Token-level Proximal Policy Optimization for Query Generation

Anonymous ACL submission

Abstract

Query generation is a critical task for web search engines (e.g. Google, Bing) and recommendation systems. Recently, state-of-the-art query generation methods leverage Large Language Models (LLMs) for their strong capabilities in context understanding and text generation. However, they still face challenges in generating high-quality queries in terms of inferring user intent based on their web search interaction history. In this paper, we propose Token-level Proximal Policy Optimization (TPPO), a noval approach designed to empower LLMs perform better in query generation through fine-tuning. TPPO is based on the Reinforcement Learning from AI Feedback (RLAIF) paradigm, consisting of a token-level reward model and a token-level proximal policy optimization module to address the sparse reward challenge in traditional RLAIF frameworks. We conducted experiments on both open-source dataset and an industrial dataset that was collected from a globally-used search engine, demonstrating that TPPO significantly improves the performance of query generation for LLMs and outperforms its existing competitors. The code for TPPO is available at `https://anonymous.4open.science/r/TPPO-D6C6`.

1 Introduction

Web query generation is essential for search engines (He et al., 2009; Aggarwal et al., 2016; Cai et al., 2016; Wu et al., 2018). The task of web query generation is to make the generated queries align with users' personal preferences that better represent their search intent. Such personalized web query is inferred from user's historical search records and should be relevant and meaningful to each user (Baek et al., 2024a; Yang et al., 2023a). It is particularly important for the current personalized search engines such as Bing and Google. Large Language Models (LLMs) have improved search engines and recommendation systems through their text understanding capabilities (Li et al., 2023; Zhao et al., 2023; Wu et al., 2024). However, there still exist challenges in domain-specific tasks such as web query generation in terms of inferring user intents from historical short and ambiguous search queries.

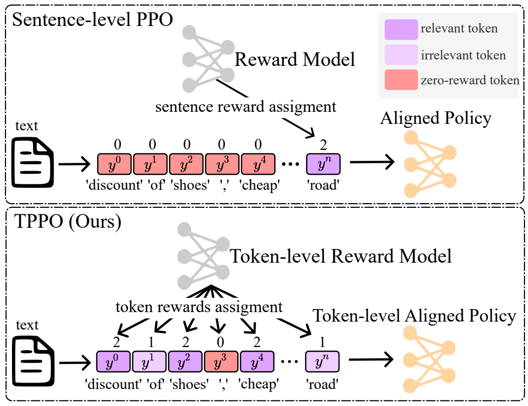

Figure 1: Reward assignment in sentence-level PPO and token-level PPO (TPPO). Sentence-level PPO assigns reward only at the end of a response, whereas TPPO assigns reward for each token in a response.

Supervised fine-tuning (SFT) shows promise for improving LLMs' query generation (Li et al., 2023). However, it faces challenges with language variability, as queries like "cheap flights to New York" and "budget flights NYC" demonstrate diverse phrasing that fixed ground-truth labels can't fully capture. Reinforcement Learning from Human Feedback (RLHF) (Christiano et al., 2017; Ziegler et al., 2019; Stiennon et al., 2020; Ouyang et al., 2022; Bai et al., 2022a) or AI Feedback (RLAIF) (Bai et al., 2022b; Lee et al., 2023) potentially offers better performance than SFT. By incorporating feedback into RL, these approaches learn reward functions and optimize policies to generate aligned responses (Ouyang et al., 2022; Ziegler et al., 2019), helping LLMs better adapt to domain-specific tasks (Kirk et al., 2023; Wang et al., 2023; Ge et al., 2024).

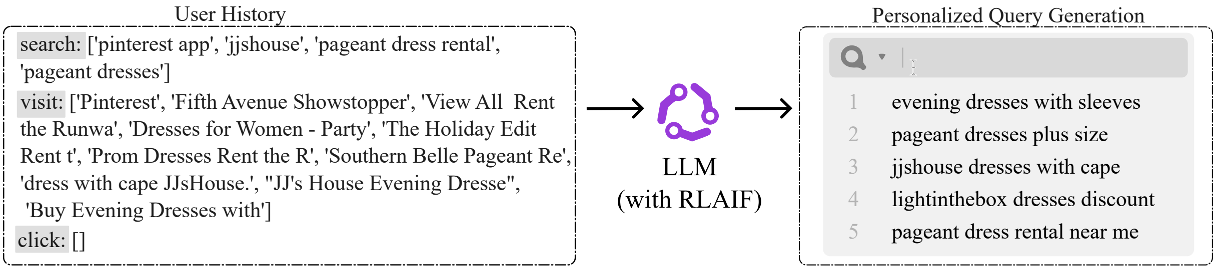

Figure 2: The Query Generation Task. Taking user history as input, the LLM after RLAIF alignment outputs several personalized queries that the user is interested in.

As a seminal policy gradient algorithm in RL, Proximal Policy Optimization (PPO) (Schulman et al., 2017) plays a key role in optimizing agent policies within the RLAIF framework. However, the training of PPO is known to be unstable (Christiano et al., 2017; Rafailov et al., 2024b; Zhong et al., 2024) and one potential reason could be that the reward signal is typically provided at the end of the response sentence, making the reward sparse. The sparse reward makes current PPO in RLAIF actually be sentence-level[1], resulting in some inherent limitations. Firstly, the sparsity of sentence-level rewards creates challenges (Guo et al., 2024; Wu et al., 2023; Rafailov et al., 2024a). Rewards only appear at sentence end, while each token generation is an action receiving no explicit feedback. This sparsity leads to inefficient exploration and makes sentence-level PPO struggle to identify good versus bad actions within sentences. Additionally, sentence-level PPO suffers from temporal delay (Arjona-Medina et al., 2019; Hung et al., 2018) between token generation and rewards, causing training instability. Secondly, traditional PPO formulation mismatches with sentence-level rewards (Uesato et al., 2022; Lightman et al., 2023). While PPO is designed for multi-step RL with step-by-step value estimation, sentence-level rewards prevent the value function from accurately capturing individual actions' long-term impact, resulting in sub-optimal policy updates.

To address above limitations and challenges, our work proposes a token-level PPO (TPPO) as shown in Figure 1. By using token-level reward models and corresponding policies, we mitigate sparse rewards issues and increase training stability. Firstly, to tackle sentence-level reward sparsity, we propose a token-level reward model that assigns rewards to individual tokens within sentences, providing fine-grained feedback. Secondly, to address the PPO formulation mismatch, we introduce a token-level PPO policy aligned with the token-level reward model. This policy learns a value function estimating expected rewards at token level, enabling more informed decisions based on each action's immediate impact. This alignment between token-level reward model and PPO policy mitigates sub-optimal updates. By assigning rewards to individual tokens, the algorithm more accurately attributes credit to specific actions, resulting in more stable updates.

[1]Throughout the paper, we use the term "sentence-level" to represent the sparse reward cases where reward is given at the end of a response or each sentence.

We conduct experiments on both industrial dataset and public benchmarks. Results show TPPO increases query generation relevance by 2%-4% compared to PPO, with 2%-8% higher win rate in item-by-item comparisons. TPPO demonstrates better convergence with steadily increasing rewards, smaller variance, and improved loss. Our model has been successfully deployed in real-world applications. The key contributions are summarized as follows:

- We propose token-level Proximal Policy Optimization (TPPO) for RLAIF, incorporating token-level reward labeling, reward model training, and token-level PPO.
- We are the first to adopt TPPO to empower the query generation task that benefits both academia and industry.
- Comprehensive experiments validate our approach's effectiveness and practicality.

2 Related Work

2.1 Query Generation

Query generation in web search creates new queries aligned with user interests based on search history, browsing behaviors, and contextual information. The aim is to anticipate future information needs and provide relevant search suggestions (He et al., 2009; Aggarwal et al., 2016; Cai et al., 2016). Figure 2 shows the inference process: by analyzing historical queries, browsing behavior, and context, the system generates queries matching the user's specific interests.

However, query generation faces challenges in inferring user intent from short queries and understanding search context (Mustar et al., 2021; Jannach et al., 2022). Recent works leverage LLMs for query generation in recommendation systems (Li et al., 2023; Zhao et al., 2023; Wu et al., 2024; Wei et al., 2024; Lin et al., 2024; Li et al., 2024; Baek et al., 2024b). For instance, GPT4Rec (Li et al., 2023) uses queries generated by fine-tuned GPT-2 to retrieve recommendation items. Despite LLMs' knowledge and in-context learning capabilities, their performance in domain-specific tasks remains suboptimal due to differences between training and domain-specific tasks, and inadequate domain knowledge in pretraining (Bao et al., 2023; Zhang et al., 2023; Yang et al., 2023b; Wang et al., 2024). Reinforcement Learning from AI Feedback (RLAIF) (Bai et al., 2022b; Lee et al., 2023) better aligns LLMs with human preferences in domain-specific tasks. We apply RLAIF to query generation, enabling LLMs to generate queries better aligned with user preferences.

2.2 Proximal Policy Optimization

Proximal Policy Optimization (PPO) (Schulman et al., 2017) is a popular and effective algorithm for policy optimization in reinforcement learning (Kakade and Langford, 2002). Recently, researchers have explored the usage of PPO in the context of RLAIF for natural language processing (NLP) tasks (Ziegler et al., 2019; Bai et al., 2022a; Yue et al., 2023). However, adapting PPO in RLAIF leads to unstable training (Rafailov et al., 2024b; Zhong et al., 2024). Traditional PPO rewards each action, while RLAIF PPO treats entire responses as actions with rewards only at completion. In practice, LLMs generate tokens sequentially, with each token being an action, making sentence-level rewards sparse. This mismatch causes inefficient exploration, sub-optimal updates, and training instability (Xia et al.; Xu et al., 2024). In this paper, we propose token-level PPO that rewards each token to address sparse reward and temporal delay issues (Arjona-Medina et al., 2019; Hung et al., 2018), aligning RLAIF PPO with traditional RL PPO to improve stability and enhance LLM performance in web search query generation.

3 Methodology

In this section, we introduce the problem formulation for the query generation task in Section 3.1 and we then describe the workflow of our token-level PPO within RLAIF framework consists of token-level reward labeling (Section 3.2), reward model training (Section 3.3), and LLM training with token-level PPO (Section 3.4).

3.1 Problem Formulation

We formulate query generation task as a sequential token generation problem. Given an input prompt $\mathbf{x}$ and the previously generated $t-1$ tokens $\{y^{<t}\} = [y^1, y^2, \ldots, y^{t-1}]$ of the query[2], the language model, i.e., the policy π_θ predicts the probability distribution of the next token $\pi_\theta(\cdot|\mathbf{x}, \{y^{<t}\})$. In the our token-level PPO formulation, the *state* of the t^{th} step s_t is a concatenation of the input prompt and the generated response up to this step, denoted as $s_t = [\mathbf{x}, \{y^{<t}\}]$. An *action* corresponds to the next generated token, denoted as $a_t = y^t$, and the *reward* at this step is defined as $R_t = R(s_t, a_t)$. Our objective is to maximize the expected cumulative reward over the sequence of tokens generated by a policy π_θ. The state-action value function is defined as: $Q_{\pi_\theta}(s_t, a_t) = \sum_{k=0}^{\infty} \gamma^k R_{t+k}$. Then, we define the state value function $V_{\pi_\theta}(s_t) = E_{a_t \sim \pi_\theta}[Q_{\pi_\theta}(s_t, a_t)]$ and the advantage function $A_{\pi_\theta}(s_t, a_t) = Q_{\pi_\theta}(s_t, a_t) - V_{\pi_\theta}(s_t, a_t)$ for π_θ.

3.2 Token-Level Reward Labeling

Labeling token-level rewards manually is costly and time-consuming, while LLM-based annotation provides comparable performance(Bai et al., 2022b; Lee et al., 2023; Zheng et al., 2023; Chen et al., 2024). We validated this approach in real-world projects, finding high consistency between LLM and human judgments across sentence-level, word-level annotation, and evaluation. By using word-level rather than token-level annotation, and employing global (sentence-level) and local (word-level) annotations as mutual checks, we further ensure labeling accuracy and quality.

In this paper, we adopt LLaMA 3 (70B) to label token-level rewards due to its strong labeling capability (Touvron et al., 2023) as Phase I in Figure 3 shows, where the query responses are generated by SFT-tuned Mistral-7B model. Compared with sentence-level reward which overlooks the impact of individual tokens (Zeng et al., 2024; Cao et al.,

[2]The initial token is generated given the prompt $\mathbf{x}$ only.

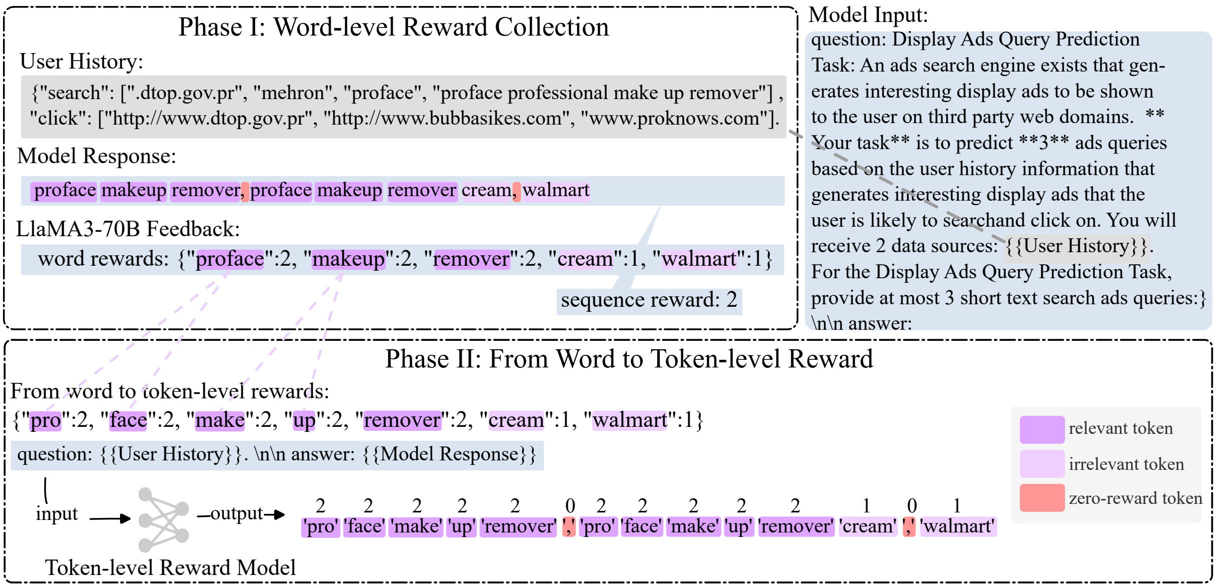

Figure 3: Token-level reward labeling. In phase I, we use LLaMA 3 (70B) to label word-level and sentence-level rewards for the dataset. In phase II, we map the word-level rewards to token-level rewards. The model response and user history are used to construct input for token-level reward model and the mapped token rewards are used as the ground truth for output.

2024; Zeng et al., 2024), the token-level reward is assigned to each token, capturing finer-grained feedback. On the other side, the sentence-level reward provides a holistic feedback on the entire generated query/response, and we include the sentence-level reward to guide the token-level rewards as it is typically easier and less prone to noise.

We design prompts for LLaMA 3 (70B) to score each token's relevance in generated queries (token-level reward) and provide overall query relevance (sentence-level reward). Rewards use three categories: 0 for non-reward/masked tokens, 1 for irrelevant tokens, and 2 for relevant tokens. Only categories 1 and 2 are used for PPO policy updates.

From word-level reward to token-level reward. We first label at word-level for better manageability and generalizability across models with different tokenizers, then map to token-level rewards as shown in Figure 3. For example, the word `relevant` is assigned with a word-level reward of 2. Depending on the tokenizer, the word could split into:

- Model 1: `["re", "levant"]`
- Model 2: `["relev", "ant"]`

We assign the same reward category 2 to each token:

- Model 1: `"re"` $\rightarrow$ `2`, `"levant"` $\rightarrow$ `2`
- Model 2: `"relev"` $\rightarrow$ `2`, `"ant"` $\rightarrow$ `2`

3.3 Reward Model Training

As shown in Figure 4, we use token-level and sentence-level rewards from LLaMA 3 (70B) to design local and global losses for training our token-level reward model.

Local loss of reward model. We apply attention and activation masks to exclude padding areas and non-response tokens. To address class imbalance, we use a probability mask maintaining a 1:3 to 3:1 ratio between label 2 and label 1 tokens, while preserving label 0 tokens. This ensures stable training and proper model convergence.

After applying these masks, the remaining tokens form the *valid set*, denoted as $\mathcal{V}$, which is used for loss computation and gradient backpropagation. The local loss is defined as a weighted cross-entropy loss over a batch of n samples, predicting the probability of each token belonging to one of the three reward categories $\{0, 1, 2\}$. The loss function $\mathcal{L}_{local}(\phi)$ is expressed as:

$$-\frac{1}{n}\sum_{i=1}^{n}\sum_{(s_t,a_t)\in\mathcal{V}}\sum_{c=0}^{2} w_c \mathbf{1}_{[R_\phi(s_t,a_t)=c]}\log P(c\mid s_t,a_t), \tag{1}$$

where $P(c \mid s_t, a_t)$ is the predicted probability that token (s_t, a_t) belongs to class c given by the reward model R_ϕ parameterized by ϕ, and w_c is a class weight. The indicator function $\mathbf{1}_{[\cdot]}$ equals 1 when the condition holds, and 0 otherwise. Specifically, the reward model R_ϕ is implemented using Longformer (Beltagy et al., 2020).

Global loss of reward model. The global loss provides partial supervision by aligning the average

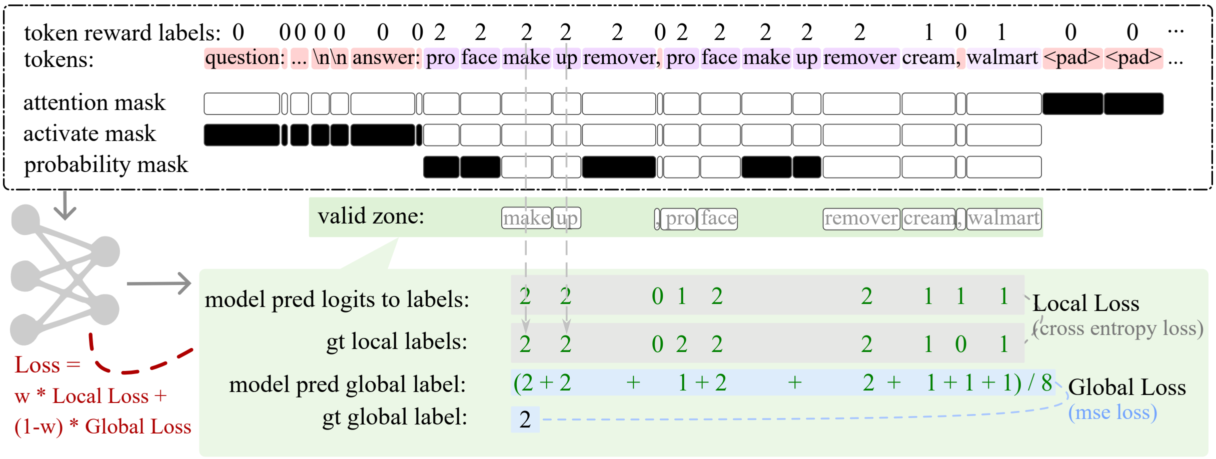

Figure 4: Objectives of token-level reward model. The position after masking (valid zone) is used to calculate the loss and return gradient. The loss of the token-level reward model is the weighted sum of local loss and global loss.

token rewards in $\mathcal{V}$ with the sentence-level reward. It is formulated as the mean squared error (MSE) loss over n samples, measuring the difference between the average token reward and the ground truth global reward. The loss function $\mathcal{L}_{global}(\phi)$ is expressed as:

$$\frac{1}{n}\sum_{i=1}^{n}\left(\frac{1}{|\mathcal{V}|}\sum_{(s_t,a_t)\in\mathcal{V}} R_\phi(s_t,a_t) - R_{global}\right)^2, \tag{2}$$

where $|\mathcal{V}|$ is the number of tokens in the valid set, $R_\phi(s_t,a_t) = \arg\max_{c\in\{0,1,2\}} P(c \mid s_t,a_t)$ is the predicted token reward, and R_{global} is the ground truth global reward. This loss encourages consistency between token-level and sentence-level rewards, ensuring coherent supervision across different levels of granularity.

The total loss for training the reward model combines the local and global losses:

$$\mathcal{L}_{total}(\phi) = \lambda_{local}\mathcal{L}_{local}(\phi) + \lambda_{global}\mathcal{L}_{global}(\phi), \tag{3}$$

where λ_{local} and λ_{global} are hyperparameters controlling the trade-off between local and global supervision.

Length-weighted penalty. When applying the token reward model with PPO, we introduce a length-weighted penalty (lwp) to prevent overly long responses:

$$lwp(l) = \frac{1}{1 + e^{\alpha(l - sl) - 6}}, \tag{4}$$

Here, l is the current token's position, sl is the suggested length (estimated reasonable response length), and α controls penalty intensity. The sl is calculated as the median token length of all generated queries multiplied by the number of queries.

Equation 4 ensures tokens before sl have $lwp \approx 1$, while tokens beyond sl have rapidly decreasing lwp toward 0. We multiply original token-level rewards by this position-specific penalty:

$$R'_\phi(s_t,a_t) = lwp(l) \cdot R_\phi(s_t,a_t). \tag{5}$$

3.4 LLM Training with Token-Level PPO

We introduce token-level PPO where the formulation is matched with token-level reward signal.

Token-level PPO objective function. The token-level PPO objective function is formulated as:

$$\max_{\pi_\theta} E_{\mathbf{x}, y^{<t}\sim\mathcal{D},\, y^t\sim\pi_\theta(\cdot|[\mathbf{x},y^{<t}])}\Big[\min\Big(r_t(\theta)A_{\pi_{\text{ref}}}(s_t,a_t), \text{clip}\left(r_t(\theta),1-\epsilon,1+\epsilon\right)A_{\pi_{\text{ref}}}(s_t,a_t)\Big)\Big], \tag{6}$$

where

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\text{ref}}(a_t \mid s_t)} \tag{7}$$

is the ratio between the new policy π_θ and the old policy π_{ref} at the token level. Here, ϵ is a hyperparameter controlling the clipping range, and $A_{\pi_{\text{ref}}}(s_t,a_t)$ is the advantage function based on the reference policy π_{ref}.

Derivation of the optimal policy. Starting from the token-level PPO objective in Equation 6, we aim to derive the optimal policy π^*_θ. To ensure the policy remains close to a reference policy π_{ref}, we introduce a Kullback–Leibler (KL) divergence constraint. The optimization problem is formulated as:

$$\pi^*_\theta = \arg\max_{\pi_\theta} E_{s_t\sim\mathcal{D},\, a_t\sim\pi_\theta(\cdot|s_t)}\Big[A_{\pi_{\text{ref}}}(s_t,a_t) - \beta\,\text{KL}\left(\pi_\theta(\cdot \mid s_t)\,\|\,\pi_{\text{ref}}(\cdot \mid s_t)\right)\Big], \tag{8}$$

where $\beta > 0$ controls the strength of the KL divergence regularization. The closed-form solution to the optimization problem in Equation 8 is:

$$\pi_\theta^*(a_t \mid s_t) = \frac{\pi_{\text{ref}}(a_t \mid s_t) \exp\left(\frac{1}{\beta} A_{\pi_{\text{ref}}}(s_t, a_t)\right)}{Z(s_t; \beta)}, \quad (9)$$

where $Z(s_t; \beta) = \sum_{a_t} \pi_{\text{ref}}(a_t \mid s_t) \exp\left(\frac{1}{\beta} A_{\pi_{\text{ref}}}(s_t, a_t)\right)$ is the partition function ensuring that π_θ^* is a valid probability distribution.

The optimization problem in Equation 8 yields the optimal policy as given in Equation 9.

4 Experiments

This experiments aim to answer two questions:

Is TPPO more effective than SFT and RL baselines in query generation tasks? To answer this question, we select strong baselines from SFT-based methods and RL-base methods: GPT4Rec (Li et al., 2023) (enhanced with Mistral 7B (Jiang et al., 2023) instead of GPT-2) for SFT methods, and PPO for RL methods. We evaluate using: (1) **Relevance rate**: alignment scores between generated queries and user history via LLaMA 3 (70B), calculated as total relevance scores divided by sample count; and (2) **Win-Tie-Lose rates**: pairwise comparisons between models using LLaMA 3 (70B) voting. Evaluation prompts and additional details are in Appendices D and E-G.

Is TPPO more stable and convergent than PPO in query generation tasks? To investigate this question, we conduct experimental analysis on reward model training and PPO training process separately. Specifically, we compare token-level versus sentence-level reward models through evaluation loss curves and weighted AUC metrics, then analyze TPPO versus PPO training trajectories using mean scores and standard deviations to assess stability properties.

4.1 Experiments on Open-source Data

4.1.1 Dataset Description

The query generation field has limited public datasets, with AOL being the most widely used benchmark. The AOL dataset contains approximately 20 million web queries from about 650k users over three months (MacAvaney et al., 2022), providing real query log data for search research. As shown in Table1, we filtered 27k data points from AOL to create an open-source dataset for query generation. Each data point includes user history (earlier search queries and click records) and the newest 3 search queries as generation targets. We used 25k data for supervised finetuning (2k for evaluation), 10k Llama3-70B-labeled data for token-level reward model training, and 20k data for token-level PPO training (2k for evaluation).

Table 1: Dataset information.

	Industrial Dataset	Open Dataset
User History Keys	search, click, purchase, visit	search, click
Dataset Size	Train: 200k, Eval1: 2k, Eval2: 2k	Train: 25k, Eval: 2k
Tagert Query Num	10	3

Table 2: Results of Relevance Rate on Open-source Dataset.

	GPT4Rec	PPO	TPPO
Relevance rate	41.25	47.65	50.00

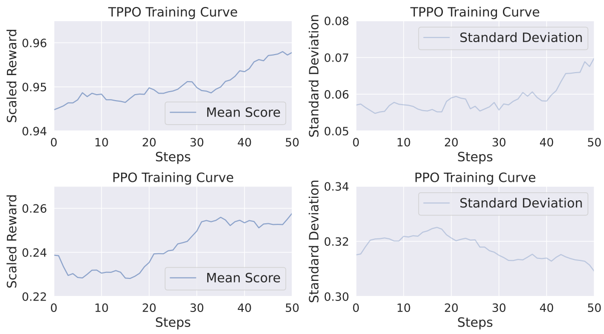

Figure 5: PPO Training Curves on Open-source Dataset.

4.1.2 Results of Token-level PPO Policy

As shown in Figure 7 and Table 2, we compared GPT4Rec, PPO, and TPPO using average relevance scores and pairwise comparisons. TPPO consistently outperforms both alternatives, with win rates 8.75% higher than GPT4Rec and 2.35% higher than PPO in win-tie-lose comparisons.

Moreover, we compared the training performance of models obtained using the traditional PPO and our token-level PPO policy on the same training set. Figure 5 shows that token-level PPO training is more stable (smaller variance) and learns reward model preferences more efficiently (faster score improvement) than traditional PPO.

4.1.3 Results of Token-level Reward Model

As shown in Figure 6, we compared the training curves of the traditional sentence-level reward model and our token-level reward model, both of which have undergone class balancing to achieve best performance. This comparison highlights the advantages of the token-level reward model over

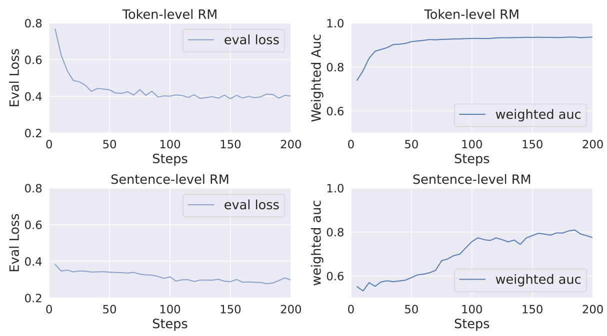

Figure 6: Reward Model Training Curve on Open-source Dataset.

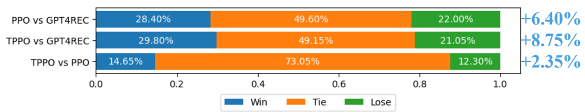

Figure 7: Win-Tie-Lose Comparisons on Open-source Dataset.

the traditional sentence-level approach. Benefited from more granular information, the token-level reward model demonstrates better training stability, faster convergence, and higher performance in terms of AUC (Yang and Ying, 2022).

4.2 Experiments on Industrial data

4.2.1 Dataset Description

As shown in Table 1, we collected industrial data from a popular search engine serving billions of users worldwide. From 400k real user data, we filtered 200k to create an industrial dataset for query generation. This dataset's user history includes four components—search, click, purchase and visit, with the recent 10 search queries serving as generation targets. We created two separate 2k-sample evaluation sets (eval1 and eval2) from different time periods to account for distribution differences.

4.2.2 Results of Token-level PPO Policy

Figure 11 compares traditional PPO with our token-level PPO policy on the same training set, demonstrating token-level PPO provides more stable training (smaller variance) and faster learning of reward model preferences (quicker score improvement).

We used two scoring templates with Llama3-70B: one directly scoring sentences and another scoring words first then sentences. We evaluated on two industrial dataset test sets from different months to validate our approach's stability and applicability. Table 3 shows our method improves relevance rates compared to GPT4Rec and PPO when

Table 3: Results of Relevance Rate on Industrial Dataset, Sentence-level Template for Evaluating.

	GPT4Rec	PPO	TPPO
Relevance rate (Eval 1)	84.10	85.05	88.85
Relevance rate (Eval 2)	84.35	87.50	91.45

Table 4: Results of Relevance Rate on Industrial Dataset, Token-level Template for Evaluating.

	GPT4Rec	PPO	TPPO
Relevance rate (Eval 1)	85.75	92.43	94.79
Relevance rate (Eval 2)	87.40	92.25	94.21

directly scoring sentences. Table 4 confirms this improvement when scoring tokens first then sentences. Figure 8 presents pairwise win-lose comparisons across both evaluation sets and templates, demonstrating TPPO's superior preference fitting. This multi-template, multi-dataset approach confirms TPPO consistently outperforms alternatives regardless of scoring mechanism or dataset timeframe, highlighting its effectiveness and adaptability in real-world industrial applications.

4.2.3 Results of Token-level Reward Model

Figure 9 demonstrates our token-level reward model's advantages over conventional sentence-level approaches. Trained and evaluated on identical datasets with class balancing, our model shows greater stability, faster convergence, and higher AUC scores by utilizing fine-grained token-level information, highlighting the effectiveness of token-level granularity in reward modeling.

5 Ablation Study

We conducted ablation experiments on the industrial dataset examining the two core components of our approach: token-level reward model and token-level PPO policy.

5.1 Losses of Token-level Reward Model

For the Token-level Reward Model training, we used a weighted sum of local and global losses. Figure 10 shows training curves for different values of local loss weight w (0-1). Higher w values produce smaller converged loss values, indicating local labels provide stronger supervision than global labels. Moderate w values (0.4-0.6) show the largest loss reduction from start to convergence, suggesting balanced combination of local and global loss optimizes reward model learning. This confirms the importance of integrating both token-level and sentence-level information.

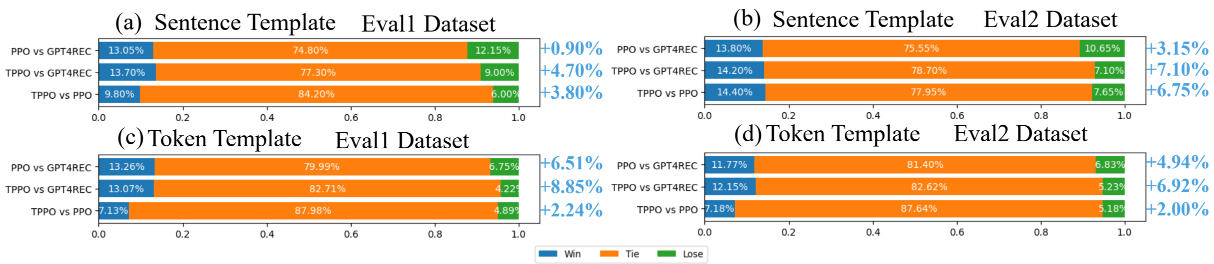

Figure 8: Win-Tie-Lose Comparisons on Industrial Dataset.

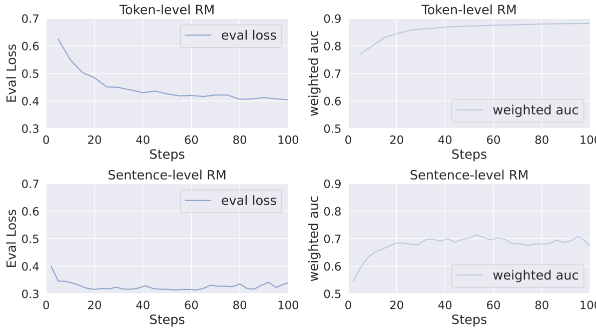

Figure 9: Reward Model Training Curve on Industrial Dataset

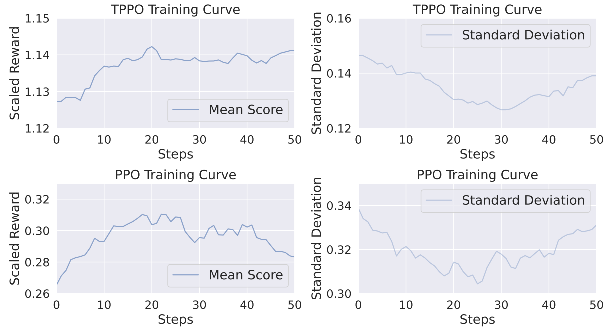

Figure 11: PPO Training Curves on Industrial Dataset.

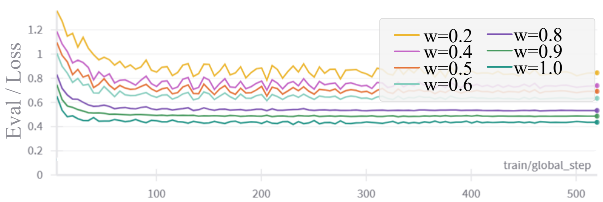

Figure 10: Ablation Study of Losses in Token-level Reward Model Training.

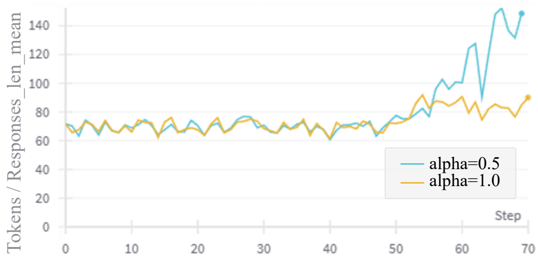

Figure 12: Ablation Study of Length Penalty in Token-level PPO Training.

5.2 Length Penalty of Token-level PPO Policy

We proposed Token-level PPO (TPPO), a novel approach addressing PPO limitations in existing RLHF frameworks for query generation. By introducing token-level reward models and policies, TPPO mitigates sparse rewards, aligns PPO in RLHF with traditional RL PPO, and improves training stability. Experiments on public and industrial datasets demonstrate TPPO's effectiveness, increasing query relevance by 2%-4% compared to PPO, with 2%-8% higher win rates in item-by-item comparisons. TPPO training shows better convergence with stable reward increases, reduced variance, and improved loss. The successful application of TPPO to the query generation task opens up new possibilities for improving the quality and relevance of search results in real-world search engines.

6 Conclusion

We proposed Token-level PPO (TPPO), addressing PPO limitations in RLHF for query generation through token-level reward models and policies. TPPO mitigates sparse rewards, aligns RL-PPO with RLHF, and improves training stability. Experiments show TPPO increases query relevance by 2%-4% with 2%-8% higher win rates versus PPO, while demonstrating better convergence. This TPPO approach enhances real-world search quality.

7 Limitations

TPPO's effectiveness may vary across domains, potentially requiring specific adaptations. In the future research, we will explore the application of our token-level to various domains and further demonstrate the generalizability of the techniques introduced in this work.

References

Charu C Aggarwal and 1 others. 2016. *Recommender systems*, volume 1. Springer.

Jose A. Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. 2019. Rudder: Return decomposition for delayed rewards. *Preprint*, arXiv:1806.07857.

Jinheon Baek, Nirupama Chandrasekaran, Silviu Cucerzan, Allen herring, and Sujay Kumar Jauhar. 2024a. Knowledge-Augmented Large Language Models for Personalized Contextual Query Suggestion. *arXiv preprint*. ArXiv:2311.06318 [cs].

Jinheon Baek, Nirupama Chandrasekaran, Silviu Cucerzan, Allen herring, and Sujay Kumar Jauhar. 2024b. Knowledge-augmented large language models for personalized contextual query suggestion. *Preprint*, arXiv:2311.06318.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, and 1 others. 2022a. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, and 1 others. 2022b. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.

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.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *Preprint*, arXiv:2004.05150.

Fei Cai, Maarten De Rijke, and 1 others. 2016. A survey of query auto completion in information retrieval. *Foundations and Trends® in Information Retrieval*, 10(4):273–363.

Meng Cao, Lei Shu, Lei Yu, Yun Zhu, Nevan Wichers, Yinxiao Liu, and Lei Meng. 2024. Drlc: Reinforcement learning with dense rewards from llm critic. *arXiv preprint arXiv:2401.07382*.

Guiming Hardy Chen, Shunian Chen, Ziche Liu, Feng Jiang, and Benyou Wang. 2024. Humans or llms as the judge? a study on judgement biases. *arXiv preprint arXiv:2402.10669*.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

Yingqiang Ge, Wenyue Hua, Kai Mei, Juntao Tan, Shuyuan Xu, Zelong Li, Yongfeng Zhang, and 1 others. 2024. Openagi: When llm meets domain experts. *Advances in Neural Information Processing Systems*, 36.

Geyang Guo, Ranchi Zhao, Tianyi Tang, Wayne Xin Zhao, and Ji-Rong Wen. 2024. Beyond imitation: Leveraging fine-grained quality signals for alignment. *Preprint*, arXiv:2311.04072.

Qi He, Daxin Jiang, Zhen Liao, Steven CH Hoi, Kuiyu Chang, Ee-Peng Lim, and Hang Li. 2009. Web query recommendation via sequential query prediction. In *2009 IEEE 25th international conference on data engineering*, pages 1443–1454. IEEE.

Chia-Chun Hung, Timothy Lillicrap, Josh Abramson, Yan Wu, Mehdi Mirza, Federico Carnevale, Arun Ahuja, and Greg Wayne. 2018. Optimizing agent behavior over long time scales by transporting value. *Preprint*, arXiv:1810.06721.

Dietmar Jannach, Massimo Quadrana, and Paolo Cremonesi. 2022. Session-based recommender systems. In *Recommender Systems Handbook*, pages 301–334. Springer.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Sham M. Kakade and John Langford. 2002. Approximately optimal approximate reinforcement learning. In *International Conference on Machine Learning*.

Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. 2023. Understanding the effects of rlhf on llm generalisation and diversity. *arXiv preprint arXiv:2310.06452*.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and 1 others. 2023. Rlaif: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*.

Jinming Li, Wentao Zhang, Tian Wang, Guanglei Xiong, Alan Lu, and Gerard Medioni. 2023. Gpt4rec: A generative framework for personalized recommendation and user interests interpretation. *arXiv preprint arXiv:2304.03879*.

Yongqi Li, Xinyu Lin, Wenjie Wang, Fuli Feng, Liang Pang, Wenjie Li, Liqiang Nie, Xiangnan He, and Tat-Seng Chua. 2024. A survey of generative search and recommendation in the era of large language models. *Preprint*, arXiv:2404.16924.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

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. *Preprint*, arXiv:2306.05817.

Sean MacAvaney, Craig Macdonald, and Iadh Ounis. 2022. Reproducing personalised session search over the aol query log. *Preprint*, arXiv:2201.08622.

Agnès Mustar, Sylvain Lamprier, and Benjamin Piwowarski. 2021. On the study of transformers for query suggestion. *ACM Transactions on Information Systems (TOIS)*, 40(1):1–27.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024a. Direct preference optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024b. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*.

Cunxiang Wang, Xiaoze Liu, Yuanhao Yue, Xiangru Tang, Tianhang Zhang, Cheng Jiayang, Yunzhi Yao, Wenyang Gao, Xuming Hu, Zehan Qi, and 1 others. 2023. Survey on factuality in large language models: Knowledge, retrieval and domain-specificity. *arXiv preprint arXiv:2310.07521*.

Yuling Wang, Changxin Tian, Binbin Hu, Yanhua Yu, Ziqi Liu, Zhiqiang Zhang, Jun Zhou, Liang Pang, and Xiao Wang. 2024. Can small language models be good reasoners for sequential recommendation? In *Proceedings of the ACM on Web Conference 2024*, pages 3876–3887.

Wei Wei, Xubin Ren, Jiabin Tang, Qinyong Wang, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. 2024. Llmrec: Large language models with graph augmentation for recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pages 806–815.

Bin Wu, Chenyan Xiong, Maosong Sun, and Zhiyuan Liu. 2018. Query suggestion with feedback memory network. In *Proceedings of the 2018 World Wide Web Conference*, pages 1563–1571.

Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, and 1 others. 2024. A survey on large language models for recommendation. *World Wide Web*, 27(5):60.

Zeqiu Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A. Smith, Mari Ostendorf, and Hannaneh Hajishirzi. 2023. Fine-grained human feedback gives better rewards for language model training. *Preprint*, arXiv:2306.01693.

Han Xia, Songyang Gao, Qiming Ge, Zhiheng Xi, Qi Zhang, and Xuanjing Huang. Inverse-Q*: Token Level Reinforcement Learning for Aligning Large Language Models without Preference Data.

Dehong Xu, Liang Qiu, Minseok Kim, Faisal Ladhak, and Jaeyoung Do. 2024. Aligning Large Language Models via Fine-grained Supervision. *arXiv preprint*. ArXiv:2406.02756 [cs].

Fan Yang, Zheng Chen, Ziyan Jiang, Eunah Cho, Xiaojiang Huang, and Yanbin Lu. 2023a. Palr: Personalization aware llms for recommendation. *Preprint*, arXiv:2305.07622.

Fangkai Yang, Pu Zhao, Zezhong Wang, Lu Wang, Bo Qiao, Jue Zhang, Mohit Garg, Qingwei Lin, Saravan Rajmohan, and Dongmei Zhang. 2023b. Empower large language model to perform better on industrial domain-specific question answering. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: EMNLP 2023 - Industry Track, Singapore, December 6-10, 2023*, pages 294–312. Association for Computational Linguistics.

Tianbao Yang and Yiming Ying. 2022. Auc maximization in the era of big data and ai: A survey. *Preprint*, arXiv:2203.15046.

Zhenrui Yue, Sara Rabhi, Gabriel de Souza Pereira Moreira, Dong Wang, and Even Oldridge. 2023. Llamarec: Two-stage recommendation using large language models for ranking. *Preprint*, arXiv:2311.02089.

Yongcheng Zeng, Guoqing Liu, Weiyu Ma, Ning Yang, Haifeng Zhang, and Jun Wang. 2024. Token-level direct preference optimization. *arXiv preprint arXiv:2404.11999*.

Jianyi Zhang, Aashiq Muhamed, Aditya Anantharaman, Guoyin Wang, Changyou Chen, Kai Zhong, Qingjun Cui, Yi Xu, Belinda Zeng, Trishul Chilimbi, and 1 others. 2023. Reaugkd: Retrieval-augmented knowledge distillation for pre-trained language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1128–1136.

Zihuai Zhao, Wenqi Fan, Jiatong Li, Yunqing Liu, Xiaowei Mei, Yiqi Wang, Zhen Wen, Fei Wang, Xiangyu Zhao, Jiliang Tang, and 1 others. 2023. Recommender systems in the era of large language models (llms). *arXiv preprint arXiv:2307.02046*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

Han Zhong, Guhao Feng, Wei Xiong, Li Zhao, Di He, Jiang Bian, and Liwei Wang. 2024. Dpo meets ppo: Reinforced token optimization for rlhf. *arXiv preprint arXiv:2404.18922*.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.

A Proof of Lemma 3.4

To derive the optimal policy π_θ^* for the optimization problem in Eq. 8, we frame the problem using the method of Lagrange multipliers to incorporate the normalization constraint of the probability distribution π_θ. The Lagrangian $\mathcal{L}$ is defined as:

$$\begin{aligned}\mathcal{L}(\pi_\theta, \lambda(s_t)) = &\sum_{a_t} \pi_\theta(a_t|s_t) \left[A_{\pi_{\text{ref}}}(s_t, a_t)\right. \\ &- \beta \log \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)} \\ &- \lambda(s_t) \left(\sum_{a_t} \pi_\theta(a_t|s_t) - 1\right),\end{aligned} \tag{10}$$

where $\lambda(s_t)$ is the Lagrange multiplier ensuring that π_θ sums to 1 over all actions a_t.

Taking the derivative of $\mathcal{L}$ with respect to $\pi_\theta(a_t|s_t)$ and setting it to zero gives:

$$\frac{\partial \mathcal{L}}{\partial \pi_\theta(a_t|s_t)} = A_{\pi_{\text{ref}}}(s_t, a_t) - \beta \left(1 + \log \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)}\right) - \lambda(s_t) = 0. \tag{11}$$

Solving for $\pi_\theta(a_t|s_t)$:

$$\beta \log \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)} = A_{\pi_{\text{ref}}}(s_t, a_t) - \beta - \lambda(s_t), \tag{12}$$

$$\log \frac{\pi_\theta(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)} = \frac{1}{\beta} A_{\pi_{\text{ref}}}(s_t, a_t) - \frac{\lambda(s_t)}{\beta} - 1, \tag{13}$$

$$\pi_\theta(a_t|s_t) = \pi_{\text{ref}}(a_t|s_t) \exp\left(\frac{1}{\beta} A_{\pi_{\text{ref}}}(s_t, a_t) - \frac{\lambda(s_t)}{\beta} - 1\right). \tag{14}$$

The terms $-\frac{\lambda(s_t)}{\beta} - 1$ are constants with respect to a_t for a given s_t and ensure that π_θ is a valid probability distribution. They can be absorbed into the partition function $Z(s_t; \beta)$. Thus, we can write:

$$\pi_\theta^*(a_t|s_t) = \frac{\pi_{\text{ref}}(a_t|s_t) \exp\left(\frac{1}{\beta} A_{\pi_{\text{ref}}}(s_t, a_t)\right)}{Z(s_t; \beta)}, \tag{15}$$

where the partition function $Z(s_t; \beta)$ is defined as:

$$Z(s_t; \beta) = \sum_{a_t} \pi_{\text{ref}}(a_t|s_t) \exp\left(\frac{1}{\beta} A_{\pi_{\text{ref}}}(s_t, a_t)\right). \tag{16}$$

This completes the proof.

B Theoratical Justification for Why Token-level Rewards Results in Better Policy

TPPO improves upon traditional PPO by addressing key limitations such as sparse rewards, delayed credit assignment, and high variance. Below, we outline a theoretical analysis to justify its superiority:

B.1 Improved Gradient Signal

PPO Gradient. The PPO objective gradient with sparse rewards is:

$$\nabla_\theta J_{\text{PPO}} = E\pi_\theta \left[A_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right], \tag{17}$$

where A_t (advantage) is based on sentence-level rewards. For $t < H$, $A_t \approx 0$, resulting in weak gradients for earlier tokens.

TPPO Gradient. TPPO uses dense token-level rewards $R(s_t, a_t)$, where:

$$\nabla_\theta J_{\text{TPPO}} = E\pi_\theta \left[A_t^{\text{token}} \nabla_\theta \log \pi_\theta(a_t \mid s_t)\right], \tag{18}$$

and $A_t^{\text{token}} = Q^{\text{token}}(s_t, a_t) - V^{\text{token}}(s_t)$. Since $R(s_t, a_t)$ provides feedback for all tokens, the gradient signal is significantly stronger:

$$\|\nabla_\theta J_{\text{TPPO}}\| > \|\nabla_\theta J_{\text{PPO}}\|. \tag{19}$$

B.2 Variance Reduction

PPO Variance In PPO, the advantage function A_t at time step t is defined as:

$$A_t = Q(s_t, a_t) - V(s_t), \tag{20}$$

where $Q(s_t, a_t)$ is the expected cumulative reward and $V(s_t)$ is the baseline value function.

For sparse sentence-level rewards, the cumulative reward $Q(s_t, a_t)$ depends primarily on the final reward r_{final}, discounted back to time step t:

$$Q(s_t, a_t) = \gamma^{H-t} r_{\text{final}}, \tag{21}$$

where H is the length of the sequence. Earlier tokens ($t < H$) rely on the final reward $r_{\text{final}}(x, y)$ as feedback. The discount factor $\gamma \in (0, 1]$ reduces the contribution of this reward as it is propagated backward, making earlier tokens highly dependent on γ^{H-t}.

The variance of the advantage function A_t is directly proportional to the variance of the discounted reward $Q(s_t, a_t)$:

$$\text{Var}(A_t) \propto \text{Var}(Q(s_t, a_t)) \propto \gamma^{2(H-t)} \cdot \text{Var}(r_{\text{final}}) \tag{22}$$

TPPO Variance. In TPPO, token-level rewards $R(s_t, a_t)$ provide stepwise feedback for each token, distributing the reward signal evenly:

$$\text{Var}(A_t^{\text{token}}) \propto \frac{1}{H} \sum_{t=1}^{H} \text{Var}(R(s_t, a_t)). \tag{23}$$

This reduces the dependency on γ^{H-t} and ensures a lower variance in advantage estimation across all tokens.

Thus, TPPO reduces variance in advantage estimation, stabilizing policy updates.

B.3 Faster Convergence

PPO Convergence. Sparse rewards delay feedback for earlier tokens, slowing learning. Improvement per update is:

$$\Delta J_{\text{PPO}} \propto \|\nabla_\theta J_{\text{PPO}}\| \cdot \frac{1}{\text{Var}(A_t)}. \tag{24}$$

TPPO Convergence. Dense rewards provide immediate feedback, increasing $\|\nabla_\theta J_{\text{TPPO}}\|$ and reducing $\text{Var}(A_t^{\text{token}})$:

$$\Delta J_{\text{TPPO}} > \Delta J_{\text{PPO}}. \tag{25}$$

Thus, TPPO achieves faster convergence due to stronger gradients and lower variance.

C Analysis and Comparison of The Algorithmic Complexity and Convergency between TPPO and Prior Work

We have analyzed the convergence properties of our method in Figure 5 and Figure 11, which demonstrate TPPO's superior convergence characteristics compared to baseline approaches. Here, we provide a brief analysis for algorithmic complexity analysis and comparisons between TPPO and PPO:

Our method, TPPO, has a similar complexity to PPO because the dominant computational cost in both methods comes from the policy update step, which scales as $\mathcal{O}(H \cdot N)$, where H is the sequence length and N is the number of model parameters. While TPPO introduces token-level rewards with an additional $\mathcal{O}(H \cdot N)$ cost for reward computation (M being the reward model complexity), this step can be efficiently parallelized. In practice, the additional overhead is negligible, and the overall training time is comparable to PPO. TPPO thus achieves better stability and performance without significant computational cost increases.

D Prompt Template for Labeling Relevance Score

Figure. 13 shows the token template for labeling relevance score, where the "User Queries" represents the ground truth, and "Returned Queries" are our model-generated outputs, with relevance scores annotated by LLM between these pairs. In this work, we use sentence templates (Figure 14) for PPO training and token templates (Figure 13) for TPPO training - they serve different purposes in our framework. Sentence templates only label sentence-level rewards, while Token templates label word-level rewards and sentence-level rewards.

```
# Task
- Background: A chatbot exists that can summarize user history into predicted ads search queries.
These ads search queries are then fed into an ads search engine to generate relevant display ads to show the user.
The order of queries does not matter.
- Task: review the input to the chatbot and the generated queries, and make various judgements about the quality
of each word in the queries generated by the chatbot relevant to the user history.  Each judgment should be
accompanied by a score, do not provide explanation.

# Judgement Criterias
## Word Sequence Metrics
- Relevance: taking the context of the returned queries into consideration, is the word relevant to the user queries?
-- Scoring rules: 0 indicates the word is irrelevant or hardly relevant, 1 indicates the word is higly relevant.
-- Scoring type: integer

## Across Queries Metrics
- Relevance: are the returned queries relevant to the user queries?
-- Scoring rules: 0 indicates the returned queries are irrelevant or hardly relevant, 1 indicates the returned queries are higly relevant.
-- Scoring type: integer

# Formatting Rules
Output the responses as a JSON Dictionaries:
<WORD_SEQUENCE_JUDGEMENTS>
{
"<word1>" : {"Relevance":{"Score" : 0 or 1}},
"<word2>" : {"Relevance":{"Score" : 0 or 1}},
"<word3>" : {"Relevance":{"Score" : 0 or 1}},
    ...
}
</WORD_SEQUENCE_JUDGEMENTS>
<ACROSS_QUERY_JUDGEMENTS>
{"Relevance":{"Score" : 0 or 1}}
</ACROSS_QUERY_JUDGEMENTS>

# Begin Task
## User History
{{USER_HISTORY_DATA}}

## User Queries
{{USER_QUERIES_DATA}}

## Returned Queries
{{RETURNED_QUERIES_DATA}}

Output the responses as a JSON Dictionaries:
```

Figure 13: Token Template for Labeling Relevance Score.

```
# Task
- Background: A chatbot exists that can summarize user history into predicted ads search queries.
These ads search queries are then fed into an ads search engine to generate relevant display ads to show the user.
The order of queries does not matter.
- Task: review the input to the chatbot and the generated queries, and make various judgements about the quality
of the queries generated by the chatbot relevant to the user history.
Each judgment should be accompanied by a score, do not provide explanation.

# Judgement Criterias

## Across Queries Metrics
- Relevance: are the returned queries relevant to the user queries?
-- Scoring rules: 0 indicates the returned queries are irrelevant or hardly relevant, 1 indicates the returned queries are higly relevant.
-- Scoring type: integer

# Formatting Rules
<ACROSS_QUERY_JUDGEMENTS>
{"Relevance":{"Score" : 0 or 1}}
</ACROSS_QUERY_JUDGEMENTS>

# Begin Task
## User History
{{USER_HISTORY_DATA}}

## User Queries
{{USER_QUERIES_DATA}}

## Returned Queries
{{RETURNED_QUERIES_DATA}}

Output the responses as a JSON Dictionaries:
```

Figure 14: Sentence Template for Labeling Relevance Score.

E Hyper-parameters of TPPO Setting in Experiment

Here we clarify the parameters used in our implementation: the hyper-parameters of TPPO in our implementation are show in Table 5, and the hyper-parameters of Token-level Reward Model are shown in Table 6.

F Brief Pseudocode for TPPO (Natural Language)

We provide a simplified pseudocode for policy training below:
Initialize the policy model (π_θ) and token-level reward model (R_ϕ) with their respective learning rates and hyperparameters (e.g., KL coefficient, clip threshold).
For each training iteration:

- Sample a batch of prompts from the dataset.
- Generate responses for each prompt using the current policy model (π_θ).
- Compute token-level rewards ($R_\phi(s_t, a_t)$) for each token in the responses using the reward model.
- Calculate token-level advantages (A_t) using the token rewards and value estimates.
- Update the policy model (π_θ) by optimizing the PPO objective, ensuring stable updates with clipping.
- Update the token-level reward model (R_ϕ) based on token labels and global constraints.

Output the optimized policy model (π_θ).

Table 5: The Hyper-parameters of TPPO.

Training Configuration	Extra Hyperparameters
Learning rate: 5e-6 Batch size: 32 Hardware: 8*A100 GPUs Training epochs: 2 KL coefficient: 0.2 Ouput max length: 400	Query nums: 10 Alpha: 2.0

Table 6: The Hyper-parameters of Token-level Reward Model.

Training Configuration	Extra Hyperparameters
Learning rate: 1e-5 Batch size: 32 Hardware: 1 x V100 GPU Gradient accumulation step: 8 Max length: 2048	Num class labels: 3 POS NEG Ratio: 3.0 Local Weight: 0.4 Global Weight: 0.6

G Models Used in Each Step and Online Serving Implementation

Model usage in each step. To further clarify the whole process, we summarize the each step with model used:

1. Train Mistral-7B by SFT. The user history and ground truth queries are given as training data.

2. Use SFT-tuned Mistral-7B to generate multiple queries. These queries are later used for labeling relevance score (reward) by comparing with ground truth user queries, as shown in Figure 13.

3. Employ Llama3-70B to generate token-level and sentence-level reward, creating data for reward model training.

4. Train Longformer as reward model through SFT. We implement a novel combination of local and global loss to enable token-level reward prediction capabilities of Longformer.

5. Finally, we optimize the SFT-tuned Mistral-7B using our TPPO policy. It is notable that, during this step, the SFT-tuned Longformer serve as the token-level reward model, without further training.

Online serving implementation. In our production environment, the in-house LLMs undergo continuous iteration and exist at various scales. Although the base LLMs vary, we consistently apply our TPPO methodology in training. For the purpose of academic demonstration in this paper, we selected Mistral-7B as our experimental base model to demonstrate the superiority of TPPO method.

H Additional DPO Experiments

To comprehensively evaluate different reinforcement learning methods, we conducted additional experiments using DPO (Direct Preference Optimization). The experimental setup and results are detailed below.

H.1 Experimental Setup

In constructing the training data, we maintained consistency with the PPO training dataset prompts, using real user queries as accept data and GPT4REC-generated queries as corresponding reject data, creating a 20k training dataset. We implemented and trained DPO with a batch size of 64 and learning rate of 5e-6 for 2 epochs until convergence, then performed inference and evaluation on the industrial dataset validation set (consistent with the validation set in the paper).

H.2 Results

The experimental results are presented in Tables 7 and 8. The results demonstrate that DPO performs slightly better than GPT4REC (relevance rate: 84.50 vs 84.10), slightly worse than PPO (84.50 vs 85.05), and significantly worse than our proposed TPPO method (84.50 vs 88.85). These findings support our conclusion that DPO may not be optimal for complex query generation tasks due to sparse reward signals. The experimental results further validate that our TPPO method outperforms both advanced SFT baselines (such as GPT4REC) and mainstream reinforcement learning approaches (including PPO and DPO).

Table 7: Relevance Rate Comparison

Model	**Relevance Rate**
GPT4REC	84.10
PPO	85.05
DPO	84.50
TPPO	88.85

Table 8: DPO Performance Comparison

Metric	**vs. GPT4REC**	**vs. PPO**	**vs. TPPO**
Win Rate	13.60%	12.20%	9.40%
Tie Rate	73.15%	75.05%	76.85%
Lose Rate	13.25%	12.75%	13.75%

I Ground Truth Validation of Reward Labels

To validate the reliability of LLaMA-3 annotations used in our reward modeling, we conducted a consistency test comparing model annotations with human labels on a sample set of 60 examples. The validation was performed at two granularity levels:

Table 9: Annotation Consistency between LLaMA-3 and Human Labels

Consistency Level	**Agreement Rate**
Word-level	91.18%
Sentence-level	80.95%

The high consistency rates, particularly the 91.18% agreement at word level, demonstrate that LLaMA-3 provides reliable annotations that align well with human judgment.