Revolutionizing the Preference Extractor in Multi-turn Dialogues: From Annotating Disasters to Accurate Preference Extraction

Anonymous ACL submission

Abstract

Identifying user preferences in dialogue systems is a pivotal aspect of providing satisfying services. Current research shows that using large language models (LLMs) to fine-tune a task-specific preference extractor yields excellent results in terms of accuracy and generalization. However, the primary challenge stems from the inherent difficulty in obtaining high-quality labeled multi-turn dialogue data. Accurately tracking user preference transitions across turns not only demands intensive domain expertise and contextual consistency maintenance for annotators (termed **"Annotating Disaster"**) but also complicates model training due to error propagation in sequential dependency learning. Inspired by the observation that multi-turn preference extraction can be decomposed into iterative executions of one-turn extraction processes. We propose a novel dialogue data generation framework named **IterChat**. First, we construct a new data format that categorizes the dialogue data into attributed historical preferences and one-turn dialogues. This reduces the probability of annotation errors and improves annotation efficiency. Then, to generate a high-quality and diverse dialogue dataset, we adopt GPT4 to pre-define the preference slots in the target preference extractor task and then randomly sample the subset of the slots and their corresponding schema values to create the dialogue datasets. Experimental results indicate that fine-tuning or only few-shot prompting with the new dialogue format yields superior performance compared to the original multi-turn dialogues. Additionally, the new data format improves annotator efficiency with a win rate of 28.4% higher than the original multi-turn dialogues.

1 Introduction

A significant challenge in web-based customer support lies in the efficient recognition of user preferences within service dialogues (Malik et al., 2024; Cheng et al., 2021; Shin et al., 2022). Unlike traditional search-based services that process single-shot queries, multi-turn conversations necessitate the identification of dynamically evolving user preferences embedded within the dialogue (Pai et al., 2024; Han et al., 2023; Feng et al., 2021). Recent studies adopt the Large Language model (LLM) to empower the ability to accurately track user preferences in real-time multi-turn user-system dialogues, thereby enabling the provision of tailored services (Xu et al., 2024; Guo et al., 2022; Ravuru et al., 2022). In contrast to the entity extraction task, which focuses on identifying and classifying specific entities within the text, preference extraction involves analyzing and deriving users' emotions, interests, and intentions from the text, requiring a deeper level of comprehension (Yi et al., 2024; Feng et al., 2024). This capability can substantially enhance both the customer experience and the quality of service, while simultaneously supporting business intelligence initiatives for companies (Zhou et al., 2022; Qixiang et al., 2022).

Recent LLM-based preference extraction focused on leveraging prompt engineering combined with few-shot examples (Feng et al., 2023; Xu et al., 2024; Malik et al., 2024). These methods utilize prompts to assign specific roles to LLMs and define the slots to be extracted. However, the few-shot performance of leading LLM, such as GPT-4, still falls short of the state-of-the-art supervised methods (Qi et al., 2023), especially when user queries are broad, ambiguous, and upper funnel (Kim et al., 2024; Heck et al., 2023). Hence, some works start to utilize the fine-tuning technique to train the foundation model with the open source datasets (Feng et al., 2023). However, practical commercial services, such as e-commerce, require a high level of accuracy in identifying complex user preference slots and require customizing additional slots to meet personalized services (Malik et al., 2024), as this directly impacts the ability to provide users

with suitable and satisfactory products.

Therefore, creating a high-quality customized dialogue dataset for a task-oriented domain is crucial to developing a well-performing preference extractor (Li et al., 2023). However, even for experts, tracing the preference transition and annotating an accurate label for the multi-turn conversation is challenging. This is because preference extraction in dialogue data not only requires attention to non-standardized and ambiguous user utterances but also involves continuously adding, removing, or updating preferences based on the user's reactions to system responses. Consequently, acquiring a large-scale golden dataset to train a task-oriented preference extractor is costly and inefficient, a phenomenon we refer to as the **"Annotating Disaster"**. For more details about the annotating disaster, please refer to Section 3.1 and Figure 1. Another significant challenge is that long conversational contexts make model training more difficult, as the cumulative errors in the preference extraction steps tend to accumulate as the dialogue context grows.

To address the aforementioned challenges, we propose a novel dialogue data generation framework named **IterChat**, which is designed to be both annotation-friendly and training-efficient. The framework is inspired by the observation that *multi-turn preference extraction can be decomposed into iterative executions of one-turn extraction processes.* This insight implies that modeling preference evolution through atomic single-turn operations can reduce annotation complexity and minimize error propagation during model training. Specifically, we transform the traditional multi-turn dialogue data into a new data format, which categorizes the dialogue data into historical preferences and the most recent one-turn dialogues. For annotators, the refined dialogue format enables them to annotate the preference transition only once. For fine-tuning LLMs, this new data format does not require long context as input, thereby saving tokens and allowing the model to learn extraction rules from simpler input. Additionally, to overcome the limitation of systematic biases inherent in LLMs and the diversity of the generated dialogue data, we utilize the assistance of LLMs to define the preference slots that need to be extracted for task-oriented preference extractors. We then randomly sample slots and their state values to generate the new form of dialogue datasets.

The main contributions of our work are summarized as follows.

- We transform the traditional multi-turn dialogue data into a new data format that categorizes dialogues into historical preferences and the most recent one-turn dialogues. This refined format reduces annotation errors improves efficiency for annotators, and optimizes the fine-tuning process by simplifying input for LLMs, thus saving tokens and enhancing the learning process.

- We propose a method to overcome the limitations of systematic biases in LLMs and the diversity of generated dialogue data by utilizing LLMs to define task-oriented preference slots. These slots are randomly sampled along with their state values to generate new dialogues, facilitating the development of accurate preference extractors.

- Experimental results demonstrate that fine-tuning or few-shot prompting with the new dialogue format yields superior performance compared to the original multi-turn dialogues. Moreover, this new data format enhances annotator efficiency, achieving a 28.4% higher win rate than the original multi-turn dialogues.

2 Related Works

2.1 Preference Extraction on LLM-based Multi-turn Dialogue

Preference extraction, also known as Dialogue State Tracking (DST), aims to track hidden preferences embedded in conversations to fulfill user goals in task-oriented dialogue systems (Gu and Yang, 2024a,b). With the emergence of LLMs exhibiting remarkable zero-shot capabilities, researchers have begun to explore using LLMs as task-oriented preference extractors. For instance, both (Lee et al., 2021) and (Yang et al., 2023) proposed a prompt-tuning method that leverages domain-specific prompts and contextual information to improve the performance of the preference extraction task. (Xu et al., 2024) constructed chain-of-thought reasoning for the preference extraction task by extracting multiple system-user utterance pairs from dialogue history that alter slot values. (Malik et al., 2024) proposed a framework in which LLMs first summarize user preferences from dialogues, followed by a dynamic example retrieval module that stores and retrieves ICL examples. Recent (Feng et al., 2023; An and Kim, 2023; Moghe et al., 2021) studies have found that few-shot learning performance remains inadequate.

Consequently, research has shifted towards fine-tuning techniques to develop more effective preference extractors. Although various methods focus on the preference extraction task, obtaining large amounts of high-quality task-oriented labeled dialogue data to address complex real-world dialogue scenarios remains a challenge. This is because annotators often face difficulties in annotating multiple turns of slot-value pairs, which can be time-consuming and complex.

2.2 Labeled Dialogue Data for Preference Extraction

Some researchers have contributed a series of labeled datasets for the preference extraction task. For example, MultiWOZ 2.2 (Eric et al., 2019) is a multi-domain task-oriented dialogue dataset that includes more than 10,000 dialogues that span 8 domains. Additionally, the Schema-Guided Dialogue (SGD) (Rastogi et al., 2020) dataset includes over 16,000 conversations between users and virtual assistants, which encompass 26 services in 16 domains, such as events, restaurants, and media. However, with the increasing number of online services providing dialogue interfaces, current open-source datasets struggle to cover all specific scenarios. Moreover, the preference slots available in these open-source datasets are limited, making it challenging for service providers to build an accurate preference extractor to handle varied and changing user preferences. Therefore, we propose a dialogue data generation framework named IterChat to help service providers quickly construct labeled dialogue datasets for their own domains.

3 Preliminaries

This section provides an overview of LLM-based multi-turn dialogue systems and the associated challenges of multi-turn preference extraction.

3.1 Multi-Turn Dialogues and Preference Extraction

In task-oriented dialogue systems, users typically engage in multi-turn interactions with a chatbot to iteratively clarify, adjust, or refine their preferences (Feng et al., 2023). This process can be modeled as a sequence of dialogue pairs: $\{(Q_1, A_1), (Q_2, A_2), \ldots, (Q_T, A_T)\}$, where Q represents the user's input queries, and A represents the chatbot's responses. Each pair represents a single dialogue turn. The dialogue context at turn t includes the entire history of interactions up to that point, incorporating both the user's queries and the chatbot's responses, and is denoted as: $X_t = \{(Q_1, A_1), (Q_2, A_2), \ldots, (Q_t, A_t)\}$. This context plays a critical role in understanding the evolving preferences and intentions of the user.

The primary task in preference extraction for dialogue systems is to identify key pieces of information from a conversation that reflect the user's current preferences (Malik et al., 2024). These preferences are typically expressed through various slots, each representing a specific aspect of the user's intent or requirement. At any given turn t, the dialogue information can be represented as a set of preference slots, each associated with a particular entity value, denoted as: $Y_t = \{(P_1 : \mathbf{V}_{1,t}), (P_2 : \mathbf{V}_{2,t}), \ldots, (P_N : \mathbf{V}_{N,t})\}$, where P_i is the preference slot, and $\mathbf{V}_{i,t}$ is the corresponding values of that slot at turn t. For instance, in an e-commerce scenario, a user might express their preference for a product in terms of a slot such as "$\langle price \rangle$", which indicates the expected price range. The corresponding value for this slot could be something like "less than $50", which specifies the user's preference in more detail. Other common preference slots in such scenarios could include "$\langle color \rangle$", "$\langle brand \rangle$", or "$\langle size \rangle$", each reflecting a specific dimension of the user's choice.

In multi-turn dialogues, large language models (LLMs) are commonly used to extract preference slots from a sequence of question-answer pairs. However, existing LLMs face inherent limitations in retaining long-term memory across extended conversations. This often leads to a phenomenon known as "preference slot oblivion", where the model loses track of earlier preferences as the dialogue progresses, resulting in inconsistencies in its understanding. To address this challenge in preference extraction within multi-turn dialogues, we propose a novel approach in the next section that reorganizes the problem into an incremental preference evolution framework. In this framework, the learning objective for the LLM is to first extract preference slots and values from the most recent one-turn dialogue. Then, it combines the user's historical preferences with the latest preferences from the current dialogue turn to form the most up-to-date user preference.

4 The Proposed Framework

In this section, we provide a detailed explanation of our proposed **IterChat** and corresponding data for-

mat, along with the annotation process and the overall framework for IterChat data generation. This includes an in-depth description of each module involved in the pipeline. As illustrated in Figure 2, the main framework consists of four key modules: the preference schema module, the dialogue sampling module, the annotation module, and the agent tuning module.

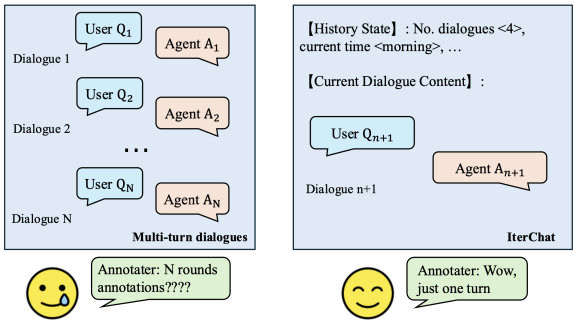

Figure 1: Comparison of multi-turn dialogues and IterChat data

4.1 Incremental Preference Evolution

Given a dialogue sequence up to the $(t+1)$-th turn: $X_{t+1} = \{(Q_1, A_1), \ldots, (Q_{t+1}, A_{t+1})\}$, instead of directly extracting the preference Y_{t+1}, we first summarize the preference information from the previous t-turn dialogue, X_t, to obtain the current preference Y_t. Subsequently, we extract the preference from the most recent one-turn dialogue (Q_{t+1}, A_{t+1}) based on the context Y_t, yielding the preference gain G_{t+1}, which involves updating the preference slots. Finally, we combine the historical preference Y_t with the newly extracted preference gain G_{t+1} to update the current preference, Y_{t+1}.

This iterative framework ensures that, by leveraging both historical preference Y_t and the most recent dialogue turn (Q_{t+1}, A_{t+1}), we can effectively extract the preference gain G_{t+1} and obtain the updated preference Y_{t+1}. This methodology effectively prevents preference slot oblivion, offering a more structured and coherent process for maintaining preference consistency throughout the dialogue. Specifically, the learning objective for the LLM is to extract preference slots and values from the most recent one-turn dialogue, then combine the user's historical preferences (as captured in the History Preference) with the latest preferences from the current dialogue turn to form the most up-to-date user preference. This approach mitigates the problem of preference slot oblivion and ensures that the model can continuously track evolving user needs. Based on this problem definition, we further propose reorganizing multi-turn dialogue data into a new, more efficient format, which will be explained in detail in the next section.

4.2 An Annotate-friendly data format

We reorganize multi-turn dialogue data into a new, more efficient format which consists of two main components:

- **History Preference**: It summarizes the user's preferences over the previous n turns of dialogue, capturing the evolving context and the user's changing preferences.

- **Most Recent One-Turn Dialogue**: It contains the latest user query and chatbot response, reflecting the immediate context of the ongoing conversation.

A comparison between multi-turn dialogues and the IterChat format is illustrated in Figure 1. By adopting this structure, we transform the original multi-turn preference extraction problem into a more manageable incremental preference evolution problem.

In addition, we introduce two new annotation outputs for preference extraction in the IterChat format: "StateGain" and "PreferenceExtraction". "StateGain" represents the information gained from the most recent dialogue turn, highlighting the new insights added to the user's preferences. On the other hand, "PreferenceExtraction" reflects the final set of preference slots after processing user history preference and the latest dialogue preference, representing the chatbot's understanding of the user's current preferences. By using this approach, human annotators only need to annotate the preference slots for the most recent one-turn dialogue, significantly reducing the likelihood of annotation errors and improving efficiency.

4.3 Preference Schema Module

Effectively extracting user preferences involves monitoring the user's shifting goals and the system's responses throughout the dialogues. To maintain consistency in understanding user preferences, it is essential to produce structured outputs. This can be achieved by extracting predefined slot-value pairs from the dialogue context at each turn, ensuring that the chatbot can interpret and act on the user's preferences with clarity and precision.

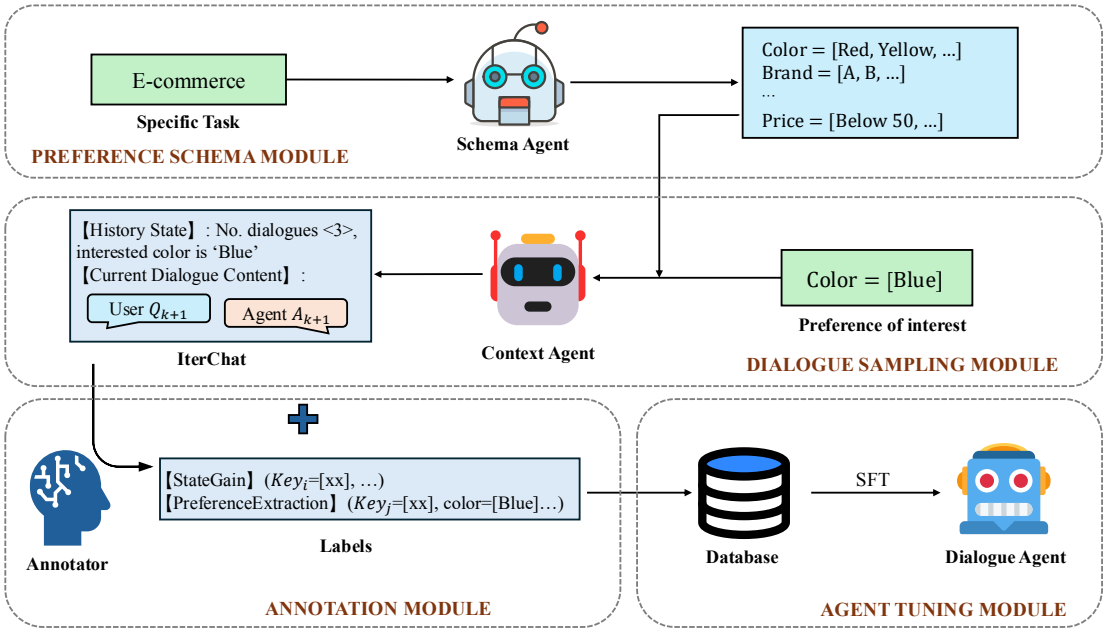

Figure 2: Overview Framework of IterChat Data Generation

In the Preference Schema Module, we first consult a schema agent to gather the necessary information about the specific task at hand. For example, the schema agent might identify the most important factors that influence a user's decision-making process in a particular task, such as budget constraints in an e-commerce scenario or preferred location in a travel planning task. Based on this information, we then refine and formalize a task-oriented preference schema, which defines a set of preference slots relevant to the task. Each preference slot represents a key aspect of user preference that must be captured during the conversation. For instance, in an e-commerce task, slots could include "$\langle price \rangle$", "$\langle brand \rangle$", or "$\langle color \rangle$". Each slot may have a range of possible values, depending on the user's preferences. For example, the "$\langle price \rangle$" slot could have values like "less than $50", "between $100 and $200", or "None". By structuring the preferences in this way, the dialogue system can consistently track and update user preferences across multiple turns, ensuring that the system's responses remain aligned with the user's evolving needs.

4.4 Dialogue Sampling Module

One of the key advantages of the preference schema is that it enables the generation of high-quality user dialogues. In this section, we outline how we leverage the preference schema to construct IterChat data which consists of the user's history state and the most recent one-turn dialogue, with updates to preference slots that can be tailored to our needs.

The dialogue sampling process begins with the construction of the "history state", which includes details such as the number of past dialogues, the current time, and other relevant context information that reflects the conversation's progression. The "history state" is generated by randomly sampling detailed contextual information using a Context Agent. This agent is responsible for selecting a variety of factors that summarize the history of the conversation, ensuring the generated state is diverse and representative of different conversational scenarios. Additionally, the Context Agent is tasked with sampling the current state, which includes preference slots of particular interest. These preference slots could correspond to factors like price, color, or brand in an e-commerce scenario, or location and date in a travel planning context.

With both the history state and current state in hand, the Context Agent generates the most recent one-turn dialogue, reflecting the update of preference slots. For instance, if the history state is:

('price' = ['less than $50'])

the current state is:

('price' = ['less than $50'], 'color' = ['red'])

the corresponding one-turn dialogue could be:

user: “I like red."

This approach allows for a smooth transition between states and generates natural dialogues that are contextually relevant and reflective of evolving user preferences. By following this pipeline, we can efficiently sample large quantities of IterChat data, each containing a rich set of preference slots that are of interest. This approach not only supports the generation of diverse dialogues but also ensures that each dialogue remains relevant to the user's preferences. Please refer to Appendix B for the prompt.

4.5 Annotation Module

In addition to this structure, we propose a new annotation output for preference extraction in the IterChat format, defined by two components:

- **StateGain:** This represents the information gained from the most recent dialogue. Specifically, it quantifies how much new information has been added to the user's preference profile after processing the latest interaction. The StateGain helps identify whether the most recent dialogue has refined or introduced new preferences.
- **PreferenceExtraction:** This denotes the final extraction of preference slots based on the dialogue so far, encompassing both the historical context and the latest one-turn. It represents the chatbot's understanding of the user's current preferences after incorporating the entire dialogue history and the most recent interaction. The PreferenceExtraction result is a comprehensive set of preference slots, each with an associated value, reflecting the user's intentions.

Note that this process involves more than just expanding or summing up slots. In real-world applications, it requires adhering to multiple inheritance rules to ensure the consistency and accuracy of preference updates.

By using the IterChat data format and its corresponding annotations, we can generate diverse and high-quality raw data while enhancing annotator efficiency, as only one-turn annotations are required, compared to the multiple-turn annotations typically needed for full dialogues.

4.6 Agent Tuning Module

Once the annotated IterChat data is collected and stored in our database, it is used to further fine-tune our dialogue agent in a supervised manner, thereby enhancing its ability to understand and respond to user preferences. The IterChat data format plays a crucial role in optimizing this fine-tuning process by providing a simplified, structured input for large language models (LLMs). This structured format not only reduces the token usage but also streamlines the learning process, allowing the agent to focus more on relevant content without the need to process lengthy multi-turn dialogues.

By leveraging the concise, one-turn structure of the IterChat format, we significantly reduce the computational overhead, allowing for more efficient training. Moreover, this format ensures that the agent can better capture evolving preferences by maintaining a clear, consistent representation of user goals and system actions. In the next section, we demonstrate that fine-tuning or few-shot prompting with IterChat yields superior performance compared to the original multi-turn dialogues.

5 Experiments

5.1 Experimental Setup

Datasets. *MultiWOZ 2.1* is a widely used, large-scale, multi-domain task-oriented dialogue (TOD) dataset with several revised iterations. For our study, we focus on the 'Hotel' domain of MultiWOZ due to its largest preference slots, which significantly increases task complexity. *Foodie (IterChat)*, is a dataset designed for food preference extraction tasks, constructed using the IterChat data format. This dataset comprises 3,500 samples generated using GPT-4, with annotations validated by experienced data annotators to ensure quality. *Foodie (multi-turn)*, a dataset where transforms the History State in IterChat into multi-turn dialogues using GPT-4. These dialogues were then manually reviewed and corrected for logical consistency. Due to the complexity and effort on this process, we curated a subset of approximately 300 labeled dialogues for training and testing purposes.

Evaluation Metrics. For easy comparison, we adopt the following metrics to evaluate the accuracy of preference extraction: (1) Exact Match (EM): Measures the percentage of predictions that exactly match the true labels; (2) F1 scores: Harmonic mean of precision and recall, balancing both in one metric; (3) Filter Edit Distance (FED) (Li et al., 2024): Counts the minimum changes needed to convert one string to another.

	Setup	Paradigm	Foodie			MltiWOZ-H		
			EM	F1	FED	EM	F1	FED
few-shot prompting	PERAL-GPT4	ICL@2 multi-turn	0.32	0.8158	1.638	**0.5674**	**0.7863**	**1.081**
	PERAL-GPT4	ICL@2 IterChat	**0.501**	**0.9021**	**0.8157**	0.5379	0.7244	1.282
	NL2API-GPT4	ICL@2 multi-turn	0.3333	0.806	1.233	**0.5538**	**0.7399**	**1.085**
	NL2API-GPT4	ICL@2 IterChat	**0.5666**	**0.8875**	**0.6333**	0.5284	0.7268	1.276
full-parameter fine-tuning	Llama-7B	multi-turn	0.1667	0.5686	1.9	0.4273	0.8981	0.9196
	Llama-7B	IterChat	**0.3333**	**0.7002**	**1.1833**	**0.7837**	**0.9363**	**0.3162**
	Llama-13B	multi-turn	0.6666	0.8914	0.3833	0.4704	0.9186	0.8436
	Llama-13B	IterChat	**0.8166**	**0.946**	**0.2166**	**0.8181**	**0.9537**	**0.2542**

Table 1: Preference Comparison between Multi-turn Dialogue and Iterchat

Baseline models. We adopt popular open-source and close-source LLMs as baseline models for experiments, including GPT4 (Achiam et al., 2023), LLaMA (Touvron et al., 2023), Qwen (Bai et al., 2023) and Pangu (Ren et al., 2023).

Baseline Method. *PEARL* (Malik et al., 2024) introduces a framework where large language models (LLMs) first summarize user preferences from dialogues, followed by a dynamic example retrieval module that stores and retrieves in-context learning (ICL) examples. *NL2API*, another baseline, employs an LLM to take the demonstrations and preference slots in the prompt and then directly identifies the final preference label. Please refer to Appendix A and Appendix B for implementation details and the prompt.

5.2 Multi-turn Dialogue vs. IterChat

We evaluated the performance of multi-turn dialogues and IterChat under both few-shot prompting and full-parameter fine-tuning scenarios. In the few-shot setting, we used GPT-4 as the foundation model. For the Multiwoz dataset, we employed 1,000 samples as the test set and 2 samples as demonstrations, while for the Foodie (IterChat/multi-turn) dataset, we used 60 samples as the test set and 2 samples as demonstrations. In the full-parameter fine-tuning scenario, we utilized 3,000 samples for training and 1,000 samples for testing on the Multiwoz dataset, whereas, for the Foodie dataset (IterChat/multi-turn), we used 228 samples for training and 260 samples for testing. Based on the results in Table 1, we observed the following: (1) In the few-shot prompting setting, IterChat did not show significant advantages over the original multi-turn dialogues. This is because IterChat requires iteratively editing the preference set based on each user utterance, using operations such as adding, removing, or updating preference slot values. (2) However, in the full-parameter fine-tuning scenario, IterChat significantly outperformed the original multi-turn dialogues. This is because IterChat's data format, which includes only the historical state and the current dialogue, makes it easier for the model to learn preference transitions. In contrast, multi-turn dialogues involve tracking preferences across multiple turns, which often leads the model to converge to suboptimal solutions.

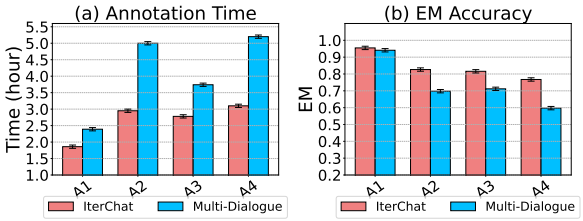

Figure 3: Annotation Efficiency and Accuracy.

5.3 Annotation Efficiency and Accuracy

To demonstrate the annotation efficiency of our proposed IterChat framework, we conducted an experiment comparing the efficiency and accuracy of human annotators when using IterChat versus traditional multi-turn dialogue context annotations. The results of the experiment, based on 288 samples for each format, are shown in Figure 3 with the following insights: (1) **Annotation Time:** On average, the IterChat format significantly reduced the time spent on annotation, with an average time of 2.92 hours compared to 4.08 hours for the multi-turn dialogue format. This represents a 28.4% reduction in annotation time, highlighting the efficiency gains of using the IterChat format. (2) **Annotation Accuracy (EM):** Despite the reduction in time, the IterChat format maintained or even improved accuracy in terms of Exact Match (EM) scores. The average EM accuracy for IterChat was 84.37%, which is 11.95% higher than the 73.42% achieved with the multi-turn dialogue format.

5.4 Generalization Ability

To evaluate the generalization ability of our proposed IterChat, we conducted an experiment to assess how well various base models, after fine-tuning on the IterChat dataset, could generalize to unseen data. In this experiment, we fine-tuned three distinct base models of varying sizes on 3000 samples from the IterChat dataset. The selected models were Llama-13B, Qwen-32B, and PanGu-38B. Each of these models was fine-tuned on the IterChat dataset and then evaluated on a held-out test set of size 200 to measure EM. The result is shown in Table 2.

Model	Llama-13B	Qwen-32B	PanGu-38B
EM	84.0%	86.5%	83.0%

Table 2: Model Performance (EM)

After fine-tuning the IterChat dataset, the Llama-13B model achieved an EM accuracy of 84.0%. It shows that even a general-purpose model with no specialized training in dialogue data can still benefit from the IterChat format. The Qwen-32B model achieved an EM accuracy of 86.5%. This improvement over Llama-13B suggests that fine-tuning IterChat data helps improve its performance, possibly due to its inherent ability to handle context-rich information better than general-purpose models. On the other hand, PanGu-38B, despite its larger size, achieved the worst performance with an EM accuracy of 83.0%. Despite its larger parameter count, PanGu-38B might not be as well-suited for the dialogue-based nature of the IterChat data.

The results of this experiment demonstrate the effectiveness of the IterChat format in enhancing the generalization capabilities of various base models. Across the three models tested, we observed consistent improvements in EM accuracy after fine-tuning on IterChat data.

5.5 Training Scaling

In this experiment, we demonstrate the scalability of our proposed IterChat dataset by evaluating how well fine-tuned large language models (LLMs) of different sizes perform as the amount of IterChat data increases. Specifically, we test the EM accuracy of two models: Llama-7B and Llama-13B after fine-tuning on IterChat datasets of different sizes, ranging from 100 to 3000 samples. After fine-tuning, both models were evaluated on a consistent set of 200 test samples to measure EM accuracy. The result is shown in Figure 4.

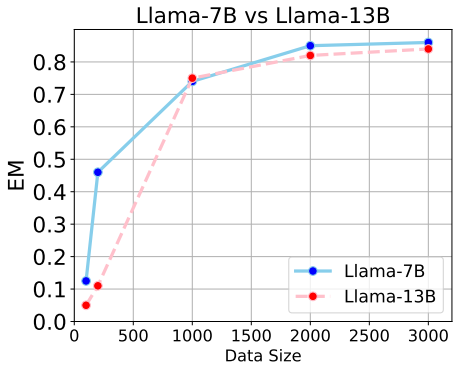

Figure 4: Scaling of Llama with different data size

It can be observed that both models show a clear correlation between the size of the training data and the EM accuracy. Llama-7B, as a smaller model, benefits significantly from smaller datasets, achieving substantial performance improvements with even relatively few samples. On the other hand, Llama-13B requires a larger amount of data to fully leverage its larger capacity, with improvements becoming more noticeable after around 1000 samples. In conclusion, IterChat offers strong scalability, and both smaller and larger models benefit from increased training data.

6 Conclusions

In this work, we have presented IterChat, a novel framework for generating high-quality dialogue datasets that address the challenges of "Annotating Disaster" and "Preference Oblivion" in multi-turn dialogue preference extraction. By decomposing the task into more manageable one-turn preference extractions, IterChat enhances both the accuracy and efficiency of dialogue data annotation. The new format, which categorizes historical preferences separately from one-turn dialogues, reduces annotation errors and simplifies model training by alleviating the issues of error propagation across multiple turns. Our experiments show that fine-tuning or few-shot prompting with the IterChat format yields significantly improved performance in preference extraction tasks compared to the traditional multi-turn dialogue format. These findings underscore the potential of IterChat to both streamline the annotation process and improve the generalization capabilities of LLMs in dialogue systems.

Ethical Statement

We have hired full-time data annotators and purchased the necessary work insurance for them. We strictly adhere to the regulation that the daily working hours shall not exceed 8 hours. Moreover, we offer salaries that are not lower than the market average.

Limitations

Dataset Construction and Generalization: Although the paper claims that the IterChat format can enhance the generalization capabilities of various base models, the generalization ability evaluation is limited. The experiments mainly focus on a few datasets (such as the 'Hotel' domain of MultiWOZ and the Foodie dataset), and it is uncertain whether the results can be extended to other domains and more complex real-world scenarios. Also, the datasets used for evaluation may not fully cover the diversity of user preferences and dialogue situations in practice.

Annotation and Data Generation: The annotation process in IterChat still requires human effort, and although it reduces the annotation time and improves accuracy compared to multi-turn dialogues, it may still be resource-intensive for large-scale datasets. Additionally, the data generation process relies on GPT-4 to pre-define preference slots and sample values, which may introduce biases from GPT - 4 itself. Also, the assumption that multi-turn preference extraction can be decomposed into one-turn extraction processes might not hold true for all types of dialogues, especially those with highly complex and intertwined preference expressions.

Acknowledgments

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Jinwon An and Misuk Kim. 2023. Dialogue specific pretraining tasks for improved dialogue state tracking. *Neural Processing Letters*, 55(6):7761–7776.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Xusen Cheng, Ying Bao, Alex Zarifis, Wankun Gong, and Jian Mou. 2021. Exploring consumers' response to text-based chatbots in e-commerce: the moderating role of task complexity and chatbot disclosure. *Internet Research*, 32(2):496–517.

Mihail Eric, Rahul Goel, Shachi Paul, Adarsh Kumar, Abhishek Sethi, Peter Ku, Anuj Kumar Goyal, Sanchit Agarwal, Shuyang Gao, and Dilek Hakkani-Tur. 2019. Multiwoz 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines. *arXiv preprint arXiv:1907.01669*.

Yue Feng, Yang Wang, and Hang Li. 2021. A sequence-to-sequence approach to dialogue state tracking. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1714–1725.

Yujie Feng, Xu Chu, Yongxin Xu, Guangyuan Shi, Bo Liu, and Xiao-Ming Wu. 2024. Tasl: Continual dialog state tracking via task skill localization and consolidation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1266–1279.

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.

Ming Gu and Yan Yang. 2024a. Plan, generate and complicate: Improving low-resource dialogue state tracking via easy-to-difficult zero-shot data augmentation. *arXiv preprint arXiv:2406.08860*.

Ming Gu and Yan Yang. 2024b. A two-dimensional zero-shot dialogue state tracking evaluation method using gpt-4. *arXiv preprint arXiv:2406.11651*.

Jinyu Guo, Kai Shuang, Jijie Li, Zihan Wang, and Yixuan Liu. 2022. Beyond the granularity: Multi-perspective dialogue collaborative selection for dialogue state tracking. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2320–2332.

Ridong Han, Tao Peng, Chaohao Yang, Benyou Wang, Lu Liu, and Xiang Wan. 2023. Is information extraction solved by chatgpt? an analysis of performance, evaluation criteria, robustness and errors. *arXiv preprint arXiv:2305.14450*.

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.

Hannah Kim, Kushan Mitra, Rafael Li Chen, Sajjadur Rahman, and Dan Zhang. 2024. Meganno+: A human-llm collaborative annotation system. *arXiv preprint arXiv:2402.18050*.

Chia-Hsuan Lee, Hao Cheng, and Mari Ostendorf. 2021. Dialogue state tracking with a language model using schema-driven prompting. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4937–4949.

Bo Li, Gexiang Fang, Yang Yang, Quansen Wang, Wei Ye, Wen Zhao, and Shikun Zhang. 2023. Evaluating chatgpt's information extraction capabilities: An assessment of performance, explainability, calibration, and faithfulness. *arXiv preprint arXiv:2304.11633*.

Jinyi Li, Yihuai Lan, Lei Wang, and Hao Wang. 2024. Pctoolkit: A unified plug-and-play prompt compression toolkit of large language models. *arXiv preprint arXiv:2403.17411*.

Vijit Malik, Akshay Jagatap, Vinayak Puranik, and Anirban Majumder. 2024. Pearl: Preference extraction with exemplar augmentation and retrieval with llm agents. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 1536–1547.

Nikita Moghe, Mark Steedman, and Alexandra Birch. 2021. Cross-lingual intermediate fine-tuning improves dialogue state tracking. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1137–1150.

Liu Pai, Wenyang Gao, Wenjie Dong, Lin Ai, Ziwei Gong, Songfang Huang, Li Zongsheng, Ehsan Hoque, Julia Hirschberg, and Yue Zhang. 2024. A survey on open information extraction from rule-based model to large language model. *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 9586–9608.

Ji Qi, Chuchun Zhang, Xiaozhi Wang, Kaisheng Zeng, Jifan Yu, Jinxin Liu, Lei Hou, Juanzi Li, and Xu Bin. 2023. Preserving knowledge invariance: Rethinking robustness evaluation of open information extraction. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5876–5890.

Gao Qixiang, Guanting Dong, Yutao Mou, Liwen Wang, Chen Zeng, Daichi Guo, Mingyang Sun, and Weiran Xu. 2022. Exploiting domain-slot related keywords description for few-shot cross-domain dialogue state tracking. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2460–2465.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8689–8696.

Lohith Ravuru, Seonghan Ryu, Hyungtak Choi, Haehun Yang, and Hyeonmok Ko. 2022. Multi-domain dialogue state tracking by neural-retrieval augmentation. In *Findings of the Association for Computational Linguistics: AACL-IJCNLP 2022*, pages 169–175.

Xiaozhe Ren, Pingyi Zhou, Xinfan Meng, Xinjing Huang, Yadao Wang, Weichao Wang, Pengfei Li, Xiaoda Zhang, Alexander Podolskiy, Grigory Arshinov, et al. 2023. Pangu-{\Sigma}: Towards trillion parameter language model with sparse heterogeneous computing. *arXiv preprint arXiv:2303.10845*.

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.

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

Lin Xu, Ningxin Peng, Daquan Zhou, See-Kiong Ng, and Jinlan Fu. 2024. Chain of thought explanation for dialogue state tracking. *arXiv preprint arXiv:2403.04656*.

Yuting Yang, Wenqiang Lei, Pei Huang, Juan Cao, Jintao Li, and Tat-Seng Chua. 2023. A dual prompt learning framework for few-shot dialogue state tracking. In *Proceedings of the ACM Web Conference 2023*, pages 1468–1477.

Zihao Yi, Jiarui Ouyang, Yuwen Liu, Tianhao Liao, Zhe Xu, and Ying Shen. 2024. A survey on recent advances in llm-based multi-turn dialogue systems. *arXiv preprint arXiv:2402.18013*.

Zijun Zhang. 2018. Improved adam optimizer for deep neural networks. In *2018 IEEE/ACM 26th international symposium on quality of service (IWQoS)*, pages 1–2. Ieee.

Yihao Zhou, Guoshuai Zhao, and Xueming Qian. 2022. Dialogue state tracking based on hierarchical slot attention and contrastive learning. In *Proceedings of the 31st ACM international conference on information & knowledge management*, pages 4737–4741.

A Implementation Details.

Baseline Settings. We evaluate the effectiveness of the IterChat data format compared to the original long-context format using two approaches: (i) few-shot prompting, each model directly processes the concatenated text of the task instruction and dialogue content (either IterChat or multi-turn dialogue) as the input prompt, generating the final set of preference labels. (ii) Full-parameter fine-tuning,

we following the conventional one-dialogue-one-sample manner which the adopted baseline models are all causal LLMs.

Parameter settings during full-parameter fine-tuninng. We conducted full-parameter fine-tuning using distributed computing, employing a total of 8 distributed nodes. Each node was equipped with 72 CPU cores and 8 Huawei Ascend GPU, each with 64 GB of memory. During the fine-tuning phase, we set the batch size to 48 and trained the model for 5 epochs. For fine-tuning on small datasets (data size < 1,000), we used a learning rate of 4e-5, while for larger datasets (data size > 1,000), we set the learning rate to 5e-5. The Adam optimizer (Zhang, 2018) was used throughout all training processes.

B Prompts

We provide the exact prompts utilized in our experiments on the MultiWoZ-H dataset, with similar prompts applied to the Foodie dataset.

IterChat Data Generation:
Human:
<purpose>
You are a dialogue data production assistant. You need to simulate a conversation between a user and a system, including two parts: the previous state and the current utterance. The previous state summarizes the user's previous context, and the current utterance represents the most recent exchange between the user and the system.
</purpose>

<operation-principles>
1. Generate the previous state:
- Mandatory state {state0}
- Optional state {state1}
2. Generate the current utterance:
- The current utterance represents one round of conversation between [User] and [System].
- [System] response should be relevant to the current state.
- [User] dialogue should meet the following requirements:
{query}
<operation-principles>

<state-definition>
State schema:
- Key: [Area, Booking Date, Price Range, Star Rating, Type, etc.]
- Value: Array of valid options from the filter table.
<state-definition>

<schema format="key:[allowed_values]">
{
"Area": ["Centre", "East", "North", "South", "West"],
"Booking Date": ["Monday",...,"Sunday"],
"Price Range": ["Expensive", "Cheap", "Moderate"],
"Type": ["Guesthouse", "Hotel"],
"Star Rating": ["0"..."5"]
}
</schema>

<processing-rules>
<rule>Format requirements:
Previous State: {Fill in the state content here}
Current Utterance: {Fill in the latest dialogue here}
</Rule>
<Rule>Dialogue requirements:
- The first turn is the system's response.
- The second turn is the user's dialogue.
</Rule>
<processing-rules>

Process:
<examples>
-- {example1}
-- {example2}
-- {example3}
</examples>

Please generate {numb} sets of data based on the examples:

Figure 5: Prompt for new data format generation

```
# Prompt for IterChat:
Human:
<purpose>
You are a state tracking assistant that dynamically updates hotel preferences by merging historical state with new information from the latest dialogue turn.
</purpose>

<operation-principles>
1. State Inheritance: Carry forward unchanged slots from historical state
2. State Mutation:
  - Add: New slots mentioned in current dialogue
  - Modify: Overwrite existing slots with new values
  - Delete: Remove slots explicitly revoked (e.g., "never mind about parking")
3. State Validation: Enforce schema compatibility before final output
</operation-principles>

<schema format="key:[allowed_values]">
{
 "hotel-pricerange": ["expensive", "cheap", "moderate"],
 "hotel-type": ["guesthouse", "hotel"],
 "hotel-parking": ["free", "no", "yes"],
 "hotel-bookday": ["monday",...,"sunday"],
 "hotel-bookpeople": ["1"..."8"],
 "hotel-bookstay": ["1"..."8"],
 "hotel-stars": ["0"..."5"],
 "hotel-internet": ["free", "no", "yes"],
 "hotel-name": "free-text",
 "hotel-area": ["centre", "east", "north", "south", "west"]
}
</schema>

<processing-rules>
<rule>Priority: Current dialogue > Historical state</rule>
<rule>For implicit changes:
- "Actually..." → Modify existing slot
- "Instead of X..." → Replace previous value
- "Don't care about..." → Delete slot
</rule>
<rule>Handle data types:
- Categorical: Match to schema values (case-insensitive)
- Free-text (hotel-name): Preserve exact spelling
- Numerical: Convert word numbers to digits ("two" → "2")
</rule>
<rule>Output MUST be valid JSON with:
- Keys: Only slots with active values
- Values: Arrays containing latest valid entries
</rule>
</processing-rules>

To process:
<dialogue-context>
Previous State: {icl_history_state}
Current Utterance: {icl_current_dialogue}
</dialogue-context>

Assistant:
```

Figure 6: Prompt for IterChat during few-shot prompting

```
# Prompt for multi-turn dialogue:
Human:
<purpose>
You are a state extraction agent that analyzes FULL DIALOGUE HISTORY to determine the FINAL hotel preferences,
capturing all valid slot values through conversational evolution.
</purpose>

<schema format="key:[allowed_values]">
{
  "hotel-pricerange": ["expensive", "cheap", "moderate"],
  "hotel-type": ["guesthouse", "hotel"],
  "hotel-parking": ["free", "no", "yes"],
  "hotel-bookday": ["monday",...,"sunday"],
  "hotel-bookpeople": ["1"..."8"],
  "hotel-bookstay": ["1"..."8"],
  "hotel-stars": ["0"..."5"],
  "hotel-internet": ["free", "no", "yes"],
  "hotel-name": "free-text",
  "hotel-area": ["centre", "east", "north", "south", "west"]
}
</schema>

<extraction-protocol>
1. Temporal Analysis:
   - Scan dialogue chronologically
   - Track value changes across turns
   - Preserve only the final valid state

2. Conflict Resolution:
   - Last-mentioned value overrides previous ones
   - Explicit revocation ("not X anymore") deletes slot
   - Implicit changes ("actually...") replace earlier values

3. Context Binding:
   - Bind numeric references to nearest hotel context
   - Ignore preferences mentioned in other domains (restaurant/taxi)
</extraction-protocol>

<normalization-rules>
• Convert word forms: {"two days" → "2", "west side" → "west"}
• Map synonyms: {"mid-priced" → "moderate", "B&B" → "guesthouse"}
• Filter tentative phrases: {"maybe 4 stars" → "4"}
</normalization-rules>

To process:
<full-dialogue>
{icl_dialogue}
</full-dialogue>

Assistant:
```

Figure 7: Prompt for multi-turn dialogue during few-shot prompting