Generation Space Preference Optimization for Few Shot Dialogue State Tracking

Anonymous ACL submission

Abstract

Dialogue State Tracking (DST) is an essential component of task-oriented dialogue systems. Few-shot DST effectively reduces the reliance on large-scale annotated data, but suffers from insufficient training. In this work, we propose a novel training method called **G**eneration **S**pace **P**reference **O**ptimization (GSPO) to mitigate insufficient training for few-shot DST, which extends preference optimization to DST and generates preference data by the model's generation space and the reuse of supervised fine-tuned (SFT) data, free of extra reward models and additional preference data. Experimental results demonstrate that our method achieves competitive performance compared to those using 100 B-scale LLMs and shows better performance with over 5% of the whole training data (400 training samples).

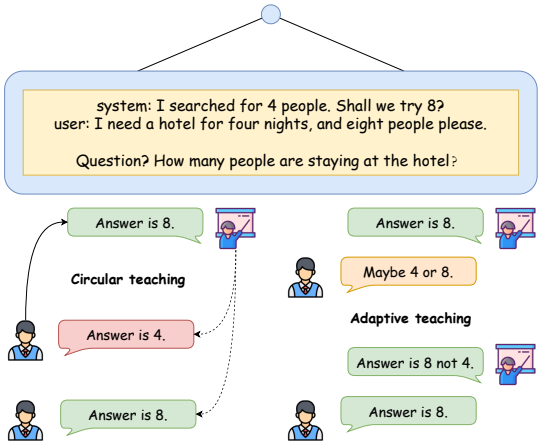


Figure 1: Examples of different ways of learning. The left is learning with circular teaching. The right is adaptive teaching, that learning by correcting and enhancing based on existing ability.

1 Introduction

As a key component in Task-Oriented Dialogue (TOD) systems, Dialogue State Tracking (DST) serves to extract the user's intentions and goals throughout the conversation, which is common in restaurant reservation, ticket booking, and so on. DST is responsible for tracking the value of predefined domain-specific slots and forming the structured dialogue state for the subsequent response generation. Many recent works (Wu et al., 2019; Heck et al., 2020) have proposed various methods that show notable performance in the DST task with a large amount of annotated training data. However, large-scale annotation of training data is time-consuming and labor-intensive. Moreover, considering the diversity of conversation scenarios, we often utilize specific DST models for different scenarios and adopt incremental training for new scenarios. Therefore, researchers have begun to explore few-shot DST.

In general, few-shot DST has the following three main directions: (1) Design universal structures suitable for various conversations, such as copy mechanisms or copy strategies (Wu et al., 2019; Heck et al., 2020). (2) Generate large amounts of synthetic data or utilize related task data for training (Lin et al., 2021a; Shin et al., 2022). (3) Adopt powerful LLMs to solve the DST task with task-specific fine-tuning or in-context learning (Hu et al., 2022; Feng et al., 2023). However, these works overlook that the fundamental problem of few-shot learning is insufficient training caused by limited training data. We observe that many correct answers exist beyond the model's top-1 prediction. In this work, we leverage the non-top-1 correct answers in the model's generation space to mitigate the influence of insufficient training.

Data scaling law (Kaplan et al., 2020; Bahri et al., 2024) shows that the model performs better as the amount of data scales up. However, in this work, we prefer studying training with limited data. Take the example of student learning shown in Figure 1, we want the student to acquire the ability to answer some questions. In an ideal scenario, we can

continue teaching students different examples until they master the methodology for solving similar problems. However, due to the limited number of available examples, the students are often compelled to memorize specific cases through circular learning in real scenarios. This method of enforced memorization is far from actual mastery. Similarly, excessive training on limited data offers little benefit for the generalization to unseen data. In fact, the students may master parts after each learning and have the possibility to solve the problem. If we can instruct them based on their current ability, it may help them master their ability more effectively.

In this work, we extend the use of preference optimization to the DST task and propose a novel training method called **G**eneration **S**pace **P**reference **O**ptimization (GSPO) for the few-shot DST task. It's a two-stage training method: (1) In the first stage, we utilize the limited data to train the model through standard instruction tuning. (2) In the second stage, we first analyze the ability of the trained model on the training data, then construct preference data based on the trained model's generation space and the reuse of training data. Finally, we optimize the trained model with preference optimization to correct errors and enhance weaknesses. The analysis of the trained model's generation space and the reuse of supervised fine-tuned (SFT) data ensures that we neither rely on the extra reward models nor additional preference data during preference optimization. We believe that the preference optimization of the second stage can further improve the model's ability and stimulate its performance under limited data training. We conduct extensive experiments on MultiWOZ 2.1 and MultiWOZ 2.4. The experimental results show the effectiveness of our proposed method. In summary, the main contributions of our work are as follows:

- We extend preference optimization to tasks characterized by unique and deterministic ground truth, and find that it efficiently mitigates the influence of insufficient training.
- We propose a novel two-stage training method called generation space preference optimization (GSPO), which constructs preference data by analyzing the model's generation space and reusing SFT data, free of extra reward models and additional preference data.
- We conduct extensive experiments on MultiWOZ 2.1 and MultiWOZ 2.4 to validate the proposed method's effectiveness and achieve competitive performance over compared baselines.

2 Related Work

2.1 Dialogue State Tracking

Few-shot dialogue state tracking has primarily evolved along three main directions. First, researchers have explored the design of specialized model architectures that can learn to identify dialogue states from limited training samples. Several studies (Zhong et al., 2018; Wu et al., 2019; Heck et al., 2020) propose copy mechanisms or copy strategies to extract dialogue states directly from the dialogue history. However, the performance of such methods tends to degrade significantly when training data is scarce. Second, data augmentation-based approaches have been introduced to alleviate the data sparsity problem. For example, Shin et al. (2022) reformulates DST as a dialogue summarization task and generates large-scale synthetic data through template-guided summarization. Lin et al. (2021a) enhances training data by employing cross-task transfer learning techniques. Third, with the rapid advancement of large language models (LLMs), numerous approaches have leveraged their capabilities to improve DST performance. Some works (Hosseini-Asl et al., 2020; Lin et al., 2021b; Feng et al., 2023) achieve strong results by fine-tuning LLMs. Others (Hu et al., 2022; Heck et al., 2023; King and Flanigan, 2023) utilize the in-context learning (ICL) abilities of LLMs, optimizing the selection of retrieval examples to boost DST performance. Despite their effectiveness, fine-tuning-based methods remain sensitive to the size of the training dataset, whereas prompt-based methods heavily rely on the inherent capabilities of the underlying LLM, typically requiring models with a very large number of parameters to perform well. Similar to fine-tuning-based approaches, our method involves an initial stage of instruction tuning on a base model. However, to further enhance performance, we introduce a second stage based on preference optimization, which encourages the model to generate more accurate dialogue states by learning from pairwise comparisons.

2.2 Preference Optimization

LLMs pre-trained on large-scale data have demonstrated powerful capabilities across various tasks

with few-shot prompts (Radford et al., 2019; Brown et al., 2020), and their performance on downstream tasks can be significantly enhanced through instruction tuning (Mishra et al., 2022; Sanh et al., 2021). However, human judgment of the quality is more meaningful than fixed metrics. As a result, some works (Kreutzer et al., 2018; Ziegler et al., 2019; Ouyang et al., 2022) aim to fine-tune LLMs using human preference datasets to further enhance their usability. These works need an extra well-trained reward model to assess the human preference for a given output, then fine-tune a language model to maximize the reward using reinforcement learning, commonly through TRPO (Schulman, 2015), PPO (Schulman et al., 2017), GRPO (Shao et al., 2024), and others. However, a well-trained reward model is very important to the final performance and usually requires specific data for training to better distinguish between different preferences. To simplify the process of reinforcement learning, some single-stage policy learning approaches have been proposed, such as DPO (Rafailov et al., 2024), CPO (Xu et al., 2024), KTO (Ethayarajh et al., 2024), and so on. Though these single-stage policy learning approaches eliminate the need for extra reward models, they usually need additional preference data for preference optimization. Similar to single-stage policy learning approaches, our proposed GSPO is also free of extra reward models. Apart from that, our method requires no additional preference data because we construct preference data from the model's generation space and the reuse of SFT data.

3 Preliminary

Dialogue State Tracking (DST) aims to monitor the evolving state of a conversation by maintaining a structured representation known as the dialogue state. This dialogue state is typically represented as a set of triples in the format "domain-slot-value", such as "hotel-area-centre". Both the domain and slot in the triple are predefined manually. The domain denotes the topic of the dialogue, while the slot corresponds to specific attributes associated with that domain. For ease of reference, we treat "slot" as a "domain-slot" pair throughout the rest of the paper. The value of the dialogue state can be of two types: enumerable and non-enumerable. An enumerable slot has a predefined set of possible values (e.g., "hotel-pricerange"), whereas a non-enumerable slot takes dynamic values that emerge during the conversation (e.g., "hotel-name"). In brief, the only task of DST is to predict the value of the slot based on the conversation.

Formally, a dialogue $\mathcal{D}$ with T turns can be represented as $\{(R_1, U_1), (R_2, U_2), \cdots, (R_T, U_T)\}$, where R represents system response and U represents user utterance. Given all predefined slots $\mathcal{S} = \{S_1, S_2, \cdots, S_N\}$, we predict the corresponding value of the slots as $\mathcal{V} = \{V_1, V_2, \cdots, V_N\}$. For some slots, the value may not exist and is defined as "none". The belief state $\mathcal{B}$ is defined as a set $\{(S_i, V_i) \mid V_i \neq none\}$. In short, DST problem is defined as $(\mathcal{D}, \mathcal{S}) \rightarrow \mathcal{B}$.

To track the dialogue state throughout a conversation, two primary approaches are commonly used: **state refresh** and **state change**. State refresh regenerates the entire dialogue state at each turn by leveraging the full dialogue history. State change updates the dialogue state incrementally using the current turn's dialogue context and the previous dialogue state. Formally, state refresh is defined as $(\mathcal{D}, \mathcal{S}) \rightarrow \mathcal{B}_t$, and state change is defined as $(\mathcal{D}_t, \mathcal{S}, \mathcal{B}_{t-1}) \rightarrow \mathcal{B}_t$. In general, state refresh can mitigate error propagation by treating each turn independently, potentially correcting earlier mistakes. However, for complex and lengthy dialogues, using the entire dialogue history may introduce unnecessary complexity and hinder comprehension. State change reduces these problems by focusing only on the current turn and the previously tracked state. In this work, we adopt the state change paradigm due to its efficiency and practical effectiveness.

4 Methodology

Figure 2 shows an overview of our proposed method. It consists of the following two stages:

1. In the first stage, we utilize standard instruction tuning to improve the LLM's ability to solve DST.

2. In the second stage, we construct preference data based on the generation space of the tuned model and the reuse of SFT data, and then perform preference optimization on the tuned model to correct errors and strengthen weaknesses.

4.1 Instruction Tuning

We utilize standard instruction tuning (Wei et al., 2021) to improve the LLM's ability on the DST task. Unlike other fine-tuning approaches, instruction tuning provides more explicit and detailed

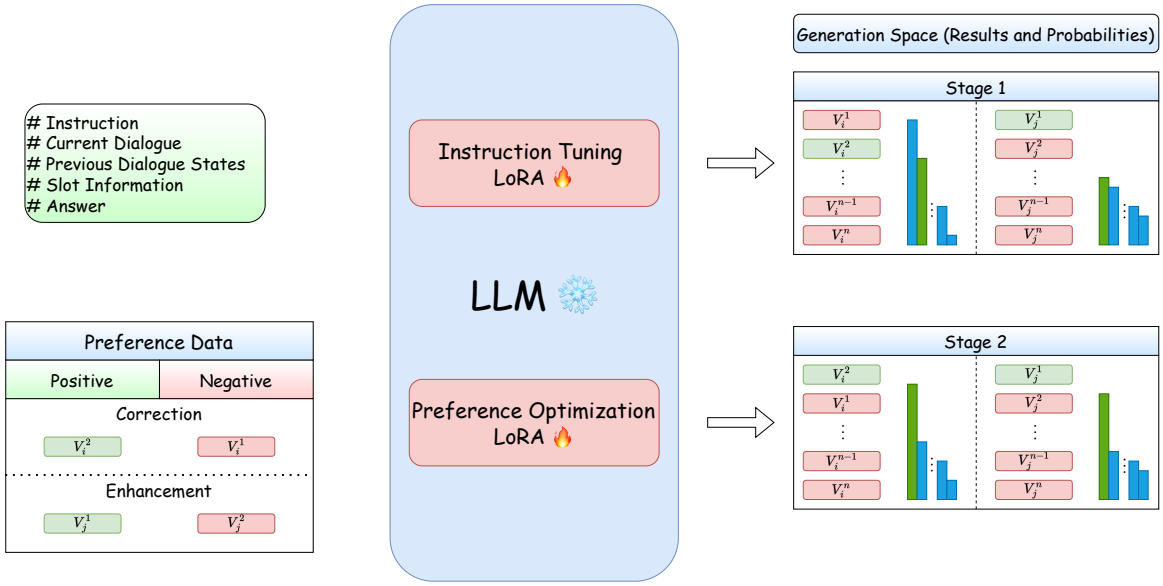


Figure 2: An illustration of our proposed method, which consists of two stages: (1) Instruction Tuning and (2) Preference Optimization. In the first stage, we apply standard instruction tuning with LoRA to enhance the model's ability in DST. In the second stage, we construct preference data based on the model's generation space and the reuse of SFT data, and then apply preference optimization for further improvement.

prompts to guide the model in solving the task, and the design of prompts has a significant impact on model performance. An illustration of the instruction tuning prompt is shown in Figure 6. The prompt consists of three parts: (1) instruction prompt, (2) input prompt, (3) output prompt.

Instruction Prompt In this work, a fixed instruction is used as the instruction prompt to explain the goal of the DST task. Since our work focuses solely on the DST task, using a fixed instruction prompt is acceptable and easy to implement, although some studies (Wei et al., 2021; Chung et al., 2024) have indicated that diverse instruction prompts can improve the performance and robustness of the model across various tasks.

Input Prompt Input prompt consists of three components: current dialogue information, previous dialogue state, and slot information. We represent the dialogue history using the previous dialogue state for lower costs and easier status tracking. The previous dialogue state includes all active slots from earlier turns, rather than only the current slot. If there are no active slots in the previous dialogue state, it will be stated in the prompt. The slot information includes the slot name along with a detailed description. For enumerable slots, all possible values will be listed. If no relevant slot information is present in the current turn, the model will be instructed to output "none". In each dialogue turn, we will construct the input prompt for all possible slots.

Output Prompt Output prompt is the desired value of the slot, which can be selected from the values enumerated in the slot information or derived from the dialogue information.

For cost efficiency, we adopt parameter-efficient fine-tuning (PEFT) to train the model. Specifically, we utilize Low-Rank Adaptaion (LoRA) (Hu et al., 2021). Unlike traditional full-parameter fine-tuning, LoRA freezes the parameters of the base model and injects trainable low-rank decomposition modules into each layer of the model. In traditional fine-tuning, the trainable parameters consist of a weight matrix $W_0 \in \mathbb{R}^{d\times k}$, which can be directly updated based on the gradient. In LoRA, the trainable modules are two weight matrices: $A \in \mathbb{R}^{d\times r}$, $B \in \mathbb{R}^{r\times k}$, and the weight is updated by $\Delta W = BA$. In brief, the weight update is represented as $W = W_0 + \Delta W = W_0 + BA$, where W_0 remains frozen and BA is trainable. For the original output $h = W_0 x$, the forward pass of LoRA is as follows:

$$h = W_0 x + \Delta W x = W_0 x + BAx \quad (1)$$

During the training process, we concatenate the three prompts mentioned above and generate the output prompt from the concatenation of the instruction prompt and input prompt in an autoregres-

sive way. We employ teacher forcing to train the model and utilize cross-entropy loss to minimize the negative log-likelihood of the correct output prompt:

$$\mathcal{L}_1 = -\sum_{j}^{N}\sum_{i}^{L_j} \log p(v_j^i|\mathcal{I}, \mathcal{D}_t, \mathcal{B}_{t-1}, S_j) \quad (2)$$

where N represents the number of all slots, L_j represents the length of V_j, v_j^i represents the i-th word in the value V_j, $\mathcal{I}$ represents instruction prompt.

4.2 Preference Data Construction

In stage 2, we perform additional preference optimization on the instruction-tuned model. Unlike previous works (Stiennon et al., 2020; Ouyang et al., 2022; Song et al., 2024) that focus on human preferences, which are subjective and vary from person to person. The preferences in our work are clear and fixed. Given the goal of the DST task, the preference for this task is only the correct dialogue state. Specifically, the preference in this work refers to the correct value for each corresponding slot.

We construct positive and negative pairs to represent preferences. The positive is the correct value, and the negative is the incorrect value. The correct value is unique and can be obtained from the annotation, whereas the incorrect value is diverse and hard to select. To address this, we select the incorrect value from the model's generation space to limit the uncertainty of the incorrect value. We utilize beam search to express the model's generation space and select the incorrect value with high probability from this generation space.

The positive and negative pairs consist of two types of data: (1) **Correction**: The value with the highest probability is incorrect, but the correct value is in the expressed generation space. (2) **Enhancement**: The value with the highest probability is correct, but its probability is significantly lower than the other correct values. The first type of data is mainly used for correction, focusing on exploring the correct candidates within the generation space that are not ranked as the top-1 candidate. The second is mainly used for enhancement, aiming to reinforce the correct value's position as the highest-probability candidate within the generation space. The correction data can be easily constructed by checking the generation space, taking the correct value as the positive sample and the highest-probability incorrect value as the negative sample. For the enhancement data, we utilize a one-sided confidence interval to identify correct values with low probabilities. If the generation probability of a top-1 correct candidate falls outside the confidence interval, we regard it as a target for enhancement. The formulation of the one-sided confidence interval is as follows:

$$P \in [\overline{p} - Z_{\frac{\alpha}{2}}\frac{\sigma}{\sqrt{n}}, 0) \quad (3)$$

where $\overline{p}$ represents the mean probability of the correct values, σ represents the standard deviation of the correct value probabilities, α represents the confidence level, and $Z_{\frac{\alpha}{2}}$ represents the corresponding Z-value obtained from the standard normal distribution table.

4.3 Preference Optimization

In this work, we utilize preference optimization to refine the model's generation space by increasing the generation probability of the correct values while suppressing the generation probability of incorrect ones. Specifically, Direct Preference Optimization (DPO) (Rafailov et al., 2024) is employed for model training in stage 2. The loss function used in DPO is formulated as follows:

$$\begin{aligned}\mathcal{L}_{DPO}(\pi_\theta, \pi_{ref}) = -\mathbb{E}_{(x,y_{pos},y_{neg})}\Big[\log\sigma \\ \Big(\beta\log\frac{\pi_\theta(y_{pos}|x)}{\pi_{ref}(y_{pos}|x)} \\ -\beta\log\frac{\pi_\theta(y_{neg}|x)}{\pi_{ref}(y_{neg}|x)}\Big)\Big]\end{aligned} \quad (4)$$

where π_θ represents the model to be trained, π_{ref} represents the reference model obtained from stage 1, and π_θ is initialized from π_{ref}.

For some data used for error correction, the probability of the positive sample is significantly lower than that of the negative sample. To accelerate model convergence and improve training stability, we incorporate an extra auxiliary loss for the positive sample in addition to the original DPO loss. The auxiliary loss is the same as the one used in instruction tuning described above. The final loss function is defined as follows:

$$\mathcal{L}_2 = \mathcal{L}_1^{pos} + \mathcal{L}_{DPO} \quad (5)$$

5 Experiments

5.1 Datasets

Our experiments are conducted on a widely used task-oriented dialogue dataset, MultiWOZ

Model	Size	MultiWOZ2.1			MultiWOZ2.4		
		1%	5%	10%	1%	5%	10%
TRADE (Wu et al., 2019)	1B	12.58	31.17	36.18	-	-	-
DS2-BART (Shin et al., 2022)		28.25	37.71	40.29	30.55	42.53	41.73
DS2-T5 (Shin et al., 2022)		33.76	44.20	45.38	36.76	49.89	51.05
IC-DST CodeGen (Hu et al., 2022)	10B	20.72	29.62	33.81	21.87	33.16	37.45
SM2-11b (Chen et al., 2023)		38.36	44.64	46.02	40.03	51.14	51.97
LDST (Feng et al., 2023)		-	-	-	46.77	56.48	62.45
IC-DST Codex-davinc (Hu et al., 2022)	100B	43.13	47.08	48.67	48.35	55.43	56.88
RefPyDST (King and Flanigan, 2023)		**47.30**	49.60	50.80	**55.20**	62.30	62.50
Ours (GSPO)	10B	42.89	**49.96**	**53.25**	47.76	**64.00**	**65.75**

Table 1: Few-shot results averaged over three runs on MultiWOZ 2.1 and MultiWOZ 2.4 with 1%, 5%, and 10% of the training data. All results are reported in joint goal accuracy (%), with the best results highlighted in bold.

(Budzianowski et al., 2018). MultiWOZ is a large-scale, multi-domain human-to-human dialogue dataset that contains over 8k dialogues across seven different domains, with detailed turn-level annotations. In this work, we utilize two versions of the MultiWOZ dataset series: MultiWOZ 2.1 and MultiWOZ 2.4. MultiWOZ 2.1 (Eric et al., 2020) addresses the issue of noisy state annotations in the training data of the original MultiWOZ dataset. MultiWOZ 2.4 (Ye et al., 2022) refines the annotations in the validation and test sets based on MultiWOZ 2.1, with the training set annotations remaining the same as MultiWOZ 2.1.

5.2 Metrics

DST is typically evaluated using two metrics: Slot Accuracy (SA) and Joint Goal Accuracy (JGA). SA measures the prediction accuracy for each slot, including those with a value of none. JGA is a turn-level metric that evaluates the prediction accuracy for all slots at each turn. For JGA, a turn is considered correct only if the values of all slots at that turn are predicted accurately. To keep in line with previous work (Wu et al., 2019; Feng et al., 2023), we utilize JGA as the primary metric.

5.3 Baselines

Our method is compared with the following baselines. **TRADE** (Wu et al., 2019) proposes a transferable dialogue state generator that leverages a copy mechanism to facilitate knowledge sharing across different domains during dialogue state generation. **DS2** (Shin et al., 2022) reformulates dialogue state tracking as a dialogue summarization task. It generates synthetic, template-based summaries using specific rules to train a summarization model and recovers dialogue states from summaries. **SM2** (Chen et al., 2023) strengthens the model's robustness across diverse prompts using a meta-learning scheme and introduces an effective retrieval strategy to find ideal examples to improve the LLM's ability in in-context learning. **LDST** (Feng et al., 2023) proposes a LLM-driven DST framework based on instruction-tuned foundation models. **IC-DST** (Hu et al., 2022) formulates DST as a text-to-SQL task and utilizes an in-context learning method to retrieve dialogue states using a prompt that includes some examples. **RefPyDST** (King and Flanigan, 2023) reformulates DST as a Python Programming task and improves the in-context learning performance by retrieving diverse relevant examples.

5.4 Settings

Our method is implemented with Transformers (Wolf et al., 2020) and TRL (von Werra et al., 2020) libraries based on the PyTorch framework. We utilize Llama-3.1-8B (Dubey et al., 2024) as the base model and fine-tune it with LoRA (Hu et al., 2021) in a parameter-efficient way. The target modules trained with LoRA are $q_{proj}, k_{proj}, v_{proj}, o_{proj}$ in each layer. The rank r of LoRA is 8, and the dropout rate is 0.05. The learning rate for the first stage is 1e-4, and 1e-6 for the second stage. The batch size for both stages is 16, with gradient accumulation. All inference and data generation processes are implemented with the VLLM (Kwon et al., 2023) framework. We conduct all experi-

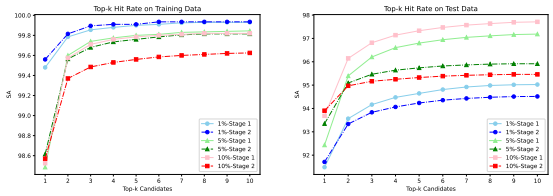
ments with one NVIDIA A100 (40GB) GPU.

6 Results and Analysis

6.1 Main Results

Table 1 shows the results of our proposed method on MultiWOZ 2.1 and MultiWOZ 2.4 on 1%, 5%, and 10% few-shot settings. As mentioned in Section 5.1, MultiWOZ 2.4 refines the annotations in both the validation and test sets, leading to generally higher performance for all methods on this version. Our method outperforms all baseline methods with fewer than 100B parameters. Compared with the best method of similar parameter scale on MultiWOZ 2.4, our method achieves improvements of 0.99% on the 1% setting, 7.52% on the 5% setting, and 5.30% on the 10% setting. Compared to methods with over 100B parameters, our method underperforms RefPyDST on the 1% setting. However, our method achieves better performance on both 5% and 10% settings, indicating that our method benefits more from increased training data. We attribute this phenomenon to the inherent differences between fine-tuning and in-context learning paradigms. In few-shot scenarios, in-context learning methods generally yield good performance with limited data, showing a higher lower bound. In contrast, fine-tuning methods are more effective at scaling with increased data, thus achieving a higher performance upper bound.

6.2 Comparison of Top-k Candidate Hit Rate within Generation Space



(a) Slot accuracy of training. (b) Slot accuracy of test.

Figure 3: Comparison of the top-k hit rate for correct slot predictions within the model's generation space on both training and test data.

To analyze the influence of preference optimization, we conduct a comparative study of the generation space between two stages. We utilize beam search to decode the model outputs, and select the top-10 highest-probability candidates to approximate the model's generation space. As shown in Figure 3, we compare the top-k correct value hit rate within the generation space on both the training and test data. We find that models trained on larger datasets tend to perform worse on the training set but better on the test set. This is because smaller training sets require more iterations to converge. The results show an improved Top-1 hit rate with preference optimization, although the overall hit rate within the generation space drops slightly. This suggests that the model becomes more confident in generating the correct Top-1 value, potentially at the cost of hit rate in its output candidates. Since only the Top-1 result is used for prediction, it is reasonable to prioritize its accuracy even if it slightly reduces the overall hit rate of the generation space.

6.3 Correct Value Probability Distribution

To verify the effectiveness of our method in increasing correct value probabilities, we present the distribution of their generation probabilities in Figure 4. We observe consistent gains in the generation probability of correct values across different data ratio settings, with mean increases of 1.0006 (1%), 0.0137 (5%), and 0.0109 (10%). In addition, improvements in the model's stability and robustness in generating correct values are observed, with the standard deviation of probabilities increasing by 0.0038, 0.0311, and 0.0340 for the 1%, 5%, and 10% settings, respectively. In general, preference optimization in stage 2 enhances the model's confidence and stability in generating correct answers.

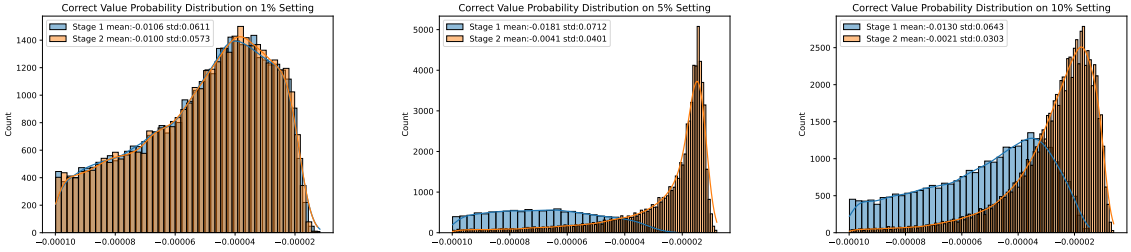
6.4 Ablation Study

	1%	5%	10%
Stage 1	40.98 (-1.91)	46.54 (-3.42)	51.70 (-1.55)
Stage 2	42.89	49.96	53.25
w/o aux loss	43.00 (+0.11)	48.01 (-1.95)	53.01 (-0.24)
w/o enhance	42.01 (-0.88)	46.37 (-3.59)	51.79 (-1.46)
w/o filter	41.83 (-1.06)	47.45 (-2.51)	51.43 (-1.82)

Table 2: Ablation study of the two-stage preference optimization, auxiliary loss, and preference data construction on MultiWOZ 2.1. Results are reported in joint goal accuracy (%).

Table 2 presents ablation studies on MultiWOZ 2.1 from three aspects: (1) Effectiveness of two-stage preference optimization, (2) Influence of auxiliary loss, (3) Method of preference data construction.

Effectiveness of two-stage preference optimization Experimental results demonstrate that the Stage 2 model outperforms Stage 1 by 1.91%,



(a) 1% data probability distribution. (b) 5% data probability distribution. (c) 10% data probability distribution.

Figure 4: Correct value probability distribution of two stages.

3.42%, and 1.55% on the 1%, 5%, and 10% settings, respectively, indicating that the instruction-tuned model can be further improved via preference optimization. Preference optimization on positive (correct) and negative (incorrect) pairs in the generation space enhances the model's ability to distinguish between candidate values and generate the most probable correct value.

Influence of auxiliary loss Without additional auxiliary loss, the result on the 1% setting slightly decreases by 0.11%, while the results of both 5% and 10% settings show an increase by 1.95% and 0.24%, respectively. Our goal of additional auxiliary loss is to increase the probability of the correct value with low probability. However, the model requires more iterations to converge on the 1% setting. This results in a better fit of the model on the training data, which in turn results in less error correction in the preference data, causing the ineffectiveness of auxiliary loss on the 1% setting.

Method of preference data construction We compare the other two methods of preference data construction. Compared to preference data containing only error correction without enhancement, our preference data construction shows higher and more stable improvement. If there is no filtering of enhancement preference data, too much data does not lead to greater improvement, but rather a decline by 1.06%, 2.51%, 1.82% on the 1%, 5%, 10% settings, respectively. These all indicate that the quality of preference data is the most important, and reasonable selection and filtering are necessary.

6.5 Error Analysis

As shown in Figure 5, we compare the slot errors between two stages. We categorize slot errors into three types: common error, stage 1 error, and stage 2 error. The common error exists in both stages, and the other two only exist in their respective

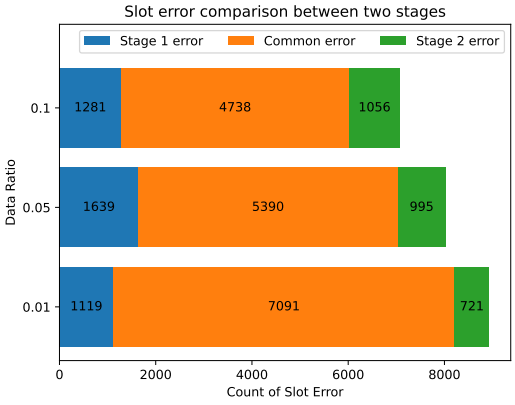


Figure 5: Slot error comparsion between two stages.

stages. There are a large number of common errors in both stages, which can be improved by more training data. As the amount of training data increases from 1% to 10%, the common errors decrease from 7091 to 4738, a reduction of 33.2%. Although we have added some preference data for enhancement during preference optimization, some correct slots in stage 1 are wrong in stage 2. In general, although the preference optimization brings some slot errors, it shows a certain improvement in the overall slot accuracy.

7 Conclusion

In this paper, we propose a novel two-stage training method called generation space preference optimization (GSPO) for the few-shot DST task. Unlike other preference optimization methods, our method is free from extra reward models or preference data. We generate preference data from the model's generation space and reuse of SFT data, and can even customize it based on the capabilities of the model. Experimental results demonstrate the effectiveness of our method.

Limitations

In this paper, we mainly focus on the few-shot DST task. We conduct experiments to validate the effectiveness of our method under limited training data settings, in which the amount of training data limits the model's performance. However, we haven't explored the effectiveness of our method when the amount of training data is not the main bottleneck of the performance. Besides, our method is a two-stage process, and the training data of the first stage is necessary, though it's free of extra reward models and additional preference data.

References

Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. 2024. Explaining neural scaling laws. *Proceedings of the National Academy of Sciences*, 121(27):e2311878121.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026.

Derek Chen, Kun Qian, and Zhou Yu. 2023. Stabilized in-context learning with pre-trained language models for few shot dialogue state tracking. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1551–1564.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, Adarsh Kumar, Anuj Goyal, Peter Ku, and Dilek Hakkani-Tur. 2020. Multiwoz 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 422–428.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*.

Yujie Feng, Zexin Lu, Bo Liu, Liming Zhan, and Xiao-Ming Wu. 2023. Towards llm-driven dialogue state tracking. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 739–755.

Michael Heck, Nurul Lubis, Benjamin Ruppik, Renato Vukovic, Shutong Feng, Christian Geishauser, Hsien-Chin Lin, Carel van Niekerk, and Milica Gasic. 2023. Chatgpt for zero-shot dialogue state tracking: A solution or an opportunity? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 936–950.

Michael Heck, Carel van Niekerk, Nurul Lubis, Christian Geishauser, Hsien-Chin Lin, Marco Moresi, and Milica Gasic. 2020. Trippy: A triple copy strategy for value independent neural dialog state tracking. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 35–44.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. *Advances in Neural Information Processing Systems*, 33:20179–20191.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Yushi Hu, Chia-Hsuan Lee, Tianbao Xie, Tao Yu, Noah A Smith, and Mari Ostendorf. 2022. In-context learning for few-shot dialogue state tracking. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2627–2643.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Brendan King and Jeffrey Flanigan. 2023. Diverse retrieval-augmented in-context learning for dialogue state tracking. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 5570–5585.

Julia Kreutzer, Joshua Uyheng, and Stefan Riezler. 2018. Reliability and learnability of human bandit feedback for sequence-to-sequence reinforcement learning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1777–1788.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Zhaojiang Lin, Bing Liu, Andrea Madotto, Seungwhan Moon, Zhenpeng Zhou, Paul A Crook, Zhiguang Wang, Zhou Yu, Eunjoon Cho, Rajen Subba, et al. 2021a. Zero-shot dialogue state tracking via cross-task transfer. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7890–7900.

Zhaojiang Lin, Bing Liu, Seungwhan Moon, Paul A Crook, Zhenpeng Zhou, Zhiguang Wang, Zhou Yu, Andrea Madotto, Eunjoon Cho, and Rajen Subba. 2021b. Leveraging slot descriptions for zero-shot cross-domain dialogue statetracking. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5640–5648.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3470–3487.

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

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

John Schulman. 2015. Trust region policy optimization. *arXiv preprint arXiv:1502.05477*.

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

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

Jamin Shin, Hangyeol Yu, Hyeongdon Moon, Andrea Madotto, and Juneyoung Park. 2022. Dialogue summaries as dialogue states (ds2), template-guided summarization for few-shot dialogue state tracking. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3824–3846.

Feifan Song, Bowen Yu, Minghao Li, Haiyang Yu, Fei Huang, Yongbin Li, and Houfeng Wang. 2024. Preference ranking optimization for human alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18990–18998.

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.

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`.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. 2019. Transferable multi-domain state generator for task-oriented dialogue systems. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 808–819.

Haoran Xu, Amr Sharaf, Yunmo Chen, Weiting Tan, Lingfeng Shen, Benjamin Van Durme, Kenton Murray, and Young Jin Kim. 2024. Contrastive preference optimization: Pushing the boundaries of llm performance in machine translation. *arXiv preprint arXiv:2401.08417*.

Fanghua Ye, Jarana Manotumruksa, and Emine Yilmaz. 2022. Multiwoz 2.4: A multi-domain task-oriented dialogue dataset with essential annotation corrections to improve state tracking evaluation. In *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 351–360.

Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Global-locally self-attentive dialogue state tracker. *arXiv preprint arXiv:1805.09655*.

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 Illustration of Instruction Tuning Prompt

An example of an instruction tuning prompt is shown in Figure 6. The prompt consists of three parts: (1) Instruction Prompt, (2) Input Prompt, (3) Output Prompt.

B Case Study

As illustrated in Figure 7, we summarize all the error types into four categories: (1) **Partial Prediction** indicates that the DST model misses some dialogue status during the conversation, which often occurs in information-intensive conversations. (2) **Over Prediction** indicates that the DST model overinterprets some dialogue states during the conversation, which often occurs between related slots. (3) **Error Prediction** indicates that the DST model predicts some similar but wrong slot values, which is often caused by the conversation involving rare information. (4) **Goal Change** indicates that the user changes their goals during the conversation, while the DST model doesn't or partially discovers that and inherits the previous dialogue states.

As shown in Figure 8, we analyze the improvement of four error types between the two stages. In general, the DST model shows improvement on all four error types after the preference optimization of the second stage. The main improvement lies in partial prediction and over prediction, accounting for more than 75% of the overall improvement, which indicates that the preference optimization is beneficial to improve the model's sensitivity to related slot information and limit the free interpretation of the model. Goal change also shows great improvement, accounting for 11% to 22% of the overall improvement, which shows that the model's flexibility to handle goal changes is effectively enhanced. Due to the strong ability of current LLMs, there are fewer cases of error prediction, accounting for less than 5% of the whole improvement.

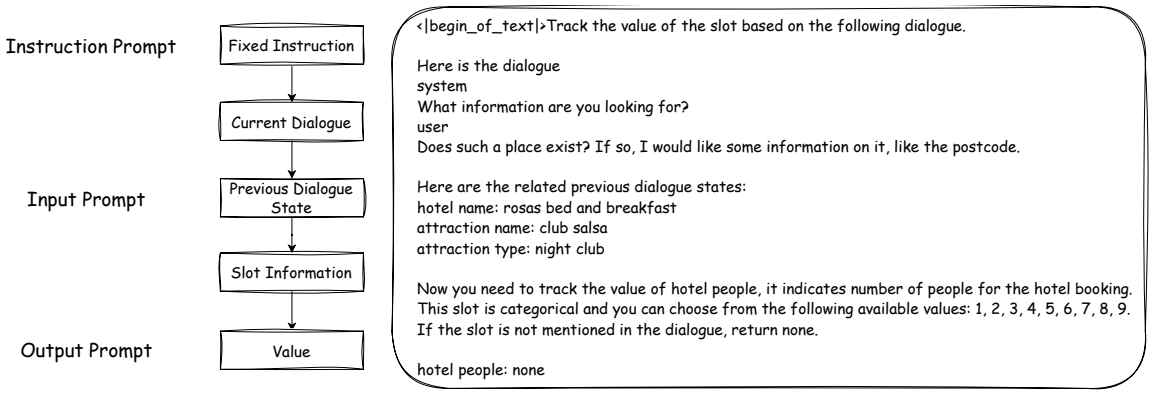
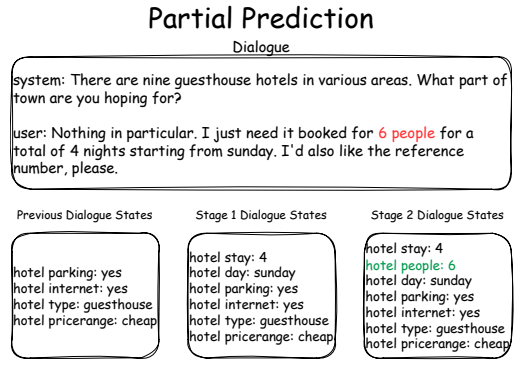
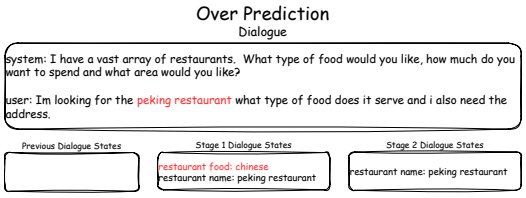


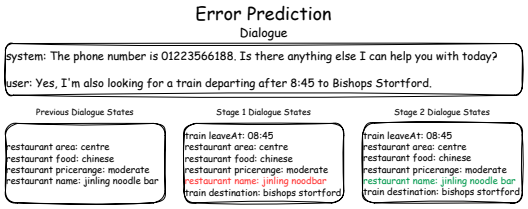
Figure 6: Illustration of instruction tuning prompt. The prompts of instruction tuning consist of three parts: (1) Instruction Prompt, (2) Input Prompt, (3) Output Prompt.



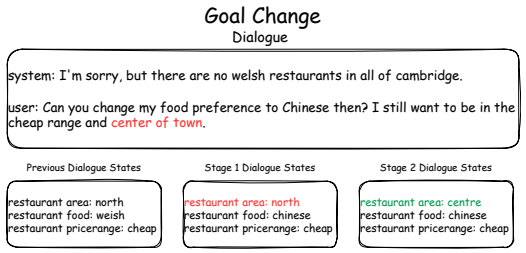
(a) Illustration of partial prediction.



(b) Illustration of over prediction.

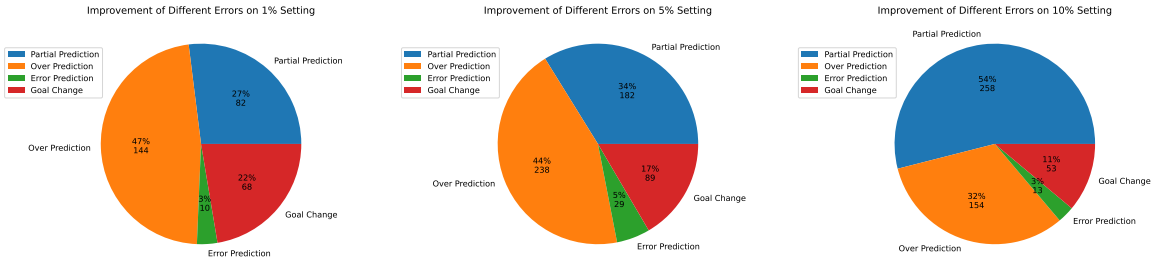


(c) Illustration of error prediction.



(d) Illustration of goal change.

Figure 7: Illustration of four error types.



(a) 1% setting (b) 5% setting (c) 10% setting

Figure 8: The error improvement distribution of the four types.