. LLMRec achieves rating prediction
 51 tasks through prompt tuning, and we select the best-performing variant from its versions as the
 52 baseline. Rec-SAVER inputs users' historical interaction data and target item metadata into a teacher
 53 model, encouraging the teacher model to generate intermediate reasoning processes, which are then
 54 distilled into a smaller model to enhance rating prediction capabilities. EXP3RT and Reason4Rec
 55 explicitly decompose single-step reasoning into multi-step processes, further improving the accuracy
 56 of reasoning-enhanced rating prediction tasks.

57 **Training Protocol.**

58 In our study, we employ the Qwen2.5-7B-Instruct-1M model as the starting point for RL due to
 59 its strong instruction-following capabilities and planning abilities acquired during pre-training. For
 60 traditional baseline experiments, we utilize a single H20 GPU, while the LLM-based baselines and
 61 our RecZero and RecOne frameworks are executed on an 8-card H20 GPU setup. All experiments
 62 are conducted using Python 3.9.

63 **Evaluation Protocol.**

64 For the rating prediction task, we employ Mean Absolute Error (MAE) and Root Mean Square Error
 65 (RMSE) as evaluation metrics. MAE measures the prediction accuracy of the model by calculating
 66 the average of the absolute errors between predicted values and true values, where a smaller value
 67 indicates better prediction performance. RMSE evaluates the model's predictive capability by
 68 computing the square root of the average of the squared errors between predicted values and true
 69 values. It is more sensitive to larger errors and provides a better reflection of the overall prediction
 70 stability of the model.

71 **D Statistics**

72 For both RecZero and RecOne, we utilize the Qwen2.5-7B-Instruct-1M model as the starting point
 73 for RL. During training, the batch size is set to 8, and the learning rate is $2e-6$. Each data sample
 74 undergoes 8 rollouts during the training process. We set the sampling temperature to 1.0, the training
 75 epoch to 1, and the KL divergence to 0. Additionally, for the cold-start process of RecOne, we
 76 employ reasoning data provided by DeepSeek-R1 for cold-start initialization. We partition the dataset
 77 and select 1000 data samples that are not included in either the training or test sets for the cold-start
 78 experiments.