FPT Education

FPT UNIVERSITY

GradBot - A unified dialogue state tracking and dialogue response model for Task Oriented Dialogues (TOD) and Open Domain Dialogues (ODD)

by

Sinh Nguyen, Truc Nguyen

THE FPT UNIVERSITY HO CHI MINH CITY

GradBot - A unified dialogue state tracking and dialogue response model for Task Oriented Dialogues (TOD) and Open Domain Dialogues (ODD)

by

Sinh Nguyen, Truc Nguyen

Supervisor: Dr. Nguyen Quoc Trung, Dr. Truong Hoang Vinh

A final year capstone project submitted in partial fulfillment of the requirement for the Degree of Bachelor of Artificial Intelligence in Computer Science

DEPARTMENT OF ITS

THE FPT UNIVERSITY HO CHI MINH CITY

April 2024 **(Month year)**

ACKNOWLEDGMENTS

The project we are working on is under the ownership of Gradients Technologies. During our internship with the company, we were granted permission by the Director to utilize this project for our academic requirements, specifically for our graduation project. The conceptualization and execution of the project plan were primarily our contributions. Prior to initiating the research, we were equipped with fundamental knowledge by our mentors and project managers. They also provided us with substantial support in refining our methodologies to optimize results. Furthermore, they assisted us in reviewing our code after each task. For the training of models across all modules in this system, we utilized the company's GPU resources.

Upon the conclusion of our internship, we, along with our instructors, were given the opportunity to continue our research and further development on the project. This professional experience has been instrumental in our academic and career growth.

AUTHOR CONTRIBUTIONS

Conceptualization, Sinh Nguyen and Truc Nguyen and Gradients Technologies.; methodology, Sinh Nguyen and Truc Nguyen and Gradients Technologies; software, Sinh Nguyen and Truc Nguyen; validation, Sinh Nguyen and Gradients Technologies; formal analysis, Sinh Nguyen and Truc Nguyen; investigation, Gradients Technologies; resources, Gradients Technologies; data curation, Sinh Nguyen and Truc Nguyen; writing—original draft preparation, Sinh Nguyen and Truc Nguyen; writing—review and editing, Sinh Nguyen and Truc Nguyen; visualization, Sinh Nguyen and Truc Nguyen; supervision, Dr.Nguyen Quoc Trung and Dr. Truong Hoang Vinh; project administration, Gradients Technologies; funding acquisition, Sinh Nguyen and Truc Nguyen. All authors have read and agreed to the Final Capstone Project document.

ABSTRACT

The system for task-oriented dialogue domain requires classifying user intent and replying to a specific goal domain. Within the task-oriented sub-module, the Dialogue State Tracker (DST) is well-known as a variety processing tracker. Nonetheless, current DST models tend to focus solely on task-oriented domains (ToD), resulting in constrained performance when deployed in varied scenarios. Besides, current dialogue response models of previous studies achieved quite poor results because the responses were not natural and fluent. In this paper, we propose GradBot, a unified system including DST model vs response model that predicts both types of tasks, task-oriented dialogue (TOD) and open domain dialogue (ODD). Our model leverages the recent advances in prompt engineering and conditional generation to perform zero-shot learning. After experiments, GradBot has achieved an 88.6% and 82.5% score on Joint Goal Accuracy metrics when evaluating the Scheme-Guided Dialogue (SGD) and FusedChat test sets correspondingly, demonstrating the adaptation ability for multi-domains.

Keywords: Task Oriented Dialogue, Open Domain Dialogue, Dialogue State Tracking, Conversational AI

CONTENTS

List of Figures

List of Tables

1. INTRODUCTION

The task-oriented domain has attracted a lot of attention not only in academics but also in industry. This objective is to achieve specific strategies, such as providing information or performing an action that satisfies the user's request.

Specifically, the task-oriented system will replace most product consultants, reservation staff and customer service staff. This system can interact with users and allow them to carry out intentions such as: buying products, searching or booking hotel rooms, restaurant tables, making medical appointments, buying music tickets, buying tickets flight, train, taxi, etc. and actions include: providing characteristics of the hotel or restaurant that the user wants to search or book, request the system about the place (does it has internet? How much does it cost ? etc), accept orders, confirm orders, etc. After receiving actions and intentions from the user, the system will respond to the user with system actions such as: providing information about hotels and restaurants that the user requests, offering hotel, restaurant in the user's destination, performing order, etc.

One of the crucial components of the task-oriented domain is Dialogue State Tracking (DST), which tries to predict appropriate actions to resolve the goals. At every turn, DST has to look up the dialogue history (whole or sliding window) to the current user query to determine user intentions, actions with specific values in the slot list [1, 2]. In our observation, there are two kinds of DST designed:

- The traditional method uses an Encoder module exploiting multihead layers to build classified data intent prediction, slots prediction, and slot filling [1, 3];
- Seq2Seq module uses prompting to show semantics between turns and ontology through conversation to predict a required value [4–6].

In industrial applications, DST is required to adapt flexibly new domains (services) without prior training for a specific task. For this purpose, the role of zeroshot prediction on unseen domains becomes important in DST. Some previous work [4–6] uses guided schema as a description to show the semantics of slots with input sentences (user query). With recent advances in pre-trained language models [7–9], augmented language techniques are gaining more and more attention. These methods have demonstrated impressive improvement and zero-shot adaptability [3, 10, 11]. Moreover, the in-context learning framework (ICL) shows efficient methods and techniques in DST without the re-training stage by combining prompting and examples for a task (few-shot) [12, 13].

Often other studies only focus on the ability to predict user actions and intentions using the DST model, without researching the ability to respond to users appropriately and fluently. Therefore, our system incorporates a Dialogue Response model designed to interact with users in a way that depends on the predictions made by the DST model (including the user's actions and intentions). Historically, dialogue response models were primarily developed for conversational AI systems, with a particular focus on question answering systems. The input model mainly consists of historical context and the current user query. In cases where the system is required to respond to untrained queries, documents retrieved from the internet or a database are added to the input model to provide knowledge to help the model answer user queries. However, to satisfy the user's task-oriented requests, the Dialogue Response model needs to include a main component, called 'system action'. This component represents the action that the system will take in response to the user, after receiving the user's actions and intentions.

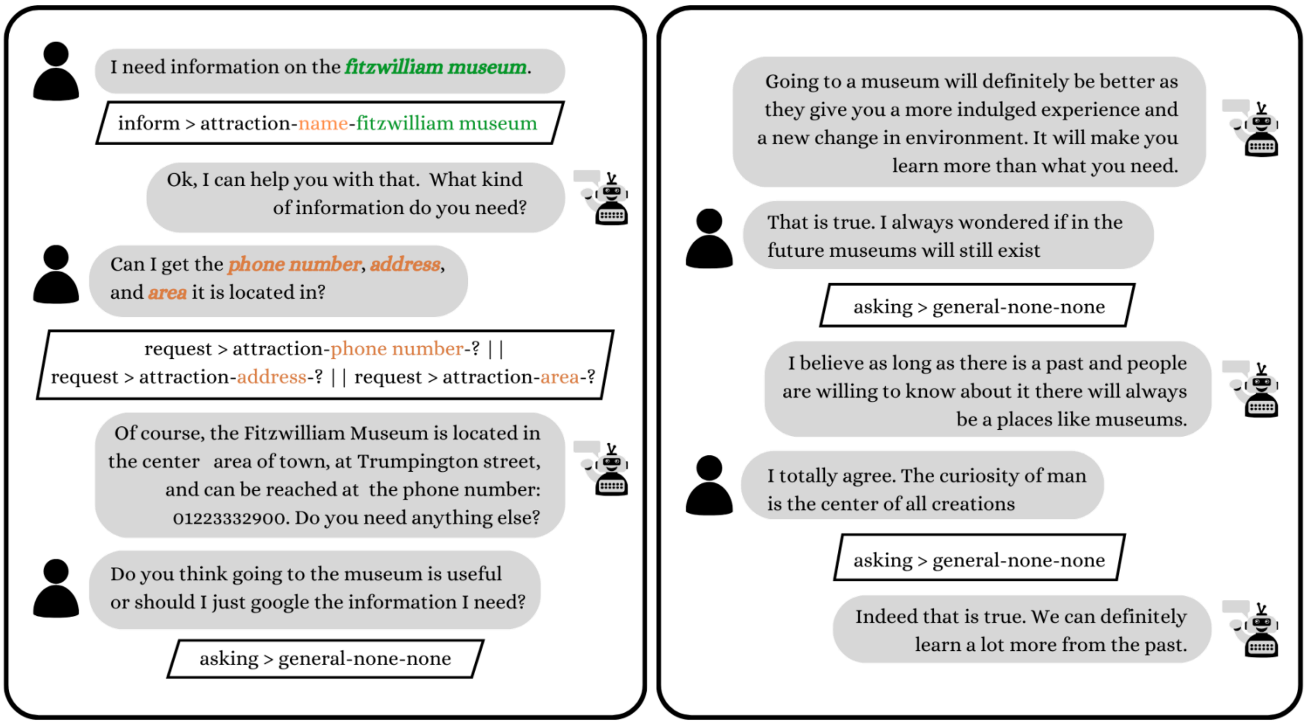

I need information on the fitzwilliam museum.
inform > attraction-name-fitzwilliam museum
Ok, I can help you with that. What kind of information do you need?
Can I get the phone number, address, and area it is located in?
request > attraction-phone number-? | | request > attraction-address-? | | request > attraction-area-?
Of course, the Fitzwilliam Museum is located in the center area of town, at Trumpington street, and can be reached at the phone number: 01223332900. Do you need anything else?
Do you think going to the museum is useful or should I just google the information I need?
asking > general-none-none
Going to a museum will definitely be better as they give you a more indulged experience and a new change in environment. It will make you learn more than what you need.
That is true. I always wondered if in the future museums will still exist
asking > general-none-none
I believe as long as there is a past and people are willing to know about it there will always be a places like museums.
I totally agree. The curiosity of man is the center of all creations
asking > general-none-none
Indeed that is true. We can definitely learn a lot more from the past.

Fig 1. *An example of Attraction Domain on Fusedchat datasets. The conversation builds from MutiWoz2.4 by rewriting the existing Task-oriented domain turns and adding new Open Dialogue domain turns.*

More specifically, Figure 1 shows an example conversation with the associated dialogue state of the attraction domain. The user wants to find information about a specified name and request more data about the phone number, address, and area. At the same time, they ask if going to the museum is useful or not (general domain)? The system must answer questions based on their knowledge (open question-answering domain) or even daily conversation (chitchat). It requires an intelligent chatbot with novel architectures and approaches such as DST, switching domain classification (attraction domain to general domain), and supporting generative AI-specific knowledge to improve the performance of the conversational system. However, there still has been a noticeable gap until now between existing benchmark datasets and real-life human conversations. These datasets cover a limited number of domains, unrealistic constraints focus on a few skill sets and do not have empathy or personality consistency, etc.

Motivated by this research, we propose a recent advance in prompt engineering and conditional generation to adapt zero-shot learning applications useful in the business domain. The effectiveness of our proposed method is experimented on the FusedChat, SGD, and MultiWoz2.4 datasets, achieving remarkable performance on some benchmarks and human evaluations. Our proposed methods can be summarized as follows:

- We introduce a simple method but effective controls tracking conversation flow and easily expand the new business domains (services). Our evidence shows that using abbreviations such as tags we often see such as: <EOS>, <CLS>, <CTX>, <P>, etc., is used to let the model know what they need to do or where to get information from ? This is not as effective as natural language descriptions with detailed instructions like: "Use the information provided from >context:... and current query:... to answer user questions", this approach better supports both the Task-Oriented Domain and the Open Dialogue Domain.
- While several researchers/developers focus on using the Large Language Model to give a strong performance experience. We are interested in improving the small language model, which achieves results equal to or superior to other large language models, on a variety of tasks in the Dialogue State Tracking model and Dialogue Response model based on a contextual semantics ontology. We are proud that our system, with its much smaller size, not only achieves results equal to or better than Large Language Models on seen domains in test set, which were trained in train set, but also achieved very high results on unseen domains in the test set, outperforming previous studies in terms of zero shot ability.
- Our full system can be applied to developing practical applications to serve businesses that need chatbots that can interact with users and allow them to make reservations, make purchases or search product's information without wasting time retraining to suit that business's domain. In addition, the system is small in size, so it is easy to set up on most businesses' platforms and the response speed will be much faster than systems trained on Large Language Models.

The remainder of the article is structured as follows. Section 2 discusses our relevant works. The key idea for the Dialogue State Tracker combines prompting and conditional generation with ontology performing the details is explained in Section 4. The outcome and the work's conclusion are then reported in Sections 7 to 8, respectively.

2. RELATED WORK

2.1 Dialogue State Tracking

The construction of a conversational Task-oriented system forms the crux of this discussion. The methodologies employed in this process can be broadly categorized into two distinct groups: Classification (Encoder) and Generation (Seq2Seq). In recent times, transformer-based pre-trained models, such as BERT [14], have made significant strides in various natural language processing tasks, demonstrating remarkable results. This success has led to the proposal of a multi-task BERT-based model [15]. This model is designed to tackle challenges such as intent prediction, slot filling, and request slot filling by encoding the history and service schema.

However, these approaches present certain limitations. They are not applicable to unseen values and struggle to scale up to large domains. To mitigate these issues, a UniDU framework is introduced in [16]. This framework facilitates effective information exchange across a diverse range of dialogue-understanding tasks. The study conducted found an intuitive multitask mixture training method. This method enables the unified model to bias convergence towards more complex tasks. This discovery is a significant step forward in the field, offering promising prospects for the development of more sophisticated and efficient task-oriented systems.

2.2 Enhance Reading Comprehension

In contrast to the research methodologies discussed earlier, several scholars have discovered that generative extractive methods, specifically Machine Reading for Question Answering (MRQA), are highly effective in addressing textual Question Answering (QA) tasks. This effectiveness stems from the MRQA’s inherent ability to comprehend the context. Leveraging this advantage, CoFunDST [17] amalgamates Dialogue State Tracking with Machine Reading Comprehension. This combination is applied to context-choice fusion, serving as an extensive knowledge base for predicting slots and values among available candidates. This approach significantly enhances zero-shot performance.

In another experiment focusing on comprehension tasks, TransferQA [18] introduces two effective methods: constructing negative question sampling and context truncation. These methods are particularly adept at handling "none" value slots and enhancing the model's generalization ability in unseen domains. Simultaneously, Moradshahi's approach [19] emphasizes that the collection of large amounts of data for every dialogue domain is often both costly and inefficient. To address this issue, his study applies the transfer learning technique. This technique utilizes a limited task-oriented subset in the source data language to construct a high-quality model for other target languages. The experiments conducted using this approach yielded unexpected results. Training with only 10% of the data points led to a 10% increase in performance compared to the previous state-of-the-art (SOTA) research on both zero-shot and few-shot learning.

In the realm of multilingual applications, PRESTO [20], a public multilingual conversation dataset for real-world Natural Language Understanding (NLU) tasks, and the application-based mT5 model are considered as the baseline training in this field. The experiments conducted using this module demonstrated its effectiveness in handling various linguistic phenomena. This underscores the potential of these methodologies in enhancing reading comprehension in task-oriented systems.

2.3 Dialogue Response

The dialogue response model constitutes a pivotal component in a task-oriented dialogue system. It plays a decisive role in determining the system's capacity to communicate effectively with the user. The field has witnessed significant advancements recently, primarily driven by the application of deep learning techniques. Among these, transformer-based models such as GPT-3 [21] have demonstrated exceptional performance in generating responses that closely resemble human interaction. In the context of multi-domain dialogues, maintaining context and coherence across diverse topics presents a substantial challenge. To tackle this, several researchers have proposed the employment of context-aware models. These models are capable of tracking the dialogue history and leveraging this information to generate responses that are not only more relevant but also exhibit greater coherence.

Despite these advancements, the field of dialogue response generation for task oriented dialogue systems continues to grapple with numerous challenges. These include the need for more effective strategies to handle out-of-domain queries and enhancing the system's ability to comprehend and generate responses in multiple languages.

Future research in this field is anticipated to concentrate on addressing these challenges. The ultimate objective is to further enhance the performance and usability of task-oriented dialogue systems, thereby making them more efficient and user friendly. This ongoing research and development in the field holds great promise for the future of task-oriented dialogue systems.

2.4 Affection dataset

Recent advancements in the realm of state-of-the-art research [22–24] have significantly improved existing Task-Oriented Dialogue (TOD) datasets. This has been achieved by designing a variety of methods aimed at enhancing context, samples, and method processing to facilitate real human-level conversation.

FusedChat [23], for instance, has restructured Task-oriented dialogue and incorporated new open domain dialogue (commonly referred to as chitchat) to create a novel dialogue. This innovative approach has broadened the scope of dialogue systems, making them more versatile and user-friendly.

In a similar vein, ACCENTOR [22] has proposed a data augmentation method specifically designed for generating conversation. This method leverages pre-trained generative models and employs a custom filter to minimize the effort required for human annotation. This approach not only streamlines the process but also enhances the quality of the generated dialogues.

Building on these approaches, we have conducted an in-depth analysis of our model training on FusedChat and SGD datasets. This analysis involved evaluating single and multi-domain dialogue, providing valuable insights into the performance and adaptability of our model. This comprehensive approach allows both TaskOriented Dialogue (TOD) and Open Domain Dialogue (ODD) to adapt seamlessly to the business domain. This adaptability is crucial in ensuring that our dialogue systems can effectively cater to a wide range of business needs and requirements. As such, our research contributes significantly to the ongoing efforts to enhance the performance and usability of task-oriented dialogue systems.

3. PROJECT MANAGEMENT PLAN

3.1 Overall Project Objective

Our primary objective is to develop two sophisticated models: the Dialogue State Tracking model and the Response model. These models are designed with the ambition to outperform all existing models in terms of their metrics, while maintaining a parameter count that is either equivalent to or less than that of their counterparts. This approach ensures an optimal balance between performance and computational efficiency.

Furthermore, we are committed to transforming these models into a practical, market-ready system. Our vision is to offer this system to businesses as a solution that enables their customers to interact and place orders seamlessly. The unique selling point of our system is its ability to facilitate customer-business interactions without the need for consultants or customer service staff. This feature not only enhances the user experience but also contributes to operational efficiency for businesses.

Lastly, one of our key goals is to design a model that exhibits robust performance across unseen domains without the necessity for fine-tuning. This characteristic is crucial for commercialization as it allows us to deploy the system across various sectors without the need for extensive customization. This, in turn, saves time and resources, making the system a cost-effective solution for businesses.

3.2 Effort Distribution

This is a table of the effort distribution of our team members.

Task name	Priority	Owner	Start date	End date	Status	Issues
Find documents	High	Sinh, Truc	1/1	8/1	Done	Nothing
Review papers	High	Sinh, Truc	1/1	8/1	Done	Nothing
Review and analyze public dataset	Medium	Sinh	9/1	11/1	Done	Nothing
Collect data	High	Truc	12/1	11/4	Done	Nothing
Experiment	High	Sinh	15/1	2/2	Done	Nothing
Code Implement	High	Sinh, Truc	3/2	1/3	Done	Nothing
Test and compare results	High	Sinh	2/3	15/3	Done	Nothing
Writing paper	High	Sinh, Truc	10/3	30/3	Done	Nothing
Writing report	Medium	Sinh, Truc	10/3	9/4	Done	Nothing
Code demo	High	Sinh, Truc	5/4	20/4	Done	Re-edit
Future work	Low	Sinh, Truc	20/4	30/6	Defined	Nothing

Table 1. *Project plan. Above are our main tasks assignments.*

Items	Link	Description
SGD	Towards Scalable Multi-Domain Conversational Agents: The Schema-Guided Dialogue Dataset (arxiv.org)	Training
Fusedchat	2109.04137.pdf (arxiv.org)	Training
Multiwoz 2.4	MultiWOZ 2.4: A Multi-Domain Task-Oriented Dialogue Dataset with Essential Annotation Corrections to Improve State Tracking Evaluation (aclanthology.org)	Evaluation

Table 2. *Source data*

4. METHODOLOGY

4.1 Set up the system's architecture

Our system consists of 3 modules, including 2 main modules that we mentioned above: DST model, Dialogue Response model, and intermediate module to handle conversion from user actions and intentions into system actions.

For the DST model, which we refer to as GradDST, we propose a methodology that incorporates template input model for training, which utilizes a set of instructions, context, current user query, ontology, and list of user actions. Subsequently, the model is tasked with choosing the most logical representation to learn through the process of reading and extracting information from the input model. After that, GradDST will have to perform 3 parallel tasks: 1. Determine whether the current user query is TOD or ODD, 2. Determine the user's actions and intentions in the current user query, 3. Determine the information that the user has provided or updated (user state) throughout the conversation. For instance, if user input: "I want to book a 5-star hotel in District 1, how much does it cost per night?", GradDST's output will be "(type) TOD (action) inform>hotel-star-5 || inform>hotel-destination-District 1 || request> hotel-none-none || inform intent>hotel-intent-ReserveHotel (state) hotel-star-5 || hotel -destination-District 1".

As for the Dialogue Response model, referred to as GradRES, we propose a methodology that incorporates a template-based input model for training. This model utilizes a set of instructions, context (including the current user query), ontology, and system actions. The primary task of GradRES is straightforward - it is required to generate a text response primarily based on system actions. For instance, if the system actions are HOTELS: [offer(name=CayXanh) and offer(star=4) and offer intent(ReserveHotel)], then GradRES will generate the response: "We recommend CayXanh, a 4-star hotel. Would you like to make a reservation?".

We have configured system actions in the template-based input model as opposed to user actions and intentions. This raises the question of the origin of these system actions. Consider a scenario where user actions and intentions from GradDST are substituted in place of system actions. In such a case, our GradRES would be tasked with executing two tasks: 1. Predicting system actions based on user actions and intentions, 2. Generating a system response from the system actions predicted in task 1. We believe that this would impose an undue burden on the learning process of GradRES and result in suboptimal outcomes for both research metrics and product development.

Therefore, we introduce an intermediate module between GradDST and GradRES which we refer to as GradACT, and is tasked with converting user actions and intentions into system actions. We do not use a language model for GradACT, we will do it by using basic functional programming. Because GradACT needs to interact with the database to get information of objects during the search process, request information, or modify the number of remaining rooms, etc. And the most important reason is that converting user actions and intentions to system actions by basic functional programming will be 100% correct.

4.2 Training method

In Task-Oriented Dialogue (TOD) systems, two prevalent training methodologies are employed: End-to-End (E2E) and Modular. The E2E method is often favored in studies conducted by large research laboratories due to its inherent advantages. These include consistency, synchronization, and smoothness between modules. This approach accumulates all losses into a single comprehensive loss, allowing for the simultaneous updating of weights across all modules. However, the Modular method is also respected due to its ability to address the limitations of the E2E method. To understand the effectiveness of these methods, we can compare them based on the following evaluation criteria.

4.2.1 Consistency and Compatibility Between Modules

As previously mentioned, the implementation of an end-to-end training methodology can significantly enhance the consistency of the modules throughout the entire process. This approach ensures that all errors are accumulated and calculated at the final stage, leading to the optimization of the entire system. The close association fostered by this method enhances the relevance of the responses generated by the GradRES module. This is achieved by aligning the responses more closely with the user's query, as well as the user's actions and intentions as interpreted by the GradDST module.

Furthermore, the GradACT module is also optimized under this methodology. It is designed to provide more appropriate system actions in response to the user's actions and intentions. This optimization process ensures that the system's responses are not only accurate but also contextually appropriate, thereby enhancing the overall user experience.

However, it is important to note a key limitation associated with the modular method. The primary issue lies in the lack of connection between the individual modules. Given that these modules are trained independently, it is possible for each module to perform well in isolation. However, when these modules are combined, the overall performance may not meet expectations.

This is primarily due to the fact that the independent training of each module does not account for the interdependencies and interactions that occur when the modules are integrated into a single system. As a result, despite the individual effectiveness of each module, the overall system performance may be suboptimal due to the lack of coordination and coherence between the modules.

4.2.2 Flexibility to Change and Upgrade the System

This evaluation criterion elucidates the distinct advantage of the modular method over the end-to-end method. In the context of training, the end-to-end method amalgamates all modules into a single entity. Consequently, the model will generate and store only 1 checkpoint file for all three modules.

This implies that if any issues arise during the training process, or if there is a need for modifications in subsequent versions, a significant amount of time will be required to retrain the system from scratch. For instance, if modules 1 and 2 are functioning optimally and a system issue is identified in module 3, the necessary revisions will be made to module 3. After that, this will necessitate a comprehensive retraining of the system, including all three modules, thereby consuming a considerable amount of time.

Similarly, if we were to discover an improved solution that yields superior results for module 1, the entire system, encompassing all three modules, would need to be retrained from the beginning. This process can be time-consuming and may not be the most efficient approach.

On the other hand, the modular method offers a more optimal solution. The key advantage of this method lies in the independent checkpoint file of each module. If any module encounters issues or requires upgrades, only the affected module needs to be retrained. This significantly reduces the time required for retraining, thereby enhancing the efficiency of the process.

In conclusion, while the end-to-end method has its merits, the modular method’s ability to independently train and upgrade each module presents a more efficient and time-saving approach. This highlights the superior advantages of the modular method in comparison to the end-to-end method.

4.2.3 Optimal training time

As previously highlighted, the modular method exhibits a significant advantage over the end-to-end method in terms of training time, particularly when there is a need to replace or upgrade the system. But what about the scenario where all three modules are functioning without any issues? Even in this case, the modular method proves to be more time-efficient.

Training three modules using the end-to-end method necessitates a large and powerful GPU that can accommodate and process all three modules simultaneously. However, this is a challenging requirement as not everyone has access to such high-quality GPUs. Therefore, the end-to-end method is typically more suitable for smaller projects, which consist of smaller modules that can operate on standard GPUs or large research laboratories, which work on projects comprising larger modules, are usually the ones that can afford to invest in these expensive hardware devices.

But what if we want to undertake projects with large modules similar to those in large research labs, but we only have standard laptops or GPUs available from platforms like Colab or Kaggle, and we aim to train quickly? Is it possible to outperform in terms of time ? The answer lies in adopting the modular approach. Instead of training all three modules on a single large GPU, we can train the three modules in parallel on three different smaller GPUs. This strategy enables us to achieve results faster than research groups using the end-to-end method, even though our equipment may not be as advanced as theirs.

In conclusion, the modular method, with its ability to independently train each module, offers a more efficient and time-saving approach, making it a superior choice over the end-to-end method, especially when resources are limited.

4.2.4 Suitability for Research or Production Needs

In this evaluation, we will set aside the previously discussed three criteria and operate under the assumption that we have an abundance of financial resources, hardware capabilities, and time. The question then arises: which method is more suitable for academic research and writing paper, and which is more appropriate for practical product application?

For research purposes, the end-to-end method is arguably more suitable. This is primarily because it tends to yield higher results compared to the modular method. Research groups often opt for the end-to-end approach as they strive to achieve the highest possible results and maintain a high standing in metric rankings. While this may entail higher costs and a longer time commitment, the priority for these groups is to achieve high results on the test set.

However, when it comes to the application of the Task-Oriented Dialogue (TOD) system as a product, the end-to-end method may not be as effective. The system will need to interact with various business domains, including unseen domains. Each business has its own unique policies and regulations, necessitating adjustments to the intermediate module to accommodate each specific business.

The end-to-end method, due to its integrated nature, cannot be disassembled. As a result, while it may perform well on test sets, it lacks the flexibility needed to adapt to the specific needs of each business. Therefore, despite its advantages in a research setting, the end-to-end method may not be the most effective approach for practical product application.

In conclusion, while both the end-to-end and modular methods have their respective strengths, their suitability varies depending on the context. The end-to-end method may be more appropriate for research purposes, while the modular method offers greater flexibility and adaptability for practical product applications.

4.3 Optimization in training

4.3.1 Model parallelism

Model parallelism constitutes a distributed training approach wherein the deep learning model undergoes partitioning across multiple devices, whether within or across instances. This overview delves into model parallelism, highlighting its utility in addressing challenges inherent in training DL models, which often boast considerable size. Additionally, it outlines offerings within the SageMaker model parallel library aimed at facilitating the management of model parallel strategies and memory consumption.

4.3.1.1 Understanding Model Parallelism

The efficacy of deep learning models in tasks like computer vision and natural language processing escalates with their increasing size, marked by expansions in layers and parameters. Nevertheless, the capacity of a single GPU's memory imposes a cap on the maximum model size feasible for training. The limitations of GPU memory pose bottlenecks during DL model training:

- They confine the model size that can be trained since the memory footprint of a model scales proportionately to the parameter count.
- They restrict the per-GPU batch size during training, thereby diminishing GPU utilization and training efficiency.

To surmount these constraints associated with single-GPU training, SageMaker offers the model parallel library. This resource aids in distributing and training DL models efficiently across multiple compute nodes. Moreover, leveraging this library enables the attainment of optimized distributed training utilizing EFA-supported devices. These devices bolster inter-node communication performance with attributes like low latency, high throughput, and OS bypass.

4.3.1.2 Assessing Memory Requirements Prior to Implementation

Before deploying the SageMaker model parallel library, it is prudent to gauge the memory prerequisites for training large DL models. Consider the following aspects:

For a training job employing AMP (FP16) and Adam optimizers, the GPU memory required per parameter amounts to approximately 20 bytes. This breakdown comprises:

- An FP16 parameter (2 bytes)
- An FP16 gradient (2 bytes)
- An FP32 optimizer state (8 bytes, based on Adam optimizers)
- An FP32 copy of the parameter (4 bytes, necessary for the optimizer apply operation)
- An FP32 copy of the gradient (4 bytes, necessary for the optimizer apply operation)

Even for relatively modest DL models featuring 10 billion parameters, the memory demand can surpass 200GB. This exceeds the typical GPU memory capacity, such as that of the NVIDIA A100 with 40GB/80GB or V100 with 16/32 GB available on a single GPU. Notably, besides the memory requirements for model and optimizer states, other factors like activations generated during the forward pass contribute to memory consumption, amplifying the overall demand. For distributed training endeavors, employing Amazon EC2 P3 and P4 instances equipped with NVIDIA V100 and A100 Tensor Core GPUs, respectively, is recommended. For detailed specifications encompassing CPU cores, RAM, attached storage volume, and network bandwidth, consult the Accelerated Computing section of the Amazon EC2 Instance Types page. Even with the utilization of accelerated computing instances, it becomes apparent that models with approximately 10 billion parameters, such as Megatron-LM and T5, and even larger models with hundreds of billions of parameters like GPT-3, cannot accommodate model replicas on individual GPU devices.

4.3.2 DistributedDataParallel

DistributedDataParallel (DDP) is a parallel computing technique used in deep learning to train models across multiple devices or nodes. It is a part of the PyTorch library and is designed to scale the training process, allowing for faster training times with larger models and datasets.In DDP, the model is replicated on every device, and each replica handles a subset of the input data. The replicas operate independently in the forward pass, computing their own outputs and gradients. In the backward pass, gradients from each replica are combined across all devices using an operation called all-reduce.The primary advantage of DDP is its scalability. By distributing the computation across multiple devices, DDP allows for training larger models and processing larger datasets than would be possible on a single device. This makes it a key tool in the training of large-scale deep learning models.However, DDP also has its challenges. One of the main challenges is the need to synchronize the model parameters across all devices after each update, which can be communication-intensive. Additionally, because each device computes its own gradients independently, there can be discrepancies between the gradients computed by different devices, leading to potential issues with model convergence.Despite these challenges, DDP remains a powerful tool for distributed deep learning, enabling researchers and practitioners to train larger and more complex models than ever before. It is continually being improved and optimized, with ongoing research aimed at addressing its limitations and expanding its capabilities.

4.3.2.1 FSDP

During DistributedDataParallel (DDP) training, each process or worker possesses a copy of the model and handles a batch of data independently. Subsequently, allreduce is employed to aggregate gradients across various workers. In DDP, both the model parameters and optimizer states are duplicated across all workers. Fractional Sharded Data Parallelism (FSDP) represents a form of data parallelism wherein model parameters, optimizer states, and gradients are partitioned across DDP ranks.

When utilizing FSDP for training, the GPU memory usage is reduced compared to DDP across all workers. This reduction enables the training of notably large models, facilitating the accommodation of larger models or batch sizes on the device. However, this advantage is counterbalanced by increased communication volume, albeit mitigated by internal optimizations like concurrent communication and computation, which alleviate communication overhead.

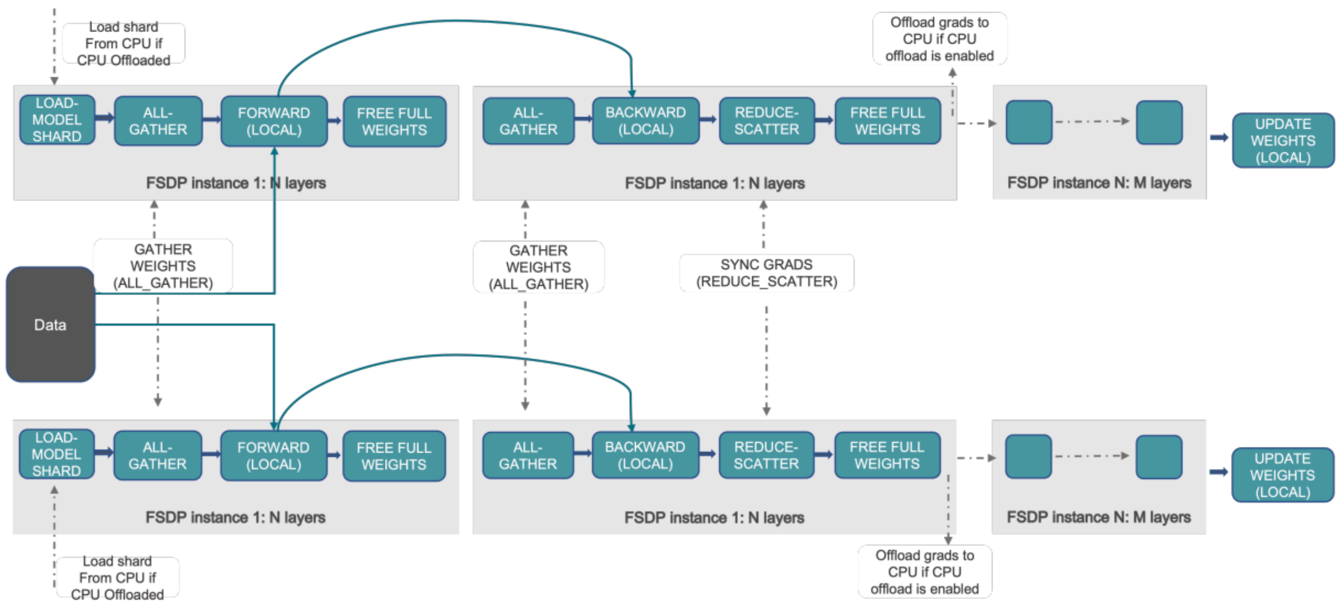

Load shard From CPU if CPU Offloaded
LOAD-MODEL SHARD
ALL-GATHER
FORWARD (LOCAL)
FREE FULL WEIGHTS
FSDP instance 1: N layers
ALL-GATHER
BACKWARD (LOCAL)
REDUCE-SCATTER
FREE FULL WEIGHTS
FSDP instance 1: N layers
Offload grads to CPU if CPU offload is enabled
FSDP instance N: M layers
UPDATE WEIGHTS (LOCAL)
GATHER WEIGHTS (ALL_GATHER)
Data
GATHER WEIGHTS (ALL_GATHER)
SYNC GRADS (REDUCE_SCATTER)
LOAD-MODEL SHARD
ALL-GATHER
FORWARD (LOCAL)
FREE FULL WEIGHTS
FSDP instance 1: N layers
ALL-GATHER
BACKWARD (LOCAL)
REDUCE-SCATTER
FREE FULL WEIGHTS
FSDP instance 1: N layers
FSDP instance N: M layers
UPDATE WEIGHTS (LOCAL)
Load shard From CPU if CPU Offloaded
Offload grads to CPU if CPU offload is enabled

Fig 2. *Pytorch Fully Sharded Data Parallel (FSDP).*

At a high level FSDP works as follow:

In constructor:

- Shard model parameters and each rank only keeps its own shard

In forward path:

- Run all gather to collect all shards from all ranks to recover the full parameter in this FSDP unit
- Run forward computation
- Discard parameter shards it has just collected

In backward path:

- Run all gather to collect all shards from all ranks to recover the full parameter in this FSDP unit
- Run backward computation
- Run reduce scatter to sync gradients
- Discard parameters.

A perspective on FSDP's sharding involves breaking down the DDP gradient all-reduce process into two distinct steps: reduce-scatter and all-gather. In this approach, during the backward pass, FSDP condenses and distributes gradients, guaranteeing that each rank retains a portion of the gradients. Following this, it adjusts the respective segment of parameters during the optimizer step. Subsequently, in the subsequent forward pass, it executes an all-gather operation to assemble and merge the updated parameter segments.

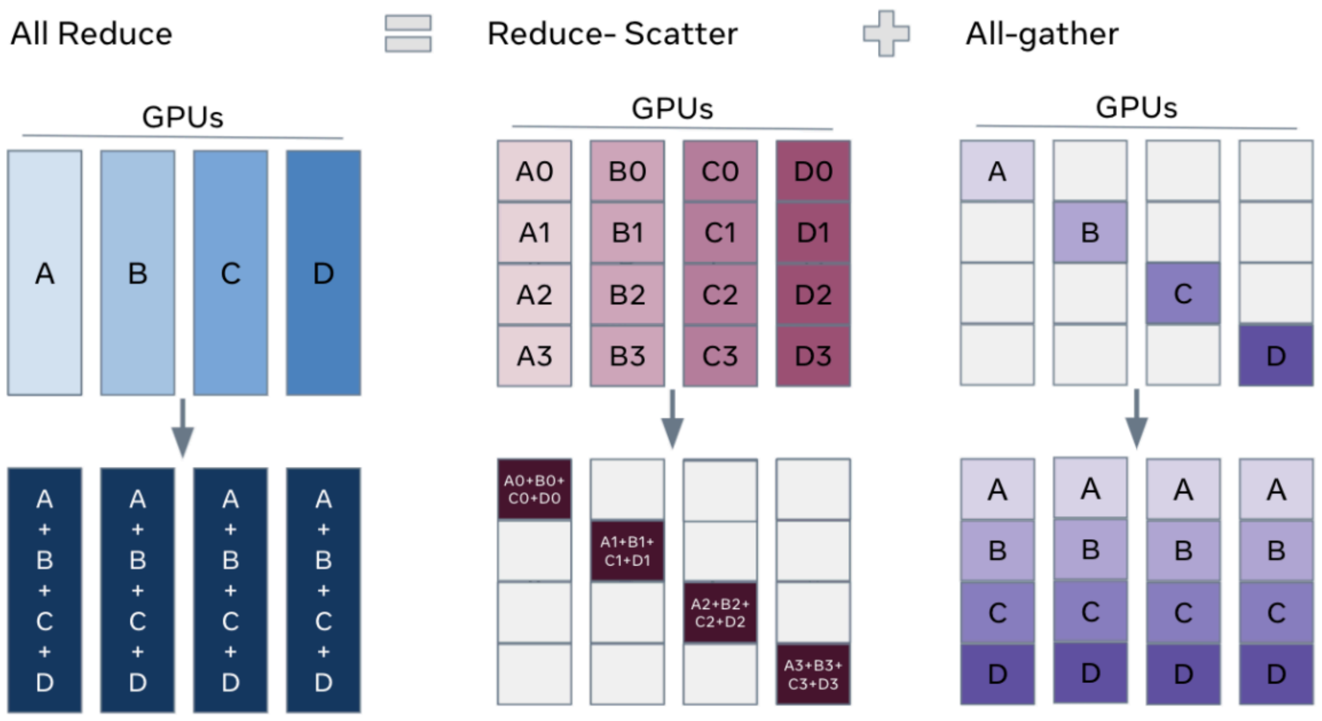


Fig 3. *Decomposing All-Reduce Operations in Distributed Data Parallel Training: A Path to Full Parameter Sharding.*

4.3.2.2 DeepSpeed

DeepSpeed stands as a PyTorch optimization library engineered to streamline distributed training, rendering it both memory-efficient and swift. Central to its functionality lies the Zero Redundancy Optimizer (ZeRO), which facilitates the training of expansive models at scale. ZeRO operates through several key stages: ZeRO-1: Divides optimizer state across GPUs. ZeRO-2: Partition gradients across GPUs. ZeRO-3: Distributes parameters across GPUs. Moreover, in environments constrained by GPU resources, ZeRO empowers the offloading of optimizer memory and computation from the GPU to the CPU, thereby enabling the training of exceedingly large models on a single GPU. DeepSpeed GradBot 17 seamlessly integrates with the Transformers Trainer class for all ZeRO stages and offloading functionalities. Users need only provide a configuration file or utilize a provided template. For inference tasks, Transformers support ZeRO-3 and offloading, facilitating the loading of substantial models. This guide elucidates the deployment of DeepSpeed training, encompassing the activation of various features, configuration file setup for distinct ZeRO stages, offloading, inference procedures, and leveraging DeepSpeed without the Trainer interface.

4.3.3 Combine Model Parallel and Data Parallel

Model parallelism and data parallelism are two distinct strategies employed in the field of machine learning to optimize computational efficiency.

Model Parallelism involves the partitioning of a model into equal segments, each of which is allocated to a separate GPU. The number of GPUs utilized is equivalent to the number of model segments post-partitioning. This approach mitigates the need for a singular, high-cost GPU to house the entire model, instead leveraging multiple, more cost-effective GPUs. On the other hand, Data Parallelism entails replicating the entire model across multiple GPUs. This strategy facilitates accelerated data training, with the speed of training proportional to the number of GPUs employed. However, this method necessitates that each GPU possesses the capacity to accommodate the full model, thereby requiring the use of high-end, expensive GPUs.

In our approach, we incorporate the principles of both model parallelism and data parallelism. As illustrated in the Figure 4, the model is segmented into four equal parts and distributed across four GPUs (labeled 0, 1, 2, 3). Each data batch is initially processed in device 0, followed by a sequential feed-forward operation across the remaining devices. Subsequently, back-propagation is performed from device 3 back to device 0. It is observed that when device 1 is processing a data batch, device 0 remains idle, and similarly, when device 2 is processing a data batch, devices 0 and 1 are idle. This represents an inefficiency in resource utilization. To address this, we propose to initiate the feed-forward operation for the next batch on device 0 while device 1 is still processing the current batch. This operation is repeated four times. While this approach does not yield results as optimal as data parallelism, it significantly outperforms model parallelism and closely reaches the speed of data parallelism. Given that it requires only four GPUs and a model size four times smaller than the data parallel strategy, yet achieves nearly the same speed, this method presents an optimal solution for maximizing hardware resources, particularly for smaller research teams.

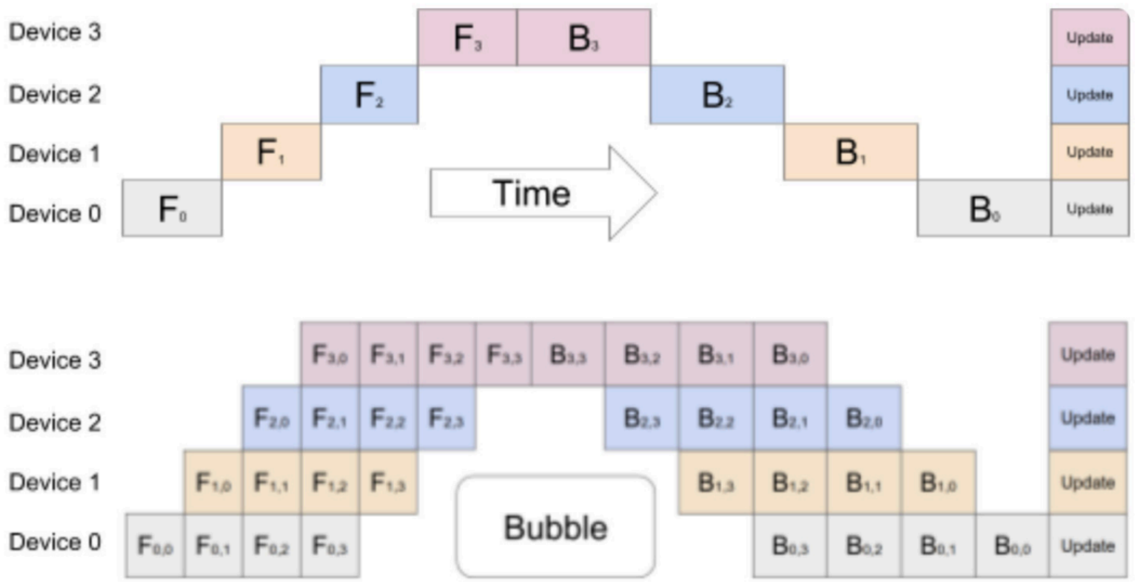

Device 3
Device 2
Device 1
Device 0
F_0 F_1 F_2 F_3 B_3 B_2 B_1 B_0
Update
Time
$F_{0,0}$ $F_{0,1}$ $F_{0,2}$ $F_{0,3}$
$F_{1,0}$ $F_{1,1}$ $F_{1,2}$ $F_{1,3}$
$F_{2,0}$ $F_{2,1}$ $F_{2,2}$ $F_{2,3}$
$F_{3,0}$ $F_{3,1}$ $F_{3,2}$ $F_{3,3}$
$B_{3,3}$ $B_{3,2}$ $B_{3,1}$ $B_{3,0}$
$B_{2,3}$ $B_{2,2}$ $B_{2,1}$ $B_{2,0}$
$B_{1,3}$ $B_{1,2}$ $B_{1,1}$ $B_{1,0}$
$B_{0,3}$ $B_{0,2}$ $B_{0,1}$ $B_{0,0}$
Bubble

Fig 4. *Efficient Machine Learning with Hybrid Parallelism: The diagram shows a hybrid approach using model and data parallelism for efficient machine learning. It optimizes resource utilization and closely matches the speed of data parallelism, ideal for smaller research teams.*

4.4 Choosing checkpoint model

Upon comprehensive analysis of prior research, we have determined that **Flan-T5 backbone** [25] is the most suitable checkpoint model for our study. Flan-T5 is a variant of T5 that robustly enhances the generality of instruction fine tuning compared with non-finetuned models. Flan-T5 model is an advanced iteration of the T5 model, which has been extensively utilized in previous studies on Dialogue State Tracking models. While Flan-T5 retains all the capabilities of its predecessor, it also introduces a host of superior features, making it particularly well-suited for our project. Except that, these flan models also prove zero-shot ability, which significantly influences our paper on experiments with hybrid dialogue. The zero-shot ability of our model is also presented in Table 4.

All previous slot values have to be utilized to compute the JGA score. Here, we clarify that there are two existing formulas. With encoders like FastSGT [1], SGD-base [26], and SGP-DST [27], these model's abilities can only predict the current slot values and have to use another set to store previous ones. FastSGT and SGD-base combine prior predicted slot values with the current expected state to compute the JGA score, while these prior predicted slot values will be replaced by the gold ones on SGP-DST. On the other hand, encoder-decoder seems naive when encouraging the LLM model itself to predict all previous ones, e.g., SDT [4], D3ST [5], AnyTOD [6]. By using encoder-decoder architecture, we mainly use the second formula to compute the JGA score and also provide results in Table 4 and Table 5..

4.4.1 Multitask pre-training

For the Dialogue State Tracking model, we have established a framework that includes three parallel tasks: 1. Classification of Task-Oriented Dialogue (TOD) or Open-Domain Dialogue (ODD), 2. Prediction of user actions and intentions, 3. Prediction of the user request state (these tasks will be elaborated upon in subsequent sections). The necessity for a multitasking pre-trained model is paramount, as it aids in reducing training time and enhancing the efficiency of our model.

Flan-T5 is a multitasking pre-trained model designed for . The model has been scaled to 1,836 fine-tuning tasks by integrating four mixtures derived from previous studies: Muffin, T0-SF, NIV2, and CoT. Muffin includes 80 tasks, comprising 62 existing tasks and an additional 26 new tasks introduced in this study, which include conversation data and program synthesis data. T0-SF consists of 193 tasks, which include tasks from T0 that do not overlap with the data used in Muffin. The remaining tasks are NIV2 (1,554 tasks) and CoT (9 tasks).

In terms of our criteria for selecting a checkpoint model, the ability to pre-train for multitasking is of utmost importance. While Flan-T5 possesses this capability, it is not the only model to do so. Other pre-trained models, including T5, the predecessor of Flan-T5, also have this ability and have been the optimal choice in most previous studies in this field. They examined the assessment results on challenging benchmarks, including: (1) MMLU, which includes exam questions from 57 tasks such as math, history, law, and medicine, (2) BBH, which includes 23 challenging tasks from BIG-Bench, (3) TyDiQA, a question answering benchmark in 8 diverse languages, and (4) MGSM, a multilingual benchmark of word problems manually translated into 10 languages.

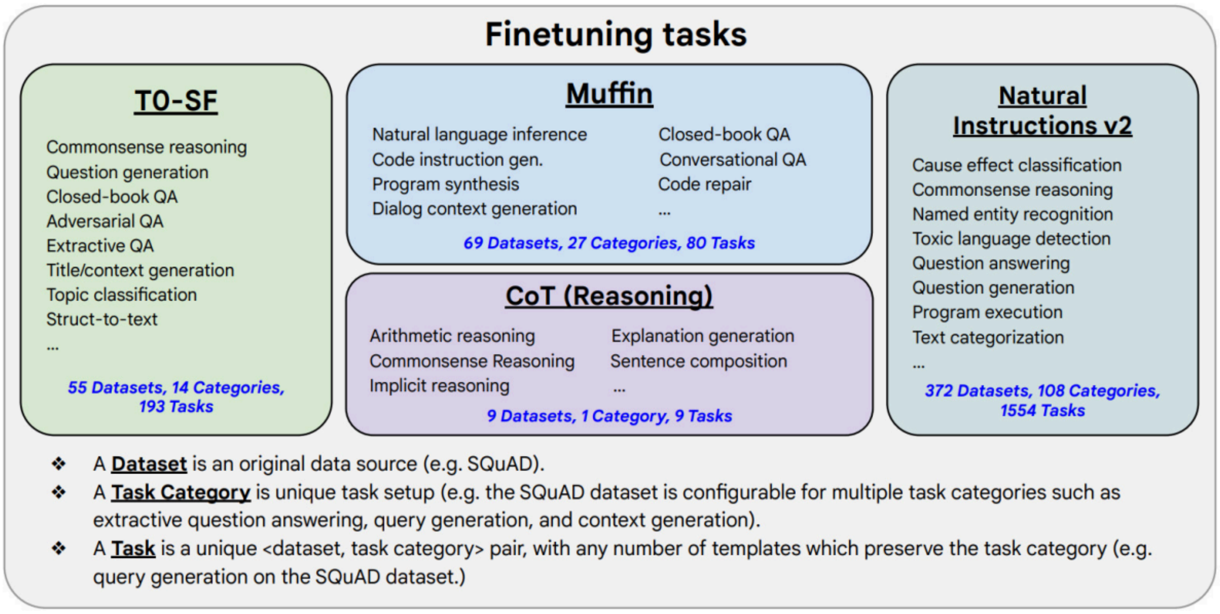
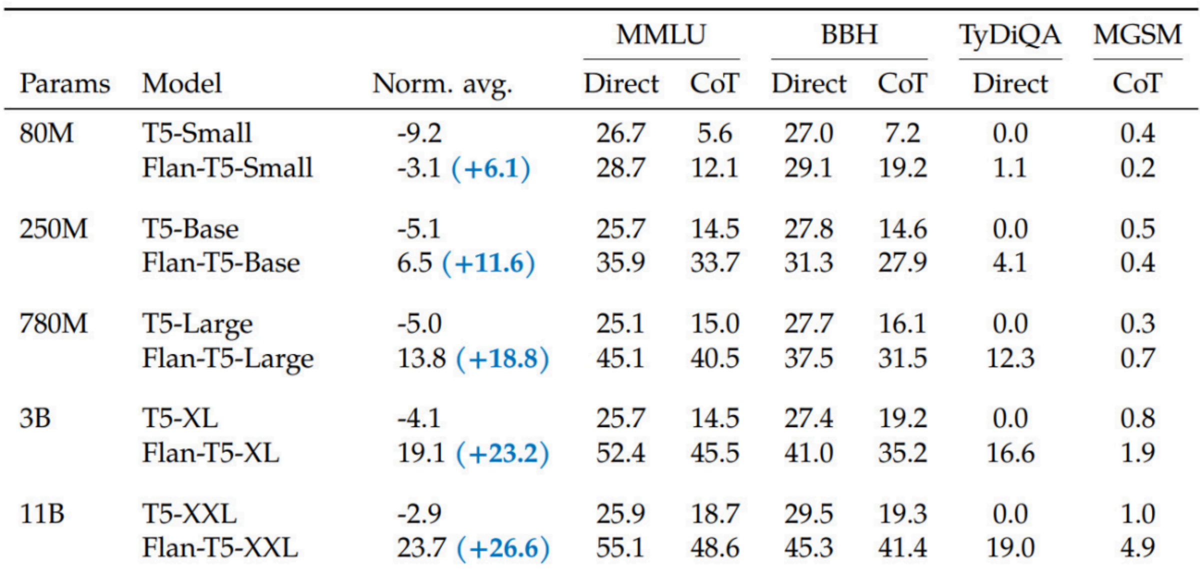

Finetuning tasks
T0-SF
Commonsense reasoning
Question generation
Closed-book QA
Adversarial QA
Extractive QA
Title/context generation
Topic classification
Struct-to-text
...
55 Datasets, 14 Categories, 193 Tasks
Muffin
Natural language inference
Code instruction gen.
Program synthesis
Dialog context generation
Closed-book QA
Conversational QA
Code repair
...
69 Datasets, 27 Categories, 80 Tasks
CoT (Reasoning)
Arithmetic reasoning
Commonsense Reasoning
Implicit reasoning
Explanation generation
Sentence composition
...
9 Datasets, 1 Category, 9 Tasks
Natural Instructions v2
Cause effect classification
Commonsense reasoning
Named entity recognition
Toxic language detection
Question answering
Question generation
Program execution
Text categorization
...
372 Datasets, 108 Categories, 1554 Tasks
❖ A Dataset is an original data source (e.g. SQuAD).
❖ A Task Category is unique task setup (e.g. the SQuAD dataset is configurable for multiple task categories such as extractive question answering, query generation, and context generation).
❖ A Task is a unique <dataset, task category> pair, with any number of templates which preserve the task category (e.g. query generation on the SQuAD dataset.)

Fig 5. *Our fine tuning data comprises 473 datasets, 146 task categories, and 1,836 total tasks.*

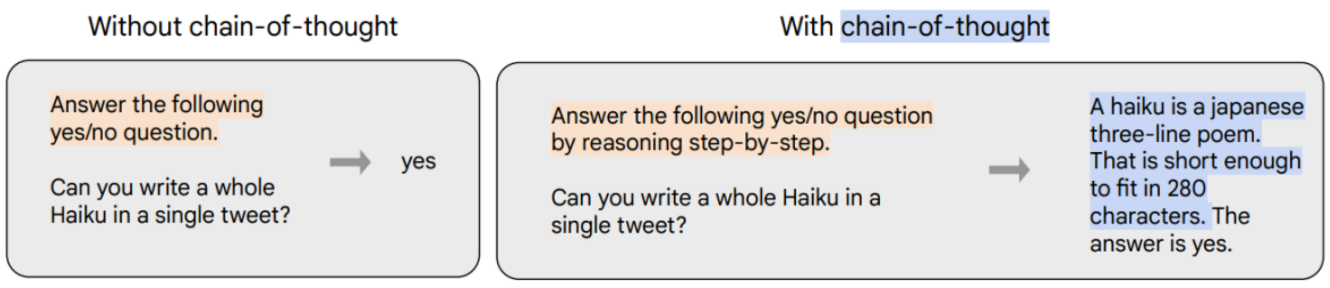


			MMLU		BBH		TyDiQA	MGSM
Params	Model	Norm. avg.	Direct	CoT	Direct	CoT	Direct	CoT
80M	T5-Small	-9.2	26.7	5.6	27.0	7.2	0.0	0.4
	Flan-T5-Small	-3.1 (+6.1)	28.7	12.1	29.1	19.2	1.1	0.2
250M	T5-Base	-5.1	25.7	14.5	27.8	14.6	0.0	0.5
	Flan-T5-Base	6.5 (+11.6)	35.9	33.7	31.3	27.9	4.1	0.4
780M	T5-Large	-5.0	25.1	15.0	27.7	16.1	0.0	0.3
	Flan-T5-Large	13.8 (+18.8)	45.1	40.5	37.5	31.5	12.3	0.7
3B	T5-XL	-4.1	25.7	14.5	27.4	19.2	0.0	0.8
	Flan-T5-XL	19.1 (+23.2)	52.4	45.5	41.0	35.2	16.6	1.9
11B	T5-XXL	-2.9	25.9	18.7	29.5	19.3	0.0	1.0
	Flan-T5-XXL	23.7 (+26.6)	55.1	48.6	45.3	41.4	19.0	4.9

Fig 6. *Results of Flan T5 and T5 on MMLU, BBH, TyDiQA, MGSM.*

4.4.2 Instruction training and Chain of Thought training

Flan T5 is trained on many tasks by using instruction training method, so this checkpoint model's ability to understand context when having to perform unseen tasks will be much better when compared to T5, in the condition that we fine tune it by instruction training method the same way it was pre-trained. In addition, Flan-T5 is also pre-trained using the Chain of Thought training method. The effect of this method is to help the model deduce the steps in the generation process logically and consistently between the steps and between the generated sentences compared to the information provided by the user. For example in the picture... the following is a comparison of using Chain of Thought training and not using Chain of Thought training. We clearly see the answer of using Chain of Thought training as much better than the other.



Without chain-of-thought

Answer the following yes/no question.

Can you write a whole Haiku in a single tweet?

→ yes

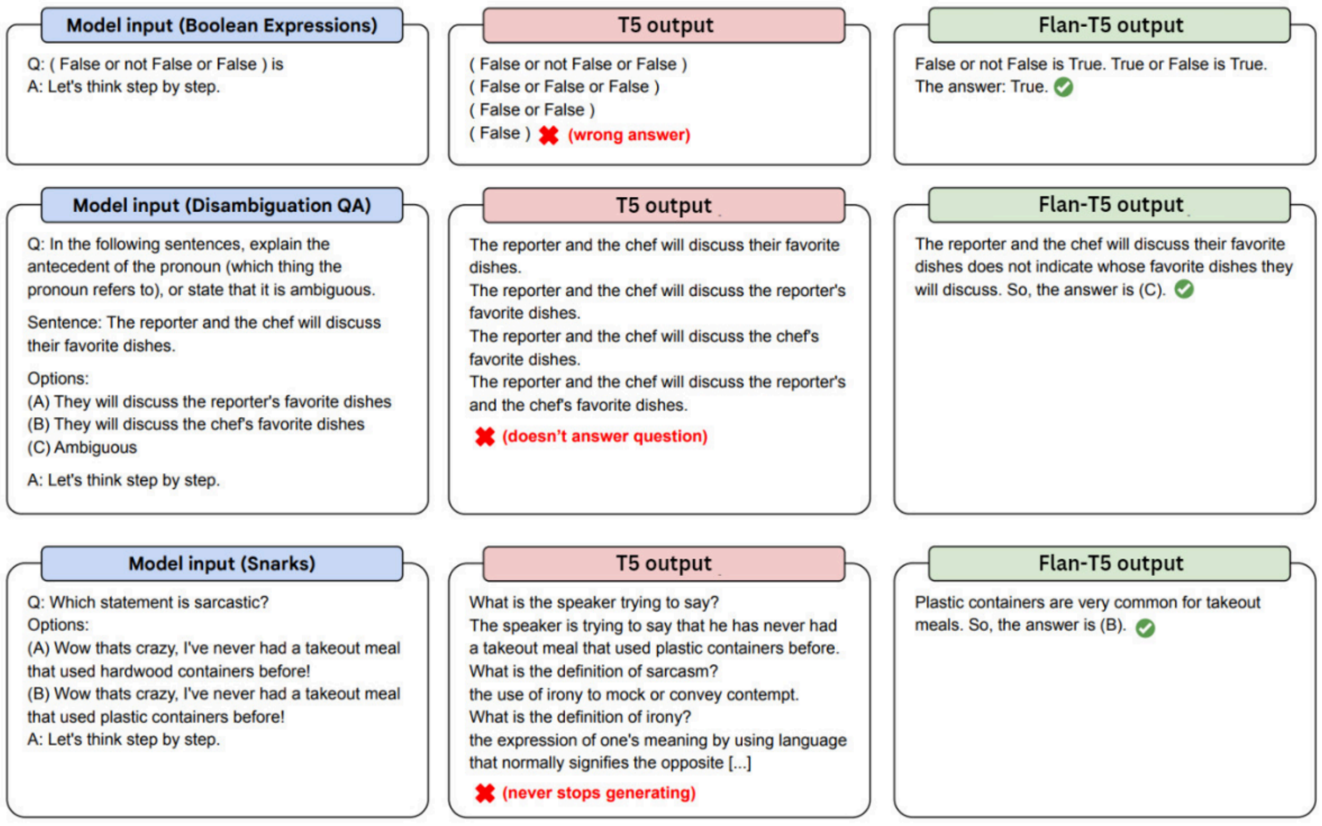
With chain-of-thought

Answer the following yes/no question by reasoning step-by-step.

Can you write a whole Haiku in a single tweet?

→ A haiku is a japanese three-line poem. That is short enough to fit in 280 characters. The answer is yes.

Fig 7. *Compare using chain-of-thought training and not using chain-of-thought training.*



Model input (Boolean Expressions)

Q: (False or not False or False) is
A: Let's think step by step.

T5 output

(False or not False or False)
(False or False or False)
(False or False)
(False) ✗ **(wrong answer)**

Flan-T5 output

False or not False is True. True or False is True. The answer: True. ✓

Model input (Disambiguation QA)

Q: In the following sentences, explain the antecedent of the pronoun (which thing the pronoun refers to), or state that it is ambiguous.

Sentence: The reporter and the chef will discuss their favorite dishes.

Options:
(A) They will discuss the reporter's favorite dishes
(B) They will discuss the chef's favorite dishes
(C) Ambiguous

A: Let's think step by step.

T5 output

The reporter and the chef will discuss their favorite dishes.
The reporter and the chef will discuss the reporter's favorite dishes.
The reporter and the chef will discuss the chef's favorite dishes.
The reporter and the chef will discuss the reporter's and the chef's favorite dishes.

✗ **(doesn't answer question)**

Flan-T5 output

The reporter and the chef will discuss their favorite dishes does not indicate whose favorite dishes they will discuss. So, the answer is (C). ✓

Model input (Snarks)

Q: Which statement is sarcastic?
Options:
(A) Wow thats crazy, I've never had a takeout meal that used hardwood containers before!
(B) Wow thats crazy, I've never had a takeout meal that used plastic containers before!
A: Let's think step by step.

T5 output

What is the speaker trying to say?
The speaker is trying to say that he has never had a takeout meal that used plastic containers before.
What is the definition of sarcasm?
the use of irony to mock or convey contempt.
What is the definition of irony?
the expression of one's meaning by using language that normally signifies the opposite [...]
✗ **(never stops generating)**

Flan-T5 output

Plastic containers are very common for takeout meals. So, the answer is (B). ✓

Fig 8. *Compare using instruction training and not using instruction training.*

4.5 Complexity of building Schema-guided Definition

For previous DST models, they could only operate on seen domains because they had to predict domain, slots which the current user query belonged to. Imagine a situation, the DST model is only trained on 2 seen domains: hotel and hospital. If the user inputs: "Please book me a table at a restaurant in District 1", the output of the DST model may be: inform<hotel -destination-district 1 || inform intent<hotel-intent-ReserveHotel". The reason for the wrong result is because the DST model only consider the domain between hotel and hospital without knowing about the existence of the restaurant domain, and after choosing the wrong domain, it will continue to predict the slot (destination) belongs to the Hotel domain, and assign the value District 1 and that wrong slot. The secret to GranDST's zero shot learning capabilities is that it is provided with an ontology (shema-guided), which is a dictionary containing information about the domain, the slots encoded as digital slots, along with descriptions for those slots. While other DST models have to predict the domain the user is talking about, GradDST is provided in the ontology, so is it wrong with the goal? The answer is no, because this system serves specific businesses, so when users interact with the system, they already know in advance what domain they and the system will chat with. Forcing the DST model to predict the domain the user is referring to is really unnecessary and can also lead to incorrect predictions.

The next issue is why we converted the slots to digital slots, along with descriptions for those slots. So what's more about a conversion like this? Let's compare these two cases.

1. HOTEL:(destination; number of rooms; check in date; number of days; star rating, hotel name; street address; phone number; price per night; has wifi)

2. HOTEL:(slot0=location of the hotel; slot1=number of rooms in the reservation; slot2=start date for the reservation; slot3=number of days in the reservation; slot4=star rating of the hotel; slot5=name of the hotel; slot6=address of the hotel; slot7=phone number of the hotel; slot8=price per night for the reservation; slot9=boolean flag indicating if the hotel has wifi)"

We believe that providing a list of original slots will continue to be a burden for GradDST because slots will use _ to join words in a slot into a new phrase, which will cause the model to learn the meaning of these original slots during training. However, learning to understand these new phrases will become meaningless if it has to be done on unseen domains, because then, the slots in those unseen domains will be new phrases and the model will have to struggle to choose. output a slot that matches the value the user input among a list of slots that have never been learned before. To solve that problem, we add after each slot a description of the meaning of those slots in both training and testing. As stated above, we use checkpoint model Flan-T5 for GradDST with the best context understanding ability at the present time, which will do a very good job of understanding the description of each slot and selecting the appropriate slot with value provided by the user. So, if we already have a description for each slot, then we don't need to leave the original slots alone so that the model has to learn these phrases. Therefore, to generalize, we will convert these slots into digital slots, so that the model only focuses on the following descriptions and does not need to know what the slot is.

GradDST requires building an ontology (shema-guided) from a business domain (single or multi-domains). This ontology helps the DST to understand its meaning and facilitates predicting the user actions and user state. For example, in an agent supporting hotel reservations chatbot, the designer defines the parameters to track along with description as [name=name of the hotel, star=star rating of the accommodation, number of rooms=number of rooms in the reservation, check in date=start date for the reservation,...].

Then, at each turn, the purpose of a DST module is to use the dialogue history up to that turn to predict a dialogue state, which represents the user's goal and progress in using the system. In a similar vein to prior research [4, 5] each element in this schema is characterized by a natural language description, and the entire dialogue flow is designed as a conversation graph flow.

4.6 Predefined structure model

4.6.1 Schema-guided representation

In the context of D3ST, as cited in [5], an ontology serves as a comprehensive dictionary that interprets the descriptions of the slots that have been mapped into ordinal slots. The process of converting slots into numeric slots significantly simplifies the model's task of performing operations for unseen domains. This is primarily because the model relies on descriptions without necessitating an understanding of the names of the slots in those domains.

An ontology encompasses all the domains present in the dialogue. Consequently, the model must learn to select domains that align with the current turn. Furthermore, the ontology is appended to the domain name at the commencement of each dictionary chain. This facilitates the model in identifying the domains at the current turn with greater ease.The overview representation is denoted as

$$O = \{s_0^{D_0} = d_0^{D_0}, s_1^{D_0} = d_1^{D_0}, \dots, s_N^{D_0} = d_N^{D_0}, \dots, s_N^{D_M} = d_N^{D_M}\}$$

where M and N are the number of domains and slots of each domain, respectively.

This representation provides a comprehensive overview of the model's structure and functioning, thereby enhancing its efficiency and effectiveness in handling task oriented dialogues. This approach underscores the pivotal role of ontologies in improving the performance of task-oriented dialogue systems.

4.6.2 Dialogue context representation

In alignment with the methodologies employed in previous research [28, 29], we designate each user's utterance as the current query sample. The remaining history is preserved as the context dialogue for each utterance. To distinguish the speaker of each turn, we append the prefix tokens ”user” and ”system” to each utterance.

Ultimately, a comprehensive dialogue context is constructed by concatenating all the system and user utterances. This context is represented as

$$C = \{u_0, s_0, \ldots, u_{t-1}, s_{t-1}\}$$

where 't' represents the current query's position in the sub-dialogue sample.

This approach ensures a systematic and organized representation of the dialogue, facilitating efficient processing and analysis.

4.6.3 Action enhances constraint

In the final stage of our process, we construct a list of user actions that will be incorporated into the input template for each turn. Given that our label contains the current action, this list of user actions serves to assist the model in selecting the most appropriate action. This, in turn, reduces the likelihood of predicting an action that is not present in the raw datasets. The formula for this is represented as

$$A = \{\text{"inform, request, ..., thank, general\_asking"}\}$$

Leveraging the advantage of the Flan-T5 checkpoint, which is based on instruction based training, we integrate Machine Reading Comprehension with instructions. This integration provides detailed explanations of tasks, thereby ensuring that the model has a clear understanding of the tasks it needs to perform and the manner in which they should be executed. In the instructions, we format the detailed content input by incorporating the aforementioned context, query, ontology, and user actions. The format of this integration is presented in Figure 9. This comprehensive approach ensures that our model is well-equipped to handle a wide range of tasks effectively and efficiently.

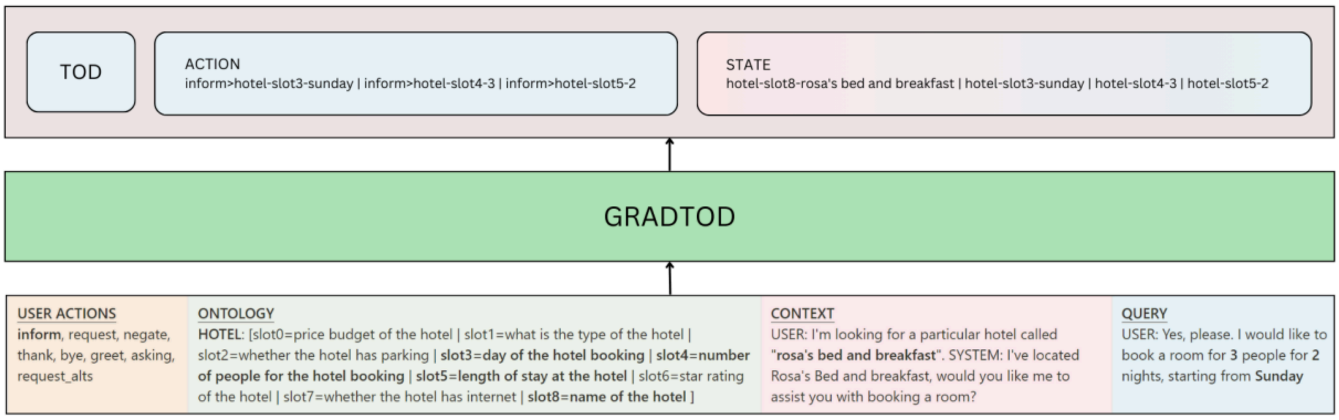

TOD
ACTION
inform>hotel-slot3-sunday | inform>hotel-slot4-3 | inform>hotel-slot5-2
STATE
hotel-slot8-rosa's bed and breakfast | hotel-slot3-sunday | hotel-slot4-3 | hotel-slot5-2
GRADTOD
USER ACTIONS
inform, request, negate, thank, bye, greet, asking, request_alts
ONTOLOGY
HOTEL: [slot0=price budget of the hotel | slot1=what is the type of the hotel | slot2=whether the hotel has parking | slot3=day of the hotel booking | slot4=number of people for the hotel booking | slot5=length of stay at the hotel | slot6=star rating of the hotel | slot7=whether the hotel has internet | slot8=name of the hotel]
CONTEXT
USER: I'm looking for a particular hotel called "rosa's bed and breakfast". SYSTEM: I've located Rosa's Bed and breakfast, would you like me to assist you with booking a room?
QUERY
USER: Yes, please. I would like to book a room for 3 people for 2 nights, starting from Sunday

Fig 9. *Overview of GradBot approach for schema-guided multi-domains dialogues. The bottom figure includes specific examples for dialogue context, user action, ontology and current query while the top figure stimulates predictions.*

4.7 DST as guided Reading Conversation

GradTOD, our model, is concurrently trained on three distinct tasks:

- **Type Classification:** This task involves classifying the dialogue as either TaskOriented Dialogue (TOD) or Open Domain Dialogue (ODD).
- **Current Action Prediction**: This task focuses on predicting the user's actions for each turn.
- **Current State Prediction:** This task aims to predict the state for each turn.

Each dialogue in the utilized chat dataset contains chitchat utterances. Therefore, in addition to defining the state like other models, we incorporate an additional task to identify the type (TOD/ODD). This task is designed to evaluate the model's ability to distinguish between general conversation and task-oriented conversation. Moreover, the current action is appended to the label to establish a strong correlation between the user's action and the user's state. Through this process, the model learns that if the user's action is 'inform', the state will be updated. However, actions such as 'request', 'thank', 'bye', and others will maintain the same state as the previous turn. This multi-task training approach equips GradTOD with the capability to handle complex dialogues effectively, thereby enhancing its performance and usability in various applications.

5 DATASETS

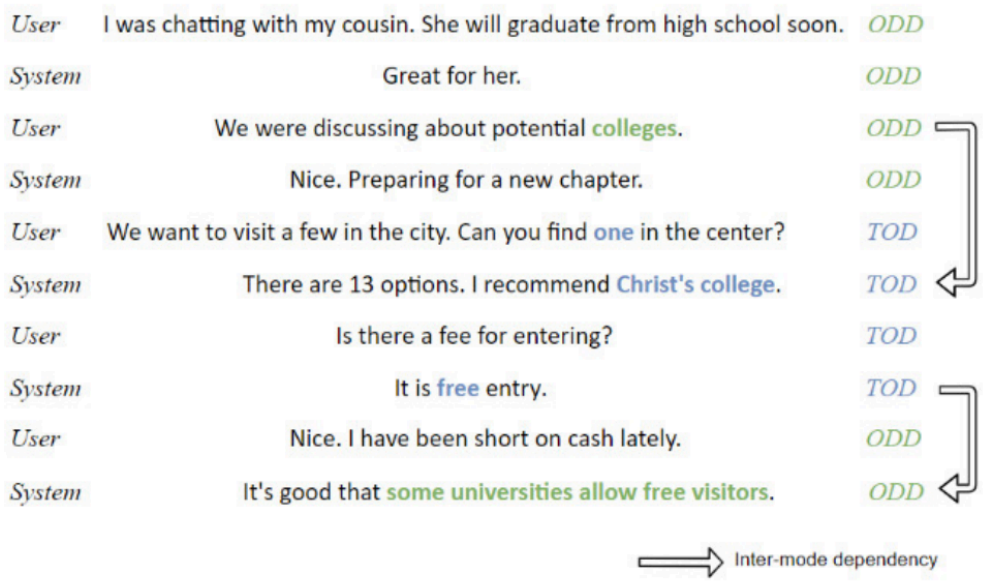
Statistics / Dataset	**FusedChat**	**SGD**	**MultiWoz2.4**
No. of domains	7	20	7
No. of dialogues	10348	22825	10348
Avg. turn per dialogue	20.3	18.36	12.27
No. of slots	45	*184*	37
No. of slot values	1936	**500**	1936

Table 3. *Statistics for SGD, FusedChat, and MultiWoz2.4, computed across train, validation, and test sets. FusedChat incorporates MultiWoz2.4, with the addition of ODD to its TOD part. In SGD, "unique" slots are represented in italics, and the number of slot values includes those for "categorical" slots.*

5.1 Fusedchat

5.1.1 Overview

Fusedchat dataset [23] is an essential dataset created by merging task-oriented dialogue (TOD) and open-domain dialogue (ODD). This amalgamation of ODD and TOD fosters a seamless connection and robust contextual interplay between the two dialogue modes. Hence, this dataset can support the backbone system to accelerate comprehension when dealing with reality. As an extension of the renowned MultiWoz dataset [30], Fusedchat integrates additional ODD turns either before or after the existing TOD turns, with 3670 and 4768 instances correspondingly according to [23]. Besides, due to the noise Fusedchat dataset itself, e.g. redundant domains, and inconsistency values, which were inherited from MultiWoz2.4 [31]. Indeed, a dialogue model that allows for rich interactions between two dialogue modes can significantly enhance the conversational experience. By not being confined to a single mode, the model can better emulate human-level conversation capabilities. For instance, it can facilitate a more engaging and dynamic interaction, akin to chatting with a friendly assistant (in Figure 10).



User	I was chatting with my cousin. She will graduate from high school soon.	*ODD*
System	Great for her.	*ODD*
User	We were discussing about potential colleges.	*ODD*
System	Nice. Preparing for a new chapter.	*ODD*
User	We want to visit a few in the city. Can you find one in the center?	*TOD*
System	There are 13 options. I recommend Christ's college.	*TOD*
User	Is there a fee for entering?	*TOD*
System	It is free entry.	*TOD*
User	Nice. I have been short on cash lately.	*ODD*
System	It's good that some universities allow free visitors.	*ODD*

Fig 10. *Their dialogue system allows a user and a digital assistant to switch between Task-Oriented Dialogue (TOD) and Open-Domain Dialogue (ODD) modes. An example includes a query about college fees (TOD) and a chat about personal growth and finance (ODD).*

5.1.2 Fusedchat Construction

In the process of constructing our dataset for inter-mode dialogue sessions, we primarily engage dialogue creators to append or prepend self-authored Open-Domain Dialogues (ODDs) to existing Task-Oriented Dialogues (TODs). The dialogue creator assumes the roles of both the user and the dialogue system.

For the existing TODs, they have chosen the MultiWOZ2.4 dataset due to its widespread recognition in the field. MultiWOZ encompasses TODs across seven domains, including restaurants, attractions, trains, police, hospitals, taxis, and hotels. Users interact with the dialogue agent to perform predefined functions, such as booking restaurants or locating hospitals. In FusedChat setting, dialogue creators have the freedom to add any ODD that is contextually consistent with the existing TOD.

5.1.2.1 General requirements for the added ODDs

This section outlines the guidelines for dialogue creators in adding Open-Domain Dialogues (ODDs) to existing Task-Oriented Dialogues (TODs).

- **Role-playing**: Creators write fictitious ODDs, playing both the "system" (an AI conversational agent capable of open-domain and task-oriented dialogues) and the "user" (a human speaker interacting with the AI for casual conversation and task completion).
- **Relevance**: To maintain relevance between the TOD and added ODD, creators are encouraged to center the ODD around similar or related topics as in the TOD. The added ODD and existing TOD should connect naturally, with strong contextual dependency.
- **Characteristics of ODDs**: The created dialogues should reflect the casual, non task-specific nature of ODDs, as opposed to the task-oriented nature of TODs.

Based on a pilot experiment, they found that creators often wrote dialogues focused on task-specific functionalities, which are technically TODs, not ODDs. To address this, they deployed a real-time turn-level ODD vs TOD classifier, trained on three traditional ODD datasets and MultiWOZ. They also provided creators with guidelines to avoid pitfalls, such as fabricating information beyond common sense.

5.1.2.2 Appending ODDs

In the appending scenario, dialogue creators add Open-Domain Dialogues (ODDs) to existing Task-Oriented Dialogues (TODs) from the MultiWOZ dataset. The added ODD should seamlessly follow the TOD.They observed that dialogues in the original MultiWOZ dataset often conclude with a "User: Thank you. System: Goodbye." exchange. To facilitate the appending of ODDs, they remove such exchanges from the end of the TOD based on dialogue act annotations.The content of the appended ODD should be contextually dependent on the preceding TOD. This is ensured by having creators write at least one round of exchange that reflects concepts or knowledge from the existing TOD segment. For instance in Figure 11, in a dialogue about querying and booking a Thai restaurant, the user might express concern about whether their friends would enjoy the restaurant. This is considered an ODD utterance as it does not invoke any task-oriented function. The system's ODD response, supported by commonsense and empathy, reflects content from a previous TOD turn.

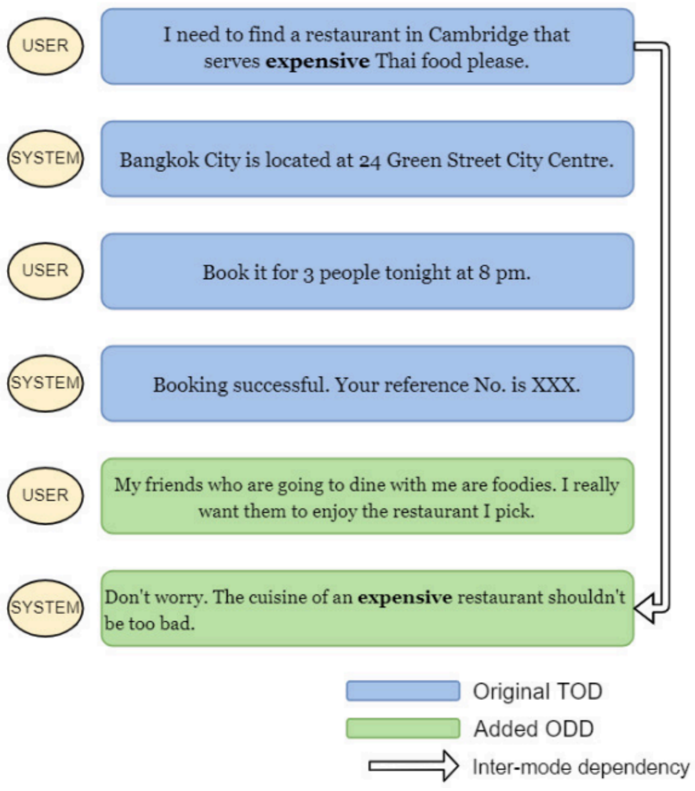

USER
I need to find a restaurant in Cambridge that serves **expensive** Thai food please.
SYSTEM
Bangkok City is located at 24 Green Street City Centre.
USER
Book it for 3 people tonight at 8 pm.
SYSTEM
Booking successful. Your reference No. is XXX.
USER
My friends who are going to dine with me are foodies. I really want them to enjoy the restaurant I pick.
SYSTEM
Don't worry. The cuisine of an **expensive** restaurant shouldn't be too bad.
Original TOD
Added ODD
Inter-mode dependency

***Fig 11.** An TOD + ODD instance from FusedChat*

5.1.2.3 Prepending ODDs

In the prepending scenario, dialogue creators are given a Task-Oriented Dialogue (TOD) segment from the MultiWOZ dataset and asked to prepend an Open-Domain Dialogue (ODD) to it. The ODD should naturally lead into the provided TOD. The original TODs in MultiWOZ are self-contained. To model inter-mode dependency, we conduct utterance rewriting based on co-reference and ellipsis, which are key to making the TOD dependent on the prepended ODD. They aim to create ODD + TOD sessions where the TOD is conditioned on the ODD. This involves dialogue state tracking, where the dialogue system processes the user utterance for [slot type, slot value] pairs (e.g., [Destination: Cambridge]) to understand the user's needs and respond appropriately (in Figure 12). Their method for modeling inter-mode dependency involves ODD-dependent dialogue state tracking. We randomly select a slot value mentioned in the first user turn in the TOD and ask the dialogue creators to use this slot value in the prepended ODD. The first dialogue user turn is then rewritten to refer to it implicitly. This rewriting mainly involves co-reference and sometimes ellipsis, both of which are important features in multi-turn TODs.

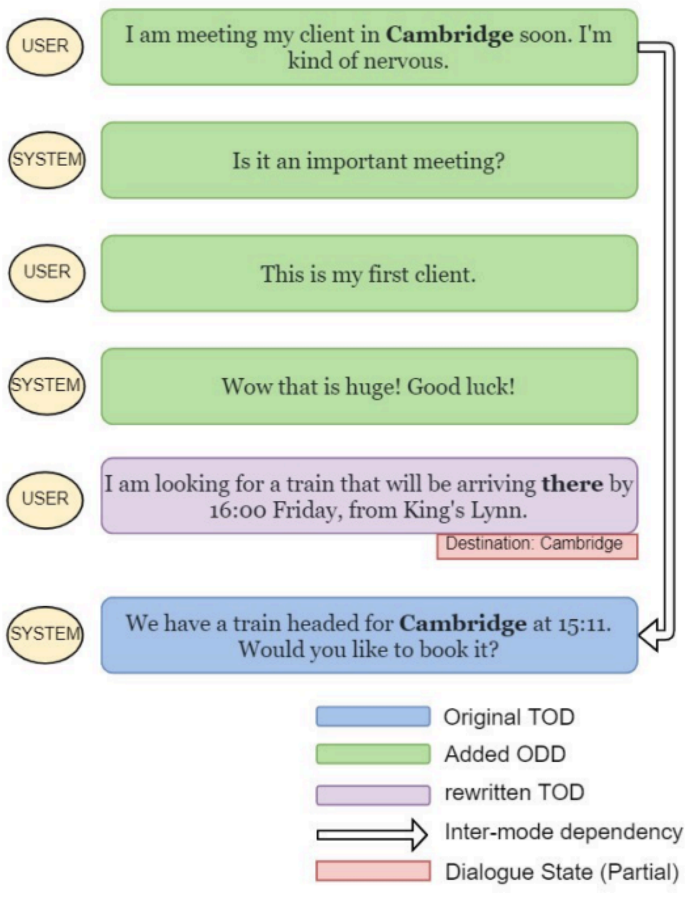
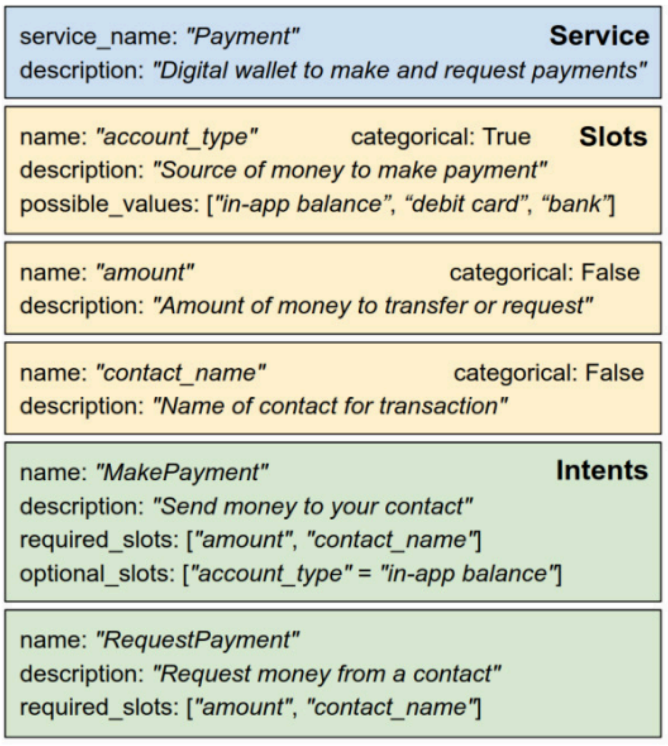

USER
I am meeting my client in **Cambridge** soon. I'm kind of nervous.
SYSTEM
Is it an important meeting?
USER
This is my first client.
SYSTEM
Wow that is huge! Good luck!
USER
I am looking for a train that will be arriving **there** by 16:00 Friday, from King's Lynn.
Destination: Cambridge
SYSTEM
We have a train headed for **Cambridge** at 15:11. Would you like to book it?
Original TOD
Added ODD
rewritten TOD
Inter-mode dependency
Dialogue State (Partial)

Fig 12. *An ODD + TOD instance from FusedChat*

5.2 Schema-Guided Dialogue (SGD)

5.2.1 Overview Schema-Guided

Dialogue dataset [26] is the largest multidomain for task-oriented dialogue datasets until now. It spans 45 diverse domains over hotels, banks, events, homes, travel, flights, media, movies, rental cars, and more. Each part continues to split into various forms, e.g., hotels decomposed as hotels 1, hotels 2, hotels 3, and so on. This dataset utilizes 25 domains for training and reuses the identical domains combined with 10 more ones for validation. This large number of dialogues will offset the minor number one in the Fusedchat dataset. The test set also comprises these 35 domains to evaluate the model's zero-shot learning capabilities.



Service
service_name: *"Payment"*
description: *"Digital wallet to make and request payments"*

Slots
name: *"account_type"* categorical: True
description: *"Source of money to make payment"*
possible_values: [*"in-app balance", "debit card", "bank"*]

name: *"amount"* categorical: False
description: *"Amount of money to transfer or request"*

name: *"contact_name"* categorical: False
description: *"Name of contact for transaction"*

Intents
name: *"MakePayment"*
description: *"Send money to your contact"*
required_slots: [*"amount", "contact_name"*]
optional_slots: [*"account_type" = "in-app balance"*]

name: *"RequestPayment"*
description: *"Request money from a contact"*
required_slots: [*"amount", "contact_name"*]

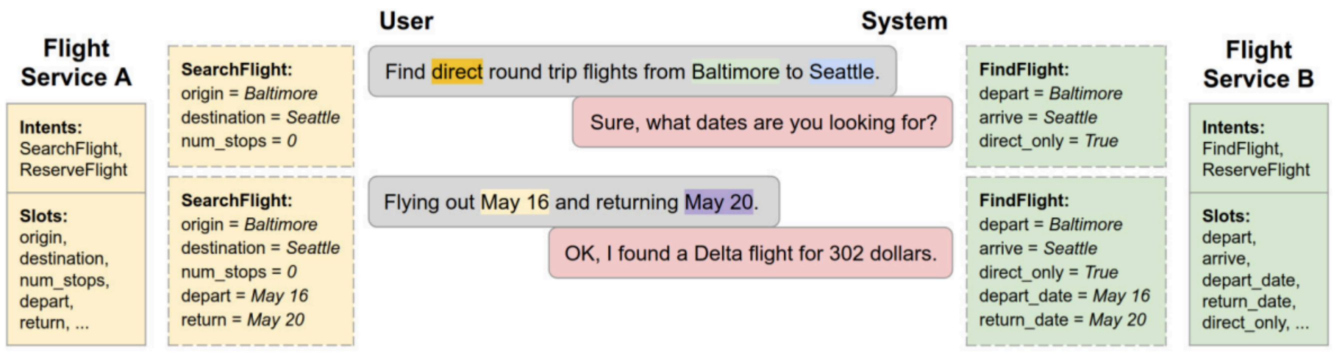
Fig 13. *Example schema for a digital wallet service*

5.2.2 Schema-Guided Approach

In the Schema-Guided approach, each service offers a schema that enumerates the supported slots and intents, accompanied by their natural language descriptions (in Figure 13). The dialogue annotations are directed by the schema of the underlying service or API (in Figure 14).

For instance, the departure and arrival cities are captured by slots in both schemas that function analogously but bear different names. Moreover, the values for the number of stops and direct-only slots underscore the idiosyncrasies between services interpreting the same concept.

The natural language descriptions in the schema are utilized to derive a semantic representation of intents and slots. The assistant employs a single, unified model devoid of domain or service-specific parameters to make predictions based on these schema elements. The use of a single model facilitates the representation and transfer of common knowledge across related concepts in different services. As the model uses the semantic representation of schema elements as input, it can interface with unseen services or APIs on which it has not been trained. It also exhibits robustness to changes such as the addition of new intents or slots to the service.



Flight Service A
Intents: SearchFlight, ReserveFlight
Slots: origin, destination, num_stops, depart, return, ...
SearchFlight: origin = Baltimore destination = Seattle num_stops = 0
SearchFlight: origin = Baltimore destination = Seattle num_stops = 0 depart = May 16 return = May 20
User
System
Find direct round trip flights from Baltimore to Seattle.
Sure, what dates are you looking for?
Flying out May 16 and returning May 20.
OK, I found a Delta flight for 302 dollars.
FindFlight: depart = Baltimore arrive = Seattle direct_only = True
FindFlight: depart = Baltimore arrive = Seattle direct_only = True depart_date = May 16 return_date = May 20
Flight Service B
Intents: FindFlight, ReserveFlight
Slots: depart, arrive, depart_date, return_date, direct_only, ...

Fig 14. *In the context of two distinct flight services, dialogue state tracking labels are applied after each user statement. With the schema-guided method, these annotations depend on the service's schema, located at the extreme left/right.*

5.2.3 Data Representation

The dataset in question comprises dialogues between a virtual assistant and a user. These dialogues can encompass multiple services spanning a variety of domains. Each dialogue is structured as a sequence of turns, with each turn containing an utterance from either the user or the system.Annotations for each turn are organized into frames, with each frame corresponding to a specific service. For user turns, the annotations include the active intent, the dialogue state, and slot spans for the various slot values mentioned in the turn. For system turns, the annotations consist of system actions that represent the semantics of the system utterance. Each system action is depicted using a dialogue act, which may have optional parameters. In addition to the dialogues, a normalized representation of the interface exposed is provided as a schema for each service used in the dataset. This schema includes details such as the name of the service, the list of tasks supported by the service (intents), and the attributes of the entities used by the service (slots).The schema also contains natural language descriptions of the service, intents, and slots. These descriptions can be utilized for the development of models that can condition their predictions on the schema. This professional description provides a comprehensive overview of the dataset's structure and content.

5.2.4 Comparison With Other Datasets

In order to mirror the constraints inherent in real-world services and APIs, they have imposed certain limitations on our dataset. For instance, SGD does not disclose the complete set of potential values for specific slots. This is because it is impractical to have such a list for slots like date or time, which have an infinite number of possible values, or for slots like movie or song names, which are periodically updated with new values. These slots are specifically designated as non-categorical slots. In our evaluation sets, they ensured the inclusion of a significant number of values that were not previously seen in the training set. This was done to assess the performance of models on unseen values. Certain slots, such as gender and number of people, are classified as categorical, and we provide a list of all possible values for these slots. However, these values are not assumed to be consistent across services. For example, different services may use ('male', 'female'), ('M', 'F'), or ('he', 'she') as possible values for the gender slot.

Real-world services can only be invoked with specific slot combinations. For instance, most restaurant reservation APIs do not allow users to search for restaurants by date without specifying a location. While this constraint does not impact the dialogue state tracking task, it does limit the possible conversational flows. Therefore, to prevent flows that are not supported by actual services, they restrict services to be called with a list of slot combinations. The different service calls supported by a service are listed as intents, with each intent specifying a list of required slots. An intent cannot be called without providing values for these required slots. Each intent also contains a list of optional slots with default values, which can be overridden by the user. In SGD, they also have multiple services per domain with overlapping functionality. The intentions across these services are similar but differ in terms of intent names, intent arguments, slot names, etc. In some cases, there is no one-to-one mapping between slot names (e.g., the ’num stops’ and ’direct only’ slots in Figure 14). With an ever-increasing number of services and service providers, they believe that having multiple similar services per domain is much closer to the situation faced by virtual assistants than having one unique service per domain.

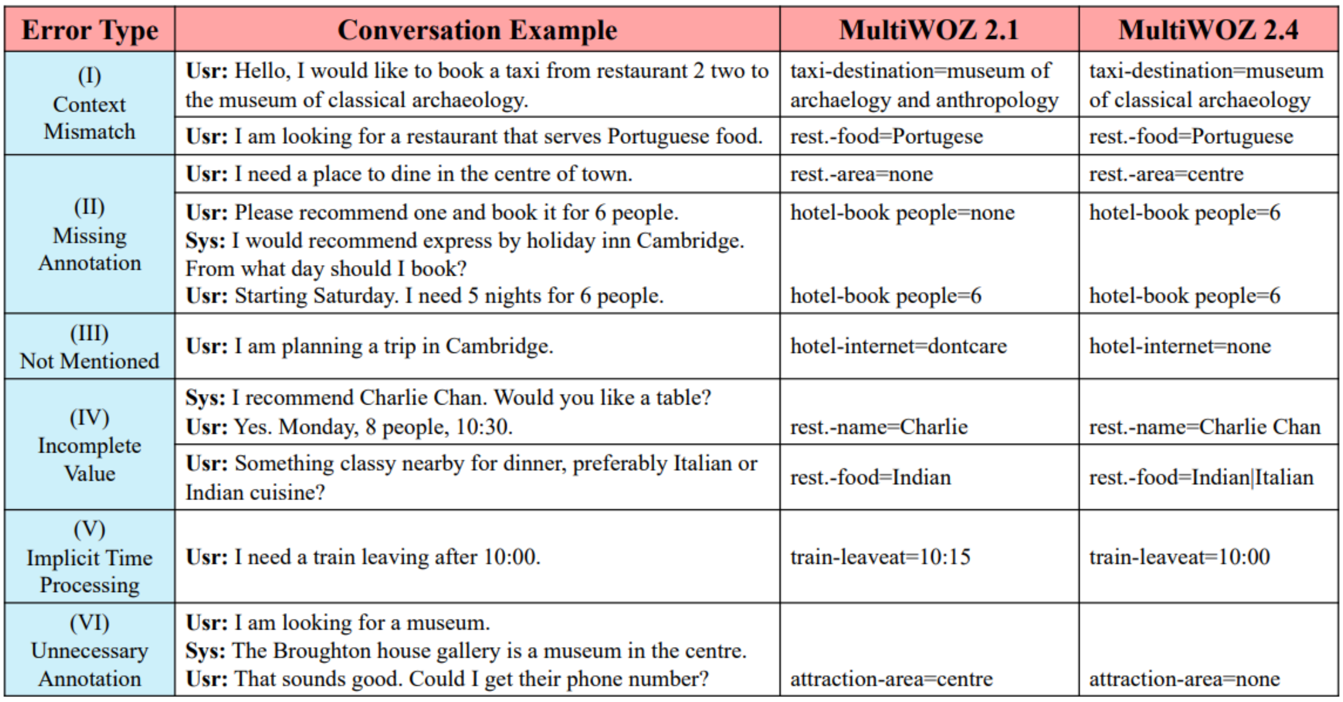
5.3 MultiWoz 2.4

5.3.1 Overview

MultiWoz2.4 dataset [31] is the last refinement for primarily evaluating metrics on task-oriented dialogue up-to-date. Statistics are the same as preceding MultiWoz versions. Because all annotation updates are refined mainly on the validation set and test set, as well as having large numbers of dialogues spanning many domains, we will use this dataset to evaluate zero-shot abilities and compare our model with other backbones. Though MultiWoz2.4 is built originally from version 2.1, which does not leverage versions 2.2 and 2.3, it still has a renowned publication and has been widely used when evaluating research in recent years.

5.3.2 Annotation Error Types

Dialogue state tracking primarily aims to monitor the user’s utterances, with the dialogue state predominantly relying on these utterances. In accordance with this premise, we have identified and rectified six types of annotation errors in the validation and test sets of MultiWOZ 2.1.



Error Type	Conversation Example	MultiWOZ 2.1	MultiWOZ 2.4
(I) Context Mismatch	**Usr:** Hello, I would like to book a taxi from restaurant 2 two to the museum of classical archaeology.	taxi-destination=museum of archaelogy and anthropology	taxi-destination=museum of classical archaeology
	Usr: I am looking for a restaurant that serves Portuguese food.	rest.-food=Portugese	rest.-food=Portuguese
(II) Missing Annotation	**Usr:** I need a place to dine in the centre of town.	rest.-area=none	rest.-area=centre
	Usr: Please recommend one and book it for 6 people. **Sys:** I would recommend express by holiday inn Cambridge. From what day should I book? **Usr:** Starting Saturday. I need 5 nights for 6 people.	hotel-book people=none hotel-book people=6	hotel-book people=6 hotel-book people=6
(III) Not Mentioned	**Usr:** I am planning a trip in Cambridge.	hotel-internet=dontcare	hotel-internet=none
(IV) Incomplete Value	**Sys:** I recommend Charlie Chan. Would you like a table? **Usr:** Yes. Monday, 8 people, 10:30.	rest.-name=Charlie	rest.-name=Charlie Chan
	Usr: Something classy nearby for dinner, preferably Italian or Indian cuisine?	rest.-food=Indian	rest.-food=Indian|Italian
(V) Implicit Time Processing	**Usr:** I need a train leaving after 10:00.	train-leaveat=10:15	train-leaveat=10:00
(VI) Unnecessary Annotation	**Usr:** I am looking for a museum. **Sys:** The Broughton house gallery is a museum in the centre. **Usr:** That sounds good. Could I get their phone number?	attraction-area=centre	attraction-area=none

Fig 15*. Examples of each error type.*

The Figure 15 are the identified error types:

- **Context Mismatch**: The slot value does not align with the dialogue context. This category also includes values with typographical errors.
- **Missing Annotation:** The slot is not labeled, despite its value being mentioned. In some instances, the annotations are deferred to subsequent turns.
- **Not Mentioned:** The slot has been annotated, but its value has not been mentioned.
- **Incomplete Value**: The slot value is either a substring or an abbreviation of its full form (e.g., "Thurs" vs. "Thursday"). In certain cases, the slot should contain multiple values, but not all values are included.
- **Implicit Time Processing:** This pertains to slots that take time as a value. Rather than replicating the time specified in the dialogue context, the value has been implicitly processed (e.g., adding 15 minutes).
- **Unnecessary Annotation:** These superfluous annotations exacerbate inconsistencies as different annotators have varying opinions on whether to annotate these slots. Generally,

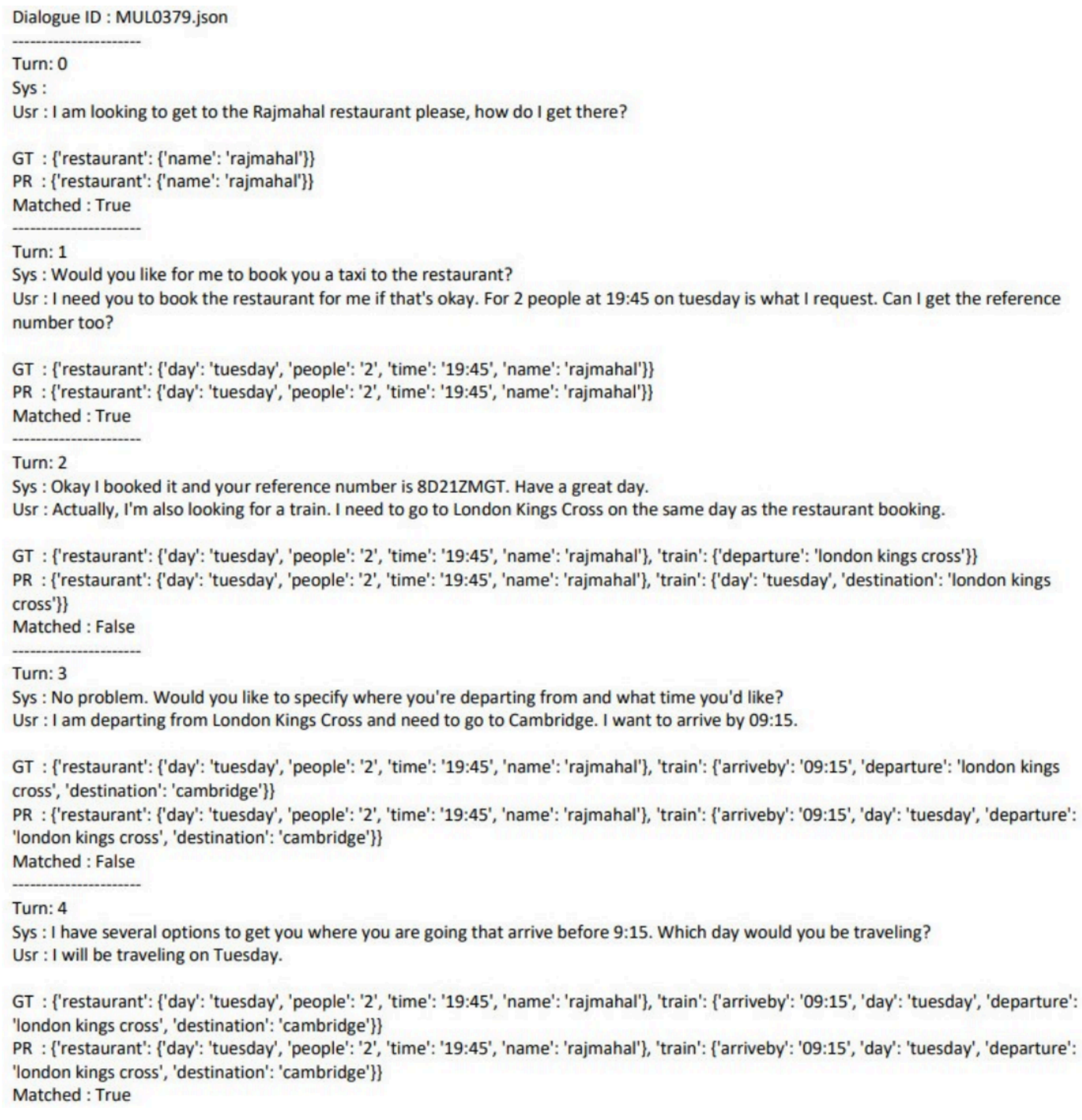
the values of these slots are mentioned by the system to respond to previous user requests or provide supplementary information. We found that in most dialogues, these slots are not annotated. Consequently, we remove these annotations. However, name-related slots are an exception. If the user requests additional information (e.g., address and postcode) about the recommended ”name”, the slots will be annotated.

6 METRICS

6.1 Joint Goal Accuracy

Joint target accuracy (JGA) is a widely recognized and frequently used metric in the field of conversation state monitoring. Its definition, as established in SGD [25], its effectiveness is very good in evaluating the performance of models in this field. The basic principle of JGA is to compare the model's belief state prediction with the actual truth label. A prediction is considered correct if it matches the truth label exactly. This strict requirement for accuracy emphasizes the precision and accuracy required by the JGA, thus making it a reliable measure. However, it is important to note that this precision also introduces certain limitations to the JGA metric. One such limitation is that the truth label includes all states from the previous state. This means that for a prediction to be considered accurate, the model needs to accurately predict not only the current state but also positions from previous dialogue turns. This requirement increases the difficulty, causing this metric's score to often be very low compared to other metrics. To elucidate the concept of Joint Goal Accuracy (JGA) in a more comprehensive and professional manner, let us consider a specific dialogue instance, identified by the ID MUL0379, which is sourced from the Fusedchat dataset (Fig. 15). This particular example is referenced from a scholarly article detailing their TRADE model, and it serves as an illustrative case for the application of the JGA metric. In the graphical representation associated with this dialogue, there are a total of five conversational turns. Within this context, 'GT' denotes the symbol representing the ground-truth belief states, while 'PR' signifies the symbol for the predicted belief states as designated by them. The criterion for a 'Matched' status being recorded as 'True' is predicated on the complete compatibility of the slot and value pairs between GT and PR. Conversely, the presence of even a singular discrepancy results in a 'Matched' status being recorded as 'False'.

A closer examination of the TRADE model's performance in this instance reveals that during turns 0, 1, and 4, there is a complete alignment between the model's predictions and the goal labels, thereby warranting a 'Matched' status of 'True' for these turns. However, for turns 2 and 3, the requirement is such that the emergence of at least one mismatched pair of slots and values will suffice to record a 'Matched' status of 'False'.



```
Dialogue ID : MUL0379.json
----------------------
Turn: 0
Sys :
Usr : I am looking to get to the Rajmahal restaurant please, how do I get there?

GT  : {'restaurant': {'name': 'rajmahal'}}
PR  : {'restaurant': {'name': 'rajmahal'}}
Matched : True
----------------------
Turn: 1
Sys : Would you like for me to book you a taxi to the restaurant?
Usr : I need you to book the restaurant for me if that's okay. For 2 people at 19:45 on tuesday is what I request. Can I get the reference
number too?

GT  : {'restaurant': {'day': 'tuesday', 'people': '2', 'time': '19:45', 'name': 'rajmahal'}}
PR  : {'restaurant': {'day': 'tuesday', 'people': '2', 'time': '19:45', 'name': 'rajmahal'}}
Matched : True
----------------------
Turn: 2
Sys : Okay I booked it and your reference number is 8D21ZMGT. Have a great day.
Usr : Actually, I'm also looking for a train. I need to go to London Kings Cross on the same day as the restaurant booking.

GT  : {'restaurant': {'day': 'tuesday', 'people': '2', 'time': '19:45', 'name': 'rajmahal'}, 'train': {'departure': 'london kings cross'}}
PR  : {'restaurant': {'day': 'tuesday', 'people': '2', 'time': '19:45', 'name': 'rajmahal'}, 'train': {'day': 'tuesday', 'destination': 'london kings
cross'}}
Matched : False
----------------------
Turn: 3
Sys : No problem. Would you like to specify where you're departing from and what time you'd like?
Usr : I am departing from London Kings Cross and need to go to Cambridge. I want to arrive by 09:15.

GT  : {'restaurant': {'day': 'tuesday', 'people': '2', 'time': '19:45', 'name': 'rajmahal'}, 'train': {'arriveby': '09:15', 'departure': 'london kings
cross', 'destination': 'cambridge'}}
PR  : {'restaurant': {'day': 'tuesday', 'people': '2', 'time': '19:45', 'name': 'rajmahal'}, 'train': {'arriveby': '09:15', 'day': 'tuesday', 'departure':
'london kings cross', 'destination': 'cambridge'}}
Matched : False
----------------------
Turn: 4
Sys : I have several options to get you where you are going that arrive before 9:15. Which day would you be traveling?
Usr : I will be traveling on Tuesday.

GT  : {'restaurant': {'day': 'tuesday', 'people': '2', 'time': '19:45', 'name': 'rajmahal'}, 'train': {'arriveby': '09:15', 'day': 'tuesday', 'departure':
'london kings cross', 'destination': 'cambridge'}}
PR  : {'restaurant': {'day': 'tuesday', 'people': '2', 'time': '19:45', 'name': 'rajmahal'}, 'train': {'arriveby': '09:15', 'day': 'tuesday', 'departure':
'london kings cross', 'destination': 'cambridge'}}
Matched : True
```

Fig 16. *Overview simultaneously enhances the construct meaning of the input and target value.*

Assuming a hypothetical scenario where the dataset comprises only five such samples, the computation of JGA would involve a binary conversion of 'Matched' statuses, with 'True' being equated to 1 and 'False' to 0. Subsequently, the average JGA for these samples would be determined by the formula:

$$JGA : \frac{(1+1+0+0+1)}{5} = 0.6$$

This is attributed to the fact that even a minor error in the model's prediction can disproportionately affect the overall evaluation, rendering all correct predictions nullified in the face of a single mistake. This underscores the need for a nuanced understanding of the metric's implications and the potential for exploring alternative measures that may offer a more forgiving assessment of a model's predictive capabilities.

6.2 Slot Accuracy

Slot Accuracy (SA) is an alternative metric to Joint Goal Accuracy (JGA) that evaluates the performance of dialogue systems by individually comparing each predicted domain-slot-value triplet against its corresponding ground-truth label. This metric is referenced in the work of Wu et al. (2019) [?]. They define S as the set of unique domain-slot pairs present within the dataset, GT represent the set of ground-truth belief states, and PR denote the set of predicted belief states at any given turn t. The formula for calculating slot accuracy at turn t is expressed as follows:

$$SA = \frac{|S| - |X| - |Y|}{|S|}$$

In this equation, X is defined as the set difference GT \ PR, and Y as PR \ GT. The terms |X| and |Y| quantify the number of false negatives and false positives, respectively. The set X encompasses the domain-slot-value triplets that are present in the ground-truth belief states but are conspicuously absent in the predicted belief states. This discrepancy indicates that the model has failed to predict these specific triplets, which is a critical oversight in terms of accuracy. Conversely, the set Y comprises domain-slot-value triplets that are found within the predicted belief states but do not have a corresponding presence in the ground-truth belief states. This suggests that the model has either incorrectly predicted a slot that does not exist or has erroneously assigned a value to a slot, thereby deviating from the ground truth.

For example, in the MultiWOZ dataset, the total number of unique domain-slot pairs $|S|$ is 30. We assume that the model is missing two domain triplets $|GT \setminus PR| = |X| = 2$ and $|GT \setminus PR| = |Y| = 0$, the slot accuracy would be calculated as $(30-2-0) / 30$, resulting in a value of 93.33%.

However, the metric of slot accuracy can sometimes present a misleadingly optimistic view of a dialogue system's performance. This illustrates how slot accuracy can overestimate the performance of a Dialogue State Tracking (DST) system. To further demonstrate this point, let us consider a scenario where no predictions are made for any turn, i.e., PR = 0 for all t. Under these circumstances, the formula for slot accuracy simplifies to $(|S|-|GT|) / |S|$. Typically, $|GT|$ is significantly smaller than $|S|$ because only a limited number of domain-slot pairs are active at any given time within a conversation. Consequently, even with no predictions made, the slot accuracy metric would remain relatively high. This example underscores the potential for slot accuracy to provide an inflated assessment of DST performance, especially in datasets with a larger number of domain-slot pairs, slot accuracy is a poor metric to evaluate DST.

6.3 F1 Score

The F1 Score is indeed a crucial metric for evaluating the predictive performance of a model, particularly in the context of classification tasks. It is valued for its ability to harmonize the precision and recall of a model, providing a single measure that balances both the false positives and false negatives. To elaborate, the F1 Score is the harmonic mean of precision and recall, where precision is the ratio of true positive predictions to the total predicted positives, and recall is the ratio of true positive predictions to the actual positives. The formula for the F1 Score is given by:

$$F1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

In the scenario of dialogue systems, where the type of current turn (ODD or TOD) might be classified, the F1 Score can be particularly insightful. It allows us to ascertain if there is any confusion between these classes by providing a measure that considers both the instances where the model correctly identifies a class (precision) and the instances where the model correctly identifies all relevant samples of a class (recall). By computing the F1 Score for each class, we can gain a nuanced understanding of the model's ability to distinguish between ODD and TOD turns. This metric, therefore, serves as a valuable tool for developers and researchers to evaluate and improve the performance of dialogue systems.

7. RESULTS AND DISCUSSION

7.1 Performance on FusedChat

System	# Params.	SA	JGA	F1	
				TOD	OOD
Two-in-one model	-	97.2	59.2	-	-
Classification-based model	-	97.3	60.0	-	-
GradTOD (ours)	*250M*	*99.6*	*82.5*	*97.2*	*95.4*

Table 4. *Experimental results on the FUSEDCHAT test set with Join Goal Accuracy (JGA), Slot Accuracy (SA), F1-score performance. Two models from FUSEDCHAT [23] are cited to compare for JGA and SA metrics, their parameters are not referred to in the original paper so that we hide it. Addition F1 column is reported to ensure proper tracking of the dialogue's type (ODD or TOD). Our GradTOD model's performance is written in italics.*

The performance of our model on FusedChat is comprehensively demonstrated in Table 4. Our model exhibits exceptional performance, achieving an F1 score above 95% for each type of dialogue. This indicates that the model is proficient in predicting the state for Task-Oriented Dialogues (TODs) and retaining the state when responding to chitchat. In addition to this, we have referenced two models from FusedChat [23] - the two-in-one and classification-based models - to compare the Joint Goal Accuracy (JGA) metric. As per the data presented in Table 3, these two models achieve a JGA score of approximately 59% to 60%. However, our GradTOD model significantly enhances performance by approximately 23%. We attribute this superior performance to the instruction-finetuned ability of the Flan-T5 model and the method proposed in Section 4. These elements enable the model to learn what information needs to be included: whether it's the classification of TOD or OpenDomain Dialogue (ODD), the action of the current user's text, or the state of the current user's text.

Furthermore, since Flan-T5 is pre-trained for the Chain of Thought task, this model will perform three tasks in order and ensure coherence between the three components. More specifically, the type of dialogue will be classified first. Then, determining the action of the second task will link the results of the first task to generate an appropriate action, which performs the suitable intent of user text. Once the results of the second task are available, our model will track which slot values are represented for the user's state based on the user's actions.

For instance, a user action that informs a specific slot in the second task will require the necessary value and update that slot in the user state in the third task. Other actions such as request, thank, bye, etc. will not update the user state. From these reasons, we confirm that using Flan T5 is the most optimal choice at present to achieve such outstanding results in both JGA and Slot Accuracy (SA) metrics, 82.5% and 99.6% respectively. This underscores the efficacy of our model in handling both TODs and ODDs, thereby providing a robust solution for dialogue systems.

7.2 Performance on SGD

Backbone	**System**	**# Params.**	**Seen**	**Unseen**	**Avg**
Encoder	SGD baseline	110M	-	-	25.4
	SGP-DST	660M	-	-	72.2
	multi-task BERT	110M	-	-	82.7
Seq2Seq	D3ST_BASE	220M	92.5	66.4	72.9
	D3ST$_{LARGE}$	770M	93.8	75.4	80.0
	D3ST$_{XXL}$	**11B**	**95.8**	**83.3**	**86.4**
	SDT$_{BASE}$	250M	-	-	76.3
	SDT$_{LARGE}$	800M	-	-	83.2
	SDT$_{XXL}$	**11B**	**95.8**	**86.4**	**88.8**
	AnyTOD$_{BASE}$	220M	89.9	62.4	76.2
	AnyTOD$_{XXL}$	**11B**	**94.8**	**82.2**	**88.5**
Seq2Seq	*GradTOD (ours)*	*250M*	*83.3*	*89.2*	*88.6*

Table 5. *Experimental results on the SGD test set with Join Goal Accuracy (JGA) performance on seen and unseen domains, the value with Large Language Model (params more than 1B) and our GradTOD model written in bold and in italics, respectively.*

The performance of various models on the SGD dataset is comprehensively presented in Table 5. In addition to evaluating Joint Goal Accuracy (JGA) across all domains, we have also recorded results on seen and unseen domains to provide a more generalized view. The models used for evaluation on the SGD test set include the baseline from SGD [26], SGP-DST [27], multi-task BERT [15], D3ST [5], SDT [4], and AT from AnyTOD [6]. As per the data in Table 4, recent encoder models such as SGP-DST and multi-task BERT have achieved commendable Average JGA scores of above 72.2% and 82.7% respectively. However, the SGD baseline model, despite its innovative encoder backbone as in the SGD+ dataset [1], has only achieved a score of 25.4%.

In contrast, Seq2Seq models have demonstrated superior performance, achieving an approximate score of 89%. This can be attributed to the scaling of the recent model transformer's size and the exploration of the inherent capabilities of Seq2Seq models. When considering the seen domain, our GradTOD model has experienced a setback compared to previous models, achieving only an 83.3% score on JGA metrics. However, this model has performed exceptionally well in unseen domains, achieving an approximate score of 89.2%. This experiment has met our expectations in demonstrating the model's ability for domain adaptation, also known as zero shot ability. With our continued development, the GradTOD model has achieved an average score of 88.6%, validating our claim about the effectiveness of leveraging the instruction-finetuning technique and the proposed method. This underscores the accuracy of our model in predicting dialogue states and its potential in enhancing the performance of dialogue systems.

7.3 Performance on MultiWoz 2.4

Backbone	System	# Params.	SA	JGA
Encoder	SOM-DST$_{BASE}$	110M	98.84	75.19
	AUX-DST	220M	99.07	78.14
	Meta AUX-DST	**220M**	**99.08**	**78.57**
Seq2Seq	D3ST$_{BASE}$	220M	-	72.1
	D3ST$_{LARGE}$	770M	-	70.8
	D3ST$_{XXL}$	**11B**	-	**75.9**
Seq2Seq	*GradTOD (ours)*	*250M*	*99.03*	*76.9*

Table 6. *Comparison of performance between state-of-the-art research on MultiWoz 2.4 test set. The result of SOM-DST on MultiWoz 2.4 is referred to on [34]. The highest score with Encoder only and Large Language Model (Seq2Seq) are written in bold while our GradTOD model is written in italics, respectively.*

The performance metrics of various models on the MultiWoz 2.4 dataset are meticulously documented in Table 6. The models under consideration for this comparative analysis include SOM-DST [32], AUX-DST & meta AUX-DST from metaASSIST [33], D3ST [5], and AnyTOD [6]. These models are instrumental in evaluating the zero-shot ability during domain transfers.

A careful observation of Table 5 reveals that encoder models exhibit a commendable proficiency in predicting the current state. Interestingly, these models even surpass Seq2Seq models in terms of Joint Goal Accuracy (JGA) scores. Encoder models boast a JGA score ranging from approximately 75% to 79%, while Seq2Seq models' scores fall within the range of 72% to 76%.

This disparity in performance can be attributed to the existing optimized techniques that are inherent in encoder models, such as slot carryover [15] and operation [32]. These techniques effectively address the underestimation of JGA metrics. On the other hand, encoder-decoder models rely solely on their intrinsic capabilities to predict the user's state. In some cases, this results in miscellaneous values during the evaluation of system turns.

In contrast, our model, which incorporates the proposed method, demonstrates some improvement when evaluated on the test set for JGA and Slot Accuracy (SA), with scores of 76.9% and 99.03% respectively. This underscores the efficacy of our proposed method in enhancing the performance of the model.

8. CONCLUSIONS AND PERSPECTIVES

In this scholarly article, we aim to demonstrate the immense potential of amalgamating prompt engineering and conditional generation to proficiently manage Dialogue State Tracking. Through the successful implementation of our system, GradBot, we have been able to architect new business logic with a minimal custom ontology. This capability empowers us to identify policy tasks or domains for unseen applications with a relative ease. Our evaluation experience, which was conducted on three benchmark datasets, has yielded promising results. We are optimistic that this study will make a significant contribution to the field of research and development, stimulating interest and fostering innovation among scholars and practitioners alike. However, it is important to acknowledge that our research is not devoid of limitations. One such limitation pertains to the scalability and complexity of context length history, especially when dealing with thousands of slot values across multiple domains. To address this challenge, we draw inspiration from the Graph-of-thoughts (GoT) model. This model effectively reduces the context length by utilizing a graph to represent an ontology. Simultaneously, we aim to address the weaknesses observed in the transition between Task-oriented Dialogue (ToD) and Open Domain Dialogue (ODD). This includes transitions between casual conversation (chitchat) and task-oriented guides within the same domain (inter-mode), as well as transitions between different domains (outer-mode). Our ultimate goal is to enhance the model's ability to generate human-like conversations that are both coherent and contextually relevant.

In conclusion, while our research does present certain challenges, it also opens up exciting avenues for further exploration and improvement in the field of dialogue state tracking. We believe that our findings and methodologies will provide a solid foundation for future research in this area, ultimately leading to more sophisticated and effective dialogue systems.

CONFLICTS OF INTEREST: No conflict of interest.

9. REFERENCES

1. Vahid Noroozi, Yang Zhang, Evelina Bakhturina, and Tomasz Kornuta. A fast and robust bert-based dialogue state tracker for schema-guided dialogue dataset. arXiv preprint arXiv:2008.12335, 2020.

2. Yushi Hu, Chia-Hsuan Lee, Tianbao Xie, Tao Yu, Noah A Smith, and Mari Ostendorf. Incontext learning for few-shot dialogue state tracking. arXiv preprint arXiv:2203.08568, 2022.

3. Chun-Mao Lai, Ming-Hao Hsu, Chao-Wei Huang, and Yun-Nung Chen. Controllable user dialogue act augmentation for dialogue state tracking. arXiv preprint arXiv:2207.12757, 2022.

4. Raghav Gupta, Harrison Lee, Jeffrey Zhao, Yuan Cao, Abhinav Rastogi, and Yonghui Wu. Show, don’t tell: Demonstrations outperform descriptions for schema-guided task-oriented dialogue. In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics, 2022.

5. Jeffrey Zhao, Raghav Gupta, Yuan Cao, Dian Yu, Mingqiu Wang, Harrison Lee, Abhinav Rastogi, Izhak Shafran, and Yonghui Wu. Description-driven task-oriented dialog modeling. arXiv preprint arXiv:2201.08904, 2022.

6. Jeffrey Zhao, Yuan Cao, Raghav Gupta, Harrison Lee, Abhinav Rastogi, Mingqiu Wang, Hagen Soltau, Izhak Shafran, and Yonghui Wu. Anytod: A programmable task-oriented dialog system. ArXiv, abs/2212.09939, 2022.

7. Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. arXiv preprint arXiv:1910.13461, 2019.

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

9. Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. The Journal of Machine Learning Research, 21(1):5485–5551, 2020.

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

11. Kang Min Yoo, Hanbit Lee, Franck Dernoncourt, Trung Bui, Walter Chang, and Sang-goo Lee. Variational hierarchical dialog autoencoder for dialog state tracking data augmentation. arXiv preprint arXiv:2001.08604, 2020.

12. Darsh J Shah, Raghav Gupta, Amir A Fayazi, and Dilek Hakkani-Tur. Robust zero-shot crossdomain slot filling with example values. arXiv preprint arXiv:1906.06870, 2019.

13. Brendan King and Jeffrey Flanigan. Diverse retrieval-augmented in-context learning for dialogue state tracking. arXiv preprint arXiv:2307.01453, 2023.

14. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

15. Eleftherios Kapelonis, Efthymios Georgiou, and Alexandros Potamianos. A multi-task bert model for schema-guided dialogue state tracking. arXiv preprint arXiv:2207.00828, 2022.

16. Zhi Chen, Lu Chen, Bei Chen, Libo Qin, Yuncong Liu, Su Zhu, Jian-Guang Lou, and Kai Yu. Unidu: Towards a unified generative dialogue understanding framework. arXiv preprint arXiv:2204.04637, 2022.

17. Ruolin Su, Jingfeng Yang, Ting-Wei Wu, and Biing-Hwang Juang. Choice fusion as knowledge for zero-shot dialogue state tracking. In ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 1–5. IEEE, 2023.

18. Zhaojiang Lin, Bing Liu, Andrea Madotto, Seungwhan Moon, Paul Crook, Zhenpeng Zhou, Zhiguang Wang, Zhou Yu, Eunjoon Cho, Rajen Subba, et al. Zero-shot dialogue state tracking via cross-task transfer. arXiv preprint arXiv:2109.04655, 2021.

19. M. Moradshahi, Sina J. Semnani, and Monica S. Lam. Zero and few-shot localization of task-oriented dialogue agents with a distilled representation. In Conference of the European Chapter of the Association for Computational Linguistics, 2023.

20. Rahul Goel, Waleed Ammar, Aditya Gupta, Siddharth Vashishtha, Motoki Sano, Faiz Surani, Max Chang, HyunJeong Choe, David Greene, Kyle He, et al. Presto: A multilingual dataset for parsing realistic task-oriented dialogs. arXiv preprint arXiv:2303.08954, 2023.

21. Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. ArXiv, 2020.

22. Kai Sun, Seungwhan Moon, Paul A. Crook, Stephen Roller, Becka Silvert, Bing Liu, Zhiguang Wang, Honglei Liu, Eunjoon Cho, and Claire Cardie. Adding chit-chat to enhance taskoriented dialogues. In North American Chapter of the Association for Computational Linguistics, 2020.

23. Tom Young, Frank Xing, Vlad Pandelea, Jinjie Ni, and E. Cambria. Fusing task-oriented and open-domain dialogues in conversational agents. In AAAI Conference on Artificial Intelligence, 2021.

24. Zhiyu Chen, Bing Liu, Seungwhan Moon, Chinnadhurai Sankar, Paul A. Crook, and William Yang Wang. Ketod: Knowledge-enriched task-oriented dialogue. ArXiv, abs/2205.05589, 2022.

25. Hyung Won Chung, Le Hou, S. Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Wei Yu, Vincent Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed Huai hsin Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models. ArXiv, abs/2210.11416, 2022.

26. Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In AAAI Conference on Artificial Intelligence, 2019.

27. Yu-Ping Ruan, Zhenhua Ling, Jia-Chen Gu, and QUAN LIU. Fine-tuning bert for schemaguided zero-shot dialogue state tracking. ArXiv, abs/2002.00181, 2020.

28. Zhaojiang Lin, Bing Liu, Seungwhan Moon, Paul Crook, Zhenpeng Zhou, Zhiguang Wang, Zhou Yu, Andrea Madotto, Eunjoon Cho, and Rajen Subba. Leveraging slot descriptions for zero-shot cross-domain dialogue state tracking. arXiv preprint arXiv:2105.04222, 2021.

29. Chia-Hsuan Lee, Hao Cheng, and Mari Ostendorf. Dialogue state tracking with a language model using schema-driven prompting. arXiv preprint arXiv:2109.07506, 2021.

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

31. Fanghua Ye, Jarana Manotumruksa, and Emine Yilmaz. Multiwoz 2.4: A multi-domain taskoriented dialogue dataset with essential annotation corrections to improve state tracking evaluation. In SIGDIAL Conferences, 2021.

32. Sungdong Kim, Sohee Yang, Gyuwan Kim, and Sang-Woo Lee. Efficient dialogue state tracking by selectively overwriting memory. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, July 2020. Association for Computational Linguistics.

33. Fanghua Ye, Xi Wang, Jie Huang, Shenghui Li, Sam Stern, and Emine Yilmaz. Metaassist: Robust dialogue state tracking with meta learning. In Conference on Empirical Methods in Natural Language Processing, 2022.

34. Fanghua Ye, Yue Feng, and Emine Yilmaz. ASSIST: Towards label noise-robust dialogue state tracking. In Findings of the Association for Computational Linguistics: ACL 2022, pages 2719–2731, Dublin, Ireland, May 2022. Association for Computational Linguistics

10. APPENDIX

The Fusedchat dataset is renowned for its numerous outstanding features, which greatly enhance its usability and potential applications. We use the metadata tag in the next system utterance (ut+1) to create the current state and the dialog act tag in the current user utterance (ut) tag to create the current action. However, there are some inconsistencies in the slots and values between these two tags. While these bugs may present challenges in terms of data processing, we have provided solutions for improvement and refinement. By addressing these issues, our goal is to further enhance the quality and reliability of the Fusedchat dataset, making it an even more valuable resource for various research and development endeavors.

Error Type	**Metadata miss slots**
Dialog's id	MUL1598
Log user (t)	1
Current user (t)	I'm looking for an **expensive** restaurant in the north area
Dialog_act (t)	[["area", "north"],["pricerange", "expensive"]]
Metadata (t+1)	{"area": "north"}

Table 7. *Example of Metadata miss slots*

Metadata miss slots: one of the bugs that we encountered in the Fusedchat dataset is related to the metadata tags. Specifically, there are some missing slots within the metadata tag, while the dialog act tag still has the action information for those slots. This inconsistency between the two tags can potentially lead to confusion and inaccuracies during data processing. For example, in dialog MUL1598 in the dialog act tag in the first log: "Restaurant-Inform": [["area", "north"], ["price range", "expensive"]], and the metadata tag in the second log is only each slot "area": "north". We resolve this issue by implementing a conditional check within the dialog act tag. If we identify a slot linked to inform action, but that slot is not specified in the metadata tag, we will include it in the metadata tag.

Error Type	Dialog_act miss slots
Dialog's id	MUL1598
Log user (t)	5
Current user (t)	Lets try Hakka. It will be just **myself** at 16:45 on tuesday.
Dialog_act (t)	[["name", "hakka"], ["time", "16:45"], ["day", "tuesday"]]
Metadata (t+1)	{"time": "16:45", **"people": "1"**, "day": "tuesday", "pricerange": "expensive", "name": "hakka", "area": "north"}

Table 8. *Example of Dialog act miss slots*

Dialog act miss slots: this bug will be the opposite of the first error. The user performs an action to provide information, but dialog act is not recorded with this slot, even though this slot is recorded in the metadata tag. For example, in dialog MUL1598 in the dialog act tag the 5th log is missing the information "people": "1", but this pair of slot values is present in the metadata tag at the 6th log. We fixed it by checking for extra slots in the metadata tag, if any slot was missing in the dialog act tag then we added an inform action for that slot in the dialog act tag

Error Type	Metadata and Dialog_act miss slots
Dialog's id	MUL0145
Log user (t)	9
Current user (t)	I would like to book a table for **three** on **Friday**, at **16:00**.
Dialog_act (t)	[]
Metadata (t+1)	{"people": " ", "day": " ", "time": " ", "name": "not mentioned", "pricerange": "moderate", "food": "european", "area": "centre"}

Table 9. *Example of Metadata and dialog act miss slots*

Metadata and dialog act miss slots: the third bug is the most difficult for us. The user performs the action informing the slots, but these slots are absent from both the metadata tag and the dialog act tag. For example, in dialog MUL0145 in the dialog act tag the 9th log and the metadata tag at the 10th log also do not record the information of "for three", "Friday", and ”16:00”. We have come up with many solutions but they are all ineffective, so we decided to filter data like this to not have a negative impact on the training results.

Error Type	**Dialog act miss slots and values**	
Dialog’s id	PMUL4645	MUL2285
Log user (t)	11	13
Current user (t)	I also need a taxi.	I don’t need to book.
Dialog_act (t)	[[”none”, ”none”]]	[]

Table 10. *Example of Dialog act miss slots and values*

Dialog act miss slots and values: there are some elements in dialog act that, although the action is informed, do not contain any slots or values, or both slot and value are none. This is obviously meaningless so we decided to eliminate these samples during training. For example, dialog PMUL4645, in the dialog act tag at 11th log: ”Taxi-Inform”: [[”none”, ”none”]]. Or dialog MUL2285, at 13th log: ”Train-Inform”: []

Error Type	**Inconsistent values**
Dialog’s id	PMUL1554
Log user (t)	7
Current user (t)	uh why yes you are very perceptive depart cambridge arrive by **9:45** leaves on sat going to ely
Dialog_act (t)	[[**”arriveby”, ”9:45”**], [”departure”, ”cambridge”] ,[”destination”, ”ely”]]
Metadata (t+1)	{”leaveAt”: ”not mentioned”, ”destination”: ”ely”, ”day”: ”saturday”, **”arriveBy”: ”09:45”**, ”departure”: ”cambridge”

Table 11. *Example of Inconsistent values*

Inconsistent values: there is another small bug that is the inconsistency of values in the whole data set such as [dontcare, dont care, don’t care] will be standardized to dontcare. The time values are not consistent between the metadata tag and the dialog act tag. For example, in dialog PMUL1554 in the 7th log, the time in the text is 9:45, but the value of the slot arrive in the metadata tag is 09:45 and the value in the dialog act tag is 9:45. These value inconsistency errors are not too serious, but they will negatively affect training because there are no common standards. And more importantly, the JGA metric result will be low, because if the label is don't care, and the prediction is don't care, then JGA will be 0, this is similar to 9:45 and 09:45.

Link of out paper about GradDST - is called GradTOD (Dialogue State Tracking model):

GradTOD - A Unified Dialogue State Tracking Model for Task-Oriented and Open Domain Dialogues | IEEE Conference Publication | IEEE Xplore

GradTOD - A Unified Dialogue State Tracking Model for Task-Oriented and Open Domain Dialogues

1st Truc Nguyen Liem
Department of Information Technology
FPT University
Ho Chi Minh City, Vietnam
Email: trucnlse161876@fpt.edu.vn

2nd Sinh Nguyen Cao Hoai *
Department of Information Technology
FPT Unviersity
Ho Chi Minh City, Vietnam
Email: sinhnchse151023@fpt.edu.vn

3rd Hung Nguyen Quoc *
Department of Information Technology
FPT Unviersity
Ho Chi Minh City, Vietnam
Email: hungnqse162007@fpt.edu.vn

4th Tien Nguyen Van
Pythera AI
Faculty of Information Technology
Open University
Ho Chi Minh City, VietNam
Email: tien.nguyen@gradients.host

5th Hieu Pham Trung
Pythera AI
Faculty of Information Technology
Open University
Ho Chi Minh City, Vietnam
Email: hieu.pham@gradients.host
Email: hieu.pt@ou.edu.vn

6th Vinh Truong Hoang
Faculty of Information Technology
Open University
Ho Chi Minh City, Vietnam
Email: vinh.th@ou.edu.vn

***Abstract*—The task-oriented dialogue domain system requires classifying intent and replying to a specific goal domain. In the sub-module of Task-oriented, the Dialogue State Tracker (DST) is well-known as a variety processing tracker. However, existing DST models often specialize in only task-oriented domains (ToD), leading to limited performance when applied to scenarios. In this paper, we propose GradTOD, a unified DST model that predicts both two task types, task-oriented dialogue (TOD) and open-domain dialogue (ODD). Our model leverages the recent advances in prompt engineering and conditional generation to perform zero-shot learning. After experiments, GradTOD has achieved an 88.6% and 82.5% score on Joint Goal Accuracy metrics when evaluating the Scheme-Guided Dialogue (SGD) and FusedChat test sets correspondingly, demonstrating the adaption ability for multi-domains.**

I. Introduction

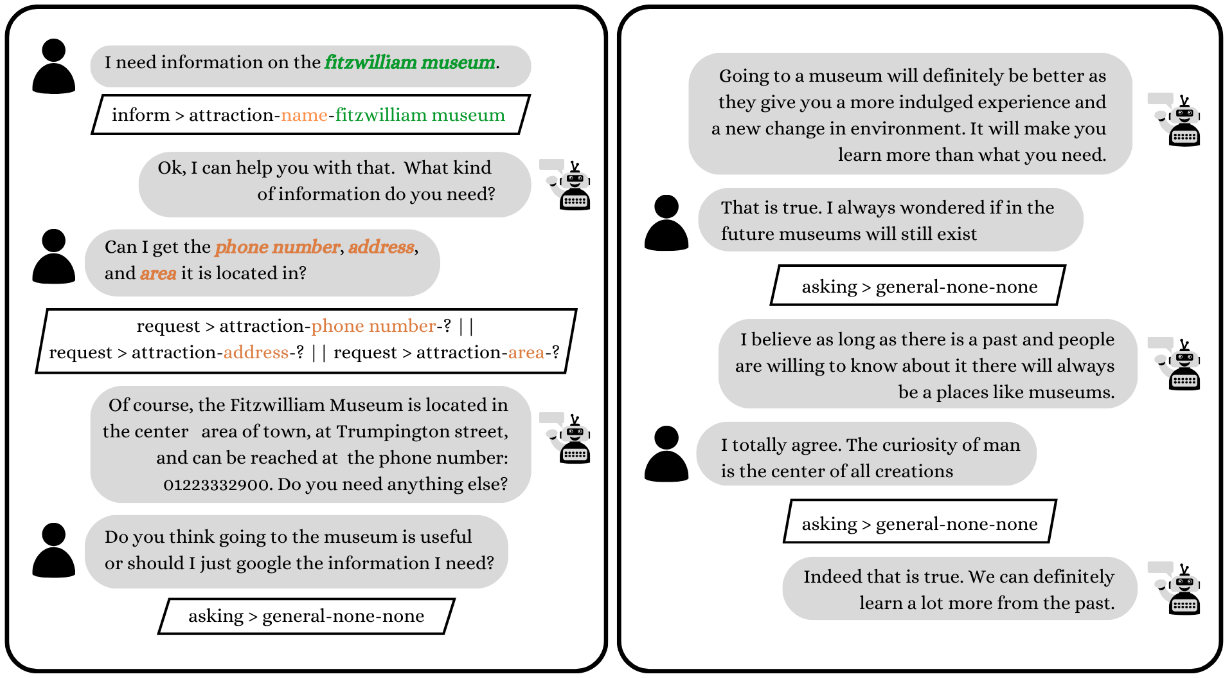
The task-oriented domain has attracted a lot of attention not only in academics but also in industry. This objective is to achieve specific strategies, such as providing information or performing an action that satisfies the user's request. One of the crucial components of the task-oriented domain is Dialogue State Tracking (DST), which tries to predict appropriate actions to resolve the goals. At every turn, DST has to look up the dialogue history (whole or sliding window) to the current turn to determine the specific slot values in the slot list [1], [2]. In our observation, there are two kinds of Dialogue State Tracking designed:

1) The traditional method uses an Encoder module exploiting multihead layers to build classified data intent prediction, slots prediction, and slot filling [1], [3];
2) Seq2Seq module uses prompting to show semantics between turns and ontology through conversation to predict a required value [4]–[6].

In industrial applications, DST is required to adapt flexibly new domains (services) without prior training for a specific task. For this purpose, the role of zero-shot prediction on unseen domains becomes indispensable in Dialogue State. Some previous work [4]–[6] uses guided schema as a description to show the semantics of schema element with input sentence. With recent advances in pre-trained language models [7]–[9], augmented language techniques are gaining more and more attention. These methods have demonstrated impressive improvement and zero-shot adaptability [3], [10], [11]. Moreover, the in-context learning framework (ICL) shows efficiency methods and techniques in DST without the re-training stage by combining prompting and examples for a task [12], [13].

More specifically, Figure 1 shows an example conversation with the associated dialog state of the attraction domain. The user wants to find information about a specified name and request more data about the phone number, address, and area. Simultaneously, they share their claim about why they need a museum when having internet. The system must answer questions based on their knowledge (close/open question-answering domain) or even daily conversation (chitchat). It requires an intelligent chatbot with novel architectures and approaches such as dialogue state tracking, switching domain classification, and supporting generative AI-specific knowledge to improve the performance of the conversational agent. However, there still haves a noticeable gap until now between existing benchmark datasets and real-life human conversations.

[*] Equal contribution



I need information on the fitzwilliam museum.
inform > attraction-name-fitzwilliam museum
Ok, I can help you with that. What kind of information do you need?
Can I get the phone number, address, and area it is located in?
request > attraction-phone number-? | | request > attraction-address-? | | request > attraction-area-?
Of course, the Fitzwilliam Museum is located in the center area of town, at Trumpington street, and can be reached at the phone number: 01223332900. Do you need anything else?
Do you think going to the museum is useful or should I just google the information I need?
asking > general-none-none
Going to a museum will definitely be better as they give you a more indulged experience and a new change in environment. It will make you learn more than what you need.
That is true. I always wondered if in the future museums will still exist
asking > general-none-none
I believe as long as there is a past and people are willing to know about it there will always be a places like museums.
I totally agree. The curiosity of man is the center of all creations
asking > general-none-none
Indeed that is true. We can definitely learn a lot more from the past.

Fig. 1. An example of Attraction Domain on Fusedchat datasets. The conversation builds from MutiWoz2.4 by rewriting the existing Task-oriented domain turns and adding new Open Dialogue domain turns.

These datasets cover a limited number of domains, unrealistic constraints focus on a few skill sets and do not have empathy or persona consistency, etc.

Motivated by this research, we propose a recent advance in prompt engineering and conditional generation to adapt zero-shot learning applications useful in the business domain. The effectiveness of our proposed method is experimented on the FusedChat, SGD, and MultiWoz2.4 datasets, achieving a remarkable performance on automatic and human evaluations. Our proposed methods can be summarized as follows:

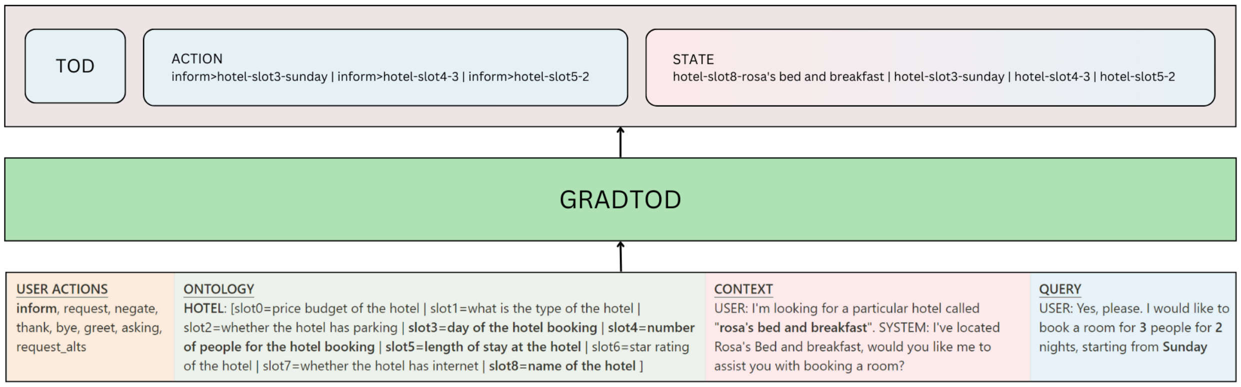
- We introduce a simple method but effective controls tracking conversation flow and easily expand the new business domains (services). Our evidence shows that, under this very general setting, natural language descriptions lead to better quality over abbreviated notations supporting Task-oriented Domain and Open Dialogue Domain.
- While several researchers/developers focus on using the Large Language Model to give a strong performance experience. We are interested in exploring the potential of small language modeling on a variety of tasks in Dialogue State Tracker based on a contextual semantics ontology.

The remainder of the article is structured as follows. Section II discusses about our relevant works. The key idea for the Dialog State Tracker combines prompting and conditional generation with ontology performing the details is explained in Section III. The outcome and the work's conclusion are then reported in Sections IV to V, respectively.

II. Related Works

Dialog State Tracking: This is the core of building a conversational system. All these methods can be divided into two categories: Classification (Encoder) and Generation (Seq2Seq). Recently, transformer-based pre-trained models, such as BERT [14] have achieved remarkable results in a range of natural language processing tasks. Thus, a multi-task BERT-based model [15] is proposed to solve intent prediction, slot filling and request slot filling by encoding the history and service schema. These approaches, however, are not applicable to unseen values and not able to scale up large domains. To address this issue, a UniDU framework is introduced in [16] that achieves effective information exchange across diverse dialogue-understanding tasks. His study found the intuitive multitask mixture training method which making the unified model bias convergence to more complex tasks.

Enhance Reading Comprehension: Unlike the above research, several researchers found that generation extractive (Machine Reading for Question Answering) works well in answering textual QA tasks by the ability to read the context. Taking this advantage, CoFunDST [17] combined Dialogue State Tracking with Machine Reading Comprehension, which applied to context-choice fusion as an extensive knowledge to predict slots and values among available candidates to improve a zero-shot performance. Another experiment of this comprehension task, TransferQA [18] introduces constructing negative question sampling and context truncation, two effective methods that handle "none" value slots and push the model better on generalization ability in unseen domains.



inform>hotel-slot3-sunday | inform>hotel-slot4-3 | inform>hotel-slot5-2

hotel-slot8-rosa's bed and breakfast | hotel-slot3-sunday | hotel-slot4-3 | hotel-slot5-2

inform, request, negate, thank, bye, greet, asking, request_alts

HOTEL: [slot0=price budget of the hotel | slot1=what is the type of the hotel | slot2=whether the hotel has parking | **slot3=day of the hotel booking** | **slot4=number of people for the hotel booking** | **slot5=length of stay at the hotel** | slot6=star rating of the hotel | slot7=whether the hotel has internet | **slot8=name of the hotel**]

USER: I'm looking for a particular hotel called "**rosa's bed and breakfast**". SYSTEM: I've located Rosa's Bed and breakfast, would you like me to assist you with booking a room?

USER: Yes, please. I would like to book a room for **3** people for **2** nights, starting from **Sunday**

Fig. 2. Overview of GradTod approach for schema-guided multi-domains dialogues. The bottom figure includes specific examples for dialogue context, user action, ontology and current query while the top figure stimulates predictions. In our observation, the state and current action labels in FusedChat have different slot values (details in Appendices A). Thus, we combine prediction all in one sample to make the same evidence between ***state*** and ***current action*** simultaneously enhance the construct meaning of the input and target value.

Simultaneously improving model abilities, Moradshahi's approach [19] stated that collecting large amounts of data for every dialogue domain is often costly and inefficient. Thus, his study applies the transfer learning technique, which only uses a limited task-oriented subset in source data language to build a high-quality model for other target ones. The experiments achieved unexpected results when training only with 10% training data points but increasing 10% performance to the previous SOTA research on both zero-shot and few-shot learning. On multilingual application, PRESTO [20], a public multilingual conversation dataset for real-world NLU tasks, and the applied-based mT5 model are considered as baseline training of this field. The experiments show this module performs well on various linguistic phenomena.

Effection dataset: Recent state-of-the-art researchs [21]–[23] improved existing ToD datasets by designing different methods such as enhancing context, samples, and method processing to make that real human-level conversation. FusedChat [22] rewrote Task-oriented dialogue and added new open domain dialogue (chitchat) into one benchmark. ACCENTOR [21] proposed a data augmentation method for generating conversation leveraging pre-trained generative models and a custom filter to minimize human annotation effort. Based on these approaches, we analyze our model training on FusedChat and SGD datasets, then evaluating single and multi-domain dialogue. This approach allows both TOD and ODD adapting to business domain.

III. Methodology

Following previous DST research, we leverage the instruction-finetuned ability of Flan-T5 [24] as our baseline and combine with it the technique of elements on multi-task [25] to define input. We propose the template training with instruction, context, ontology, and a list of user actions. After that, the model must choose which sensible representation to learn by reading and extracting the information input.

A. Complexity of building Schema-guided Definition

GradTod requires building an ontology (shema-guided) from a business domain (single or multi-domains). This ontology helps the DST to understand its meaning and facilitates predicting the user state. For example, in an agent-supporting hotel reservations chatbot, the designer defines the parameters to track along with description as [*name*=*name of the hotel,* *star*=*star rating of the accommodation,* *number_of_rooms*=*number of rooms in the reservation,* *check_in_date*=*start date for the reservation,...*].

Then, at each turn, the purpose of a DST module is to use the dialogue history up to that turn to predict a dialogue state, which represents the user's goal and progress in using the system. In a similar vein to prior research [4], [5] each element in this schema is characterized by a natural language description, and entire the dialogue flow is designed as a conversation graph flow.

B. Predefine structure model

Schema-guided representation As D3ST [5], ontology will be a dictionary that interprets the descriptions of the slots that have been mapped into ordinal slots. Converting slots into numeric slots makes it easier for the model to perform tasks for unseen domains because it relies on descriptions without understanding the names of the slots in those domains. An ontology contains all the domains in the dialogue, the model must learn to choose domains that match the current turn. Besides, the ontology is added to the domain name at the beginning of each dictionary chain, this will make it easier for the model to identify the domains at the current turn. The overview representation is $O = \{s_0^{D_0} = d_0^{D_0}, s_1^{D_0} = d_1^{D_0}, ..., s_N^{D_0} = d_N^{D_0}, ..., s_N^{D_M} = d_N^{D_M}\}$, where M, N to be a number of domains and slots of each domain, respectively.

Dialogue context representations Following the previous research [26], [27], we set each user's utterance to be the current query sample and remaining the history as context dialogue for each utterance we append prefix "user" and "system" tokens, signaling the speaker of each turn. In the end, full dialogue context is created by concatenating all the system and user utterances $C = \{u_0, s_0, ..., u_{t-1}, s_{t-1}\}$, where t=0, 1, 2,.. to be a current query's position in sub dialogue sample.

Action enhances constraint Finally, we construct the list of user actions that will be inserted into the input template for each turn. Because our label contains current action, a list of user actions will support the model in choosing the most appropriate action, reducing the probability of predicting an action that does not exist in raw datasets. Formula as $A = \{inform, request, ..., thank, general\_asking\}$

With the advantage of checkpoint Flan-T5 being instruction-based training, we combine the Machine Reading Comprehension with instructions to give detailed explanations of tasks so that the model clearly understands what tasks it needs to do and how to perform them. In the instructions, we format the detailed content input by inserting these above context, query, ontology and user actions. Format is presented in Figure 2.

C. DST as guided Reading Conversation

GradTOD is trained on three tasks at the same time:

1) type-classify TOD or ODD.
2) current_action-predict user actions for each turn.
3) current_state-predict state for each turn.

Each dialogue of the used chat dataset contains chitchat utterances, so in addition to defining state like other models, we add a task to identify the type (TOD/ODD) to evaluate the ability to classify between general conversation and task conversation. Furthermore, the current action is added to the label to create a tight binding between the user's action and the user's state. The model will learn that if the user's action is informed, the state will be updated, but actions such as request, thank, bye,... will have to maintain the same state as the previous turn.

IV. Experiment and Results

A. Datasets

Fusedchat Dataset [22] is an essential dataset created by merging task-oriented dialogue (TOD) and open-domain dialogue (ODD). This amalgamation of ODD and TOD fosters a seamless connection and robust contextual interplay between the two dialogue modes. Hence, this dataset can support the backbone system to accelerate comprehension when dealing with reality. As an extension of the renowned MultiWoz dataset [28], Fusedchat integrates additional ODD turns either before or after the existing TOD turns, with 3670 and 4768 instances correspondingly according to [22]. Besides, due to the noise Fusedchat dataset itself, e.g. redundant domains, and inconsistency values, which were inherited from MultiWoz2.4 [29], we changed some modifications and also wrote in the Appendix A.

TABLE I
Statistics of SGD, FusedChat, MultiWoz2.4. This statistics is computed on total samples of training, validation and test sets. For FusedChat and MultiWoz2.4, there exist some same numbers because FusedChat is built upon MultiWoz2.4. The difference is FusedChat combines the ODD part to MultiWoz2.4's TOD part. With SGD, number of "unique" slots will be represented (in italic) and number of slot values will contains the number of values of "categorical" slots (in bold).

Statistics / Dataset	FusedChat	SGD	MultiWoz2.4
No. of domains	7	20	7
No. of dialogues	10348	22825	10348
Avg. turn per dialogue	20.3	18.36	12.27
No. of slots	45	*184*	37
No. of slot values	1936	**500**	1936

Schema-Guided Dialogue Dataset [30] is the largest multidomain for task-oriented dialogue datasets until now. It spans 45 diverse domains over hotels, banks, events, homes, travel, flights, media, movies, rental cars, and more. Each part continues to split into various forms, e.g., hotels decomposed as hotels_1, hotels_2, hotels_3, and so on. This dataset utilizes 25 domains for training and reuses the identical domains combined with 10 more ones for validation. This large number of dialogues will offset the minor number one in the Fusedchat dataset. The test set also comprises these 35 domains to evaluate the model's zero-shot learning capabilities.

MultiWoz2.4 Dataset [29] is the last refinement for primarily evaluating metrics on task-oriented dialogue up-to-date. Statistics are the same as preceding MultiWoz versions and are referred to Table II. Because all annotation updates are refined mainly on the validation set and test set, as well as having large numbers of dialogues spanning many domains, we will use this dataset to evaluate zero-shot abilities and compare our model with other backbones. Though MultiWoz2.4 is built originally from version 2.1, which does not leverage versions 2.2 and 2.3, it still has a renowned publication and has been widely used when evaluating research in recent years.

B. Metrics

Joint Goal Accuracy (JGA) is one of the most common metrics used to evaluate the task of dialogue state tracking and is also defined widely as in SGD [24]. With the JGA metric, correct is counted when the belief state prediction exactly matches the truth label. It is also a limitation of these metrics because the label also contains all states of the previous state, which requires the model to predict even slots in previous dialogue turns.

Slot accuracy (SA) [31] is also another standard metric existing along with joint accuracy. This metric is quite naive and calculated by comparing each tuple *(domain, slot, value)* with its label correspondingly. However, this can be problematic if the number of slot values scales as in multi-domain circumstances.

TABLE II
EXPERIMENTAL RESULTS ON THE FUSEDCHAT TEST SET WITH JOIN GOAL ACCURACY (JGA), SLOT ACCURACY (SA), F1-SCORE PERFORMANCE. TWO MODELS FROM FUSEDCHAT [22] IS CITED TO COMPARE FOR JGA AND SA METRICS, THEIR PARAMETERS IS NOT REFERRED IN ORIGINAL PAPER SO THAT WE HIDE IT. ADDITION F1 COLUMN IS REPORTED TO ENSURE PROPER TRACKING OF THE DIALOGUE'S TYPE (ODD OR TOD). OUR GRADTOD MODEL'S PERFORMANCE IS WRITTEN IN ITALICS.

System	# Params.	SA	JGA	F1	
				TOD	OOD
Two-in-one model	-	97.2	59.2	-	-
Classification-based model	-	97.3	60.0	-	-
GradTOD (ours)	*250M*	*99.6*	*82.5*	*97.2*	*95.4*

F1 Score is well-known metrics for measuring model's prediction. Because this metric captures both precision and recall, we can ensure whether there is confusion between the dialogue's type of current turn (ODD or TOD) when computing the F1-score for each classes to check.

C. Experiments

Flan-T5 backbone [24] is a variant of T5 that robustly enhances the generality of instruction finetuning compared with non-finetuned models. Except that, these *flan* models also prove zero-shot ability, which significantly influences our paper on experiments with hybrid dialogue. Here, we use Flan-T5 as the backbone to demonstrate the proposed method in Section III. The zero-shot ability of our model is also presented in Table II.

All previous slot values have to be utilized to compute the JGA score. Here, we clarify that there are two existing formulas. With encoders like FastSGT [1], SGD-base [30], and SGP-DST of [32], these model's abilities can only predict the current slot values and have to use another set to store previous ones. FastSGT and SGD-base combine prior predicted slot values with the current expected state to compute the JGA score, while these prior predicted slot values will be replaced by the gold ones on SGP-DST. On the other hand, encoder-decoder seems naive when encouraging the LLM model itself to predict all previous ones, e.g., SDT [4], D3ST [5], AnyTOD [6]. By using encoder-decoder architecture, we mainly use the second formula to compute the JGA score and also provide results of experiments in Table II and Table III.

D. Results

Implicitly determining ability on FusedChat is demonstrated on Table II. Our model achieves above 95% on the F1 score for each dialogue's type, meaning the model can handle well when predicting the state for TOD or retaining the state when responding to chitchat. On top of that, the *two-in-one* and *classification-based* models are referred from FusedChat [22] to compare the JGA metric. In Table II, these two models have the JGA score of about 59% to 60%, but our GradTOD model enhances performance by approximately 23%. We believe the instruction-finetuned ability of model

TABLE III
EXPERIMENTAL RESULTS ON THE SGD TEST SET WITH JOIN GOAL ACCURACY (JGA) PERFORMANCE ON SEEN AND UNSEEN DOMAINS, THE VALUE WITH LARGE LANGUAGE MODEL (PARAMS MORE THAN 1B) AND OUR GRADTOD MODEL WRITTEN IN BOLD AND IN ITALICS, RESPECTIVELY.

Backbone	System	# Params.	Seen	Unseen	Avg
Encoder	SGD baseline	110M	-	-	25.4
	SGP-DST	660M	-	-	72.2
	multi-task BERT	110M	-	-	82.7
Seq2Seq	D3ST_BASE	220M	92.5	66.4	72.9
	D3ST$_{LARGE}$	770M	93.8	75.4	80.0
	D3ST$_{XXL}$	**11B**	**95.8**	**83.3**	**86.4**
	SDT$_{BASE}$	250M	-	-	76.3
	SDT$_{LARGE}$	800M	-	-	83.2
	SDT$_{XXL}$	**11B**	**95.8**	**86.4**	**88.8**
	AnyTOD$_{BASE}$	220M	89.9	62.4	76.2
	AnyTOD$_{XXL}$	**11B**	**94.8**	**82.2**	**88.5**
Seq2Seq	*GradTOD (ours)*	*250M*	*83.3*	*89.2*	*88.6*

Flan-T5 and the proposed method in section III make the model learn what information needs to include: TOD or ODD classification, action of the current user's text, or state of the current user's text. Besides, because Flan-T5 is pre-trained for the Chain of Thought task, this model will perform three tasks in order and ensure tightness between the three components. More specifically, the type of dialogue will be classified first, and then determining the action of the second task will link the results of the first task to generate appropriate action, which performs the suitable intent of user text. Once the results of the second task are available, our model will track which slot values are represented for the user's state based on the user's actions. For example, a user action that informs a specific slot in the second task will require the necessary value and update that slot in the user state in the third task. Other actions such as request, thank, bye, etc. will not update the user state. From those two reasons, we confirm that using Flan T5 is the most optimal choice at present to achieve such outperforming results in both JGA and SA metrics, 82.5% and 99.6% correspondingly.

Performance on SGD is provided in Table III. Except for evaluating Joint Goal Accuracy on all domains, we also record results on seen and unseen domains to generalize more views. These used models are baseline from SGD [30], SGP-DST [32], multi-task BERT [15], D3ST [5], SDT [4], AT from AnyTOD [6], which used to evaluate on SGD test set. In Table III, recent encoder models like SGP-DST, and multi-task BERT achieve above 72.2% and 82.7% on Average JGA score, except for SGD baseline model here, which only has a 25.4% score. We cite this model because of the encoder backbone itself despite having innovation as SGD+ dataset [1]. On the contrary, Seq2Seq models have better forwarding when performing approximately 89% score. The explanation can come from scaling on the recent model transformer's size and exploring the inner ability of Seq2Seq models. When considering the seen domain, GradTOD has a setback compared to previous models on the seen domain, which

TABLE IV
COMPARISON OF PERFORMANCE BETWEEN STATE-OF-THE-ART RESEARCH ON MULTIWOZ 2.4 TEST SET. THE RESULT OF SOM-DST ON MULTIWOZ 2.4 IS REFERRED TO ON [35]. THE HIGHEST SCORE WITH ENCODER ONLY AND LARGE LANGUAGE MODEL (SEQ2SEQ) ARE WRITTEN IN BOLD WHILE OUR GRADTOD MODEL IS WRITTEN IN *italics*, RESPECTIVELY.

Backbone	System	# Params.	SA	JGA
Encoder	SOM-DST$_{BASE}$	110M	98.84	75.19
	AUX-DST	220M	99.07	78.14
	Meta AUX-DST	**220M**	**99.08**	**78.57**
Seq2Seq	D3ST$_{BASE}$	220M	-	72.1
	D3ST$_{LARGE}$	770M	-	70.8
	D3ST$_{XXL}$	**11B**	-	**75.9**
Seq2Seq	*GradTOD (ours)*	*250M*	*99.03*	*76.9*

only achieved an 83.3% score on JGA metrics. Because this model has performed approximately 89.2% score on unseen ones, this experiment has gained the satisfied expectation when proving the ability to domain adaption (zero-shot ability). With our development, the GradTOD model has reached an 88.6% average score, indicating the correct prediction of our claim about leveraging the instruction-finetuning technique and proposed method.

Performance on MultiWoz2.4 is recorded in Table IV. Here, we follow models: SOM-DST [33], AUX-DST & meta AUX-DST from metaASSIST [34], D3ST [5] and AnyTOD [6] to compare zero-shot ability on domain transfers. Observation from Table IV shows that encoder models handle well for predicting the current state, which even has a higher JGA score than Seq2Seq models. Encoder models have a JGA score of around 75% to 79%, while seq2seq models are in range from 72% to 76%. A capable reason is these existing optimized techniques for encoder models such as slot carryover [15], operation [33] cover underestimation of JGA metrics, but the encoder-decoder ones completely depend on itself to predict user's state, which some cases even have miscellaneously values of system's turns. In contrast, our model with the proposed method has some improvement when evaluating the test set on JGA and SA, with 76.9% and 99.03% correspondingly.

V. CONCLUSION AND FUTURE WORK

In this paper, we demonstrate the ability to combine prompt engineering and conditional generation for handling Dialogue State Tracking. With GradTOD, we can easily design a new business logic with minimal custom ontology to determine policy tasks or domains unseen applications. This experience evaluation on three Benchmark datasets gives promising results, we hope this study can be an interesting field of research and development.

One limitation in our research is context length history can be scaled up and always complicated if has thousands of slot values in multiple domains. Inspire Graph-of-thoughts (GoT) [36], the model can reduce the context length by using a graph to represent an ontology. Simultaneously, we can focus on the weakness between the transition inter-mode ToD and ODD (transition between chitchat and task-oriented guide in the same domain) and outer-mode (transition between each domain) to generalize human-like conversations.

APPENDIX
FUSEDCHAT'S MODIFICATION

The Fusedchat dataset is renowned for its numerous outstanding features, which greatly enhance its usability and potential applications. We use the metadata tag in the next system utterance (u_{t+1}) to create the current state and the dialog_act tag in the current user utterance (u_t) tag to create the current action. However, there are some inconsistencies in the slots and values between these two tags. While these bugs may present challenges in terms of data processing, we have provided solutions for improvement and refinement. By addressing these issues, our goal is to further enhance the quality and reliability of the Fusedchat dataset, making it an even more valuable resource for various research and development endeavors.

- **Metadata miss slots**: one of the bugs that we encountered in the Fusedchat dataset is related to the metadata tags. Specifically, there are some missing slots within the metadata tag, while the dialog_act tag still has the action information for those slots. This inconsistency between the two tags can potentially lead to confusion and inaccuracies during data processing. For example, in dialog MUL1598 in the dialog_act tag in the first log: "Restaurant-Inform": [["area", "north"], ["price range", "expensive"]], and the metadata tag in the second log is only each slot "area": "north". We resolve this issue by implementing a conditional check within the dialog_act tag. If we identify a slot linked to inform action, but that slot is not specified in the metadata tag, we will include it in the metadata tag.
- **Dialog_act miss slots**: this bug will be the opposite of the first error. The user performs an action to provide information, but dialog_act is not recorded with this slot, even though this slot is recorded in the metadata tag. For example, in dialog MUL1598 in the dialog_act tag the 5th log is missing the information "people": "1", but this pair of slot values is present in the metadata tag at the 6th log. We fixed it by checking for extra slots in the metadata tag, if any slot was missing in the dialog_act tag then we added an inform action for that slot in the dialog_act tag
- **Metadata and dialog_act miss slots**: the third bug is the most difficult for us. The user performs the action informing the slots, but these slots are absent from both the metadata tag and the dialog_act tag. For example, in dialog MUL0145 in the dialog_act tag the 9th log and the metadata tag at the 10th log also do not record the information of "for three", "Friday", and "16:00". We have come up with many solutions but they are all

TABLE V
DIFFERENT TYPES OF ERRORS OCCUR IN DIALOGS FROM THE FUSEDCHAT DATASET. EACH ROW IN THE TABLE REPRESENTS AN ERROR TYPE AND THE EXAMPLES SHOW CONFLICTING SLOTS AND VALUES BETWEEN THE DIALOG_ACT TAG AND THE METADATA TAG.

Error Type	Dialog's id	Log user (t)	Current user (t)	Dialog_act (t)	Metadata (t+1)
Metadata miss slots	MUL1598	1	I'm looking for an **expensive** restaurant in the north area	”area”, ”north” **”pricerange”, ”expensive”**	”area”: ”north”
Dialog_act miss slots	MUL1598	5	Lets try Hakka. It will be just **myself** at 16:45 on tuesday.	”bookday”, ”tuesday” ”booktime”, ”16:45” ”name”, ”hakka”	”name”: ”hakka” ”reference”: ”KWV7HGEB” ”time”: ”16:45” ”day”: ”tuesday” **”people”: ”1”** ”food”: ”chinese” ”pricerange”: ”expensive” ”name”: ”not mentioned” ”area”: ”north”
Metadata and Dialog_act miss slots	MUL0145	9	I would like to book a table for **three** on **Friday**, at **16:00**.	[]	”people”: ”” ”day”: ”” ”time”: ”” ”food”: ”european” ”pricerange”: ”moderate” ”name”: ”not mentioned” ”area”: ”centre”
Dialog_act miss slots and values	PMUL4645	11	I also need a taxi.	”none”, ”none”	-
	MUL2285	13	I do not need to book.	[]	-
Inconsistent values	PMUL1554	7	uh why yes you are very perceptive depart cambridge arrive by **9:45** leaves on sat going to ely	**”arriveby”, ”9:45”** ”departure”, ”cambridge” ”destination”, ”ely”	”leaveAt”: ”not mentioned” ”destination”: ”ely” ”day”: ”saturday” **”arriveBy”: ”09:45”** ”departure”: ”cambridge”

ineffective, so we decided to filter data like this to not have a negative impact on the training results.

- **Dialog_act miss slots and values**: there are some elements in dialog_act that, although the action is informed, do not contain any slots or values, or both slot and value are none. This is obviously meaningless so we decided to eliminate these samples during training. For example, dialog PMUL4645, in the dialog_act tag at 11th log: ”Taxi-Inform”: [[”none”, ”none”]]. Or dialog MUL2285, at 13th log: ”Train-Inform”: []
- **Inconsistent values**: there is another small bug that is the inconsistency of values in the whole data set such as [dontcare, dont care, don't care] will be standardized to dontcare. The time values are not consistent between the metadata tag and the dialog_act tag. For example, in dialog PMUL1554 in the 7th log, the time in the text is 9:45, but the value of slot arrive in the metadata tag is 09:45 and the value in the dialog_act tag is 9:45. These value inconsistency errors are not too serious, but they will negatively affect training because there are no common standards. And more importantly, the JGA metric result will be low, because if the label is don't care, and the prediction is don't care, then JGA will be 0, this is similar to 9:45 and 09:45.

REFERENCES

[1] V. Noroozi, Y. Zhang, E. Bakhturina, and T. Kornuta, “A fast and robust bert-based dialogue state tracker for schema-guided dialogue dataset,” *arXiv preprint arXiv:2008.12335*, 2020.

[2] Y. Hu, C.-H. Lee, T. Xie, T. Yu, N. A. Smith, and M. Ostendorf, “In-context learning for few-shot dialogue state tracking,” *arXiv preprint arXiv:2203.08568*, 2022.

[3] C.-M. Lai, M.-H. Hsu, C.-W. Huang, and Y.-N. Chen, “Controllable user dialogue act augmentation for dialogue state tracking,” *arXiv preprint arXiv:2207.12757*, 2022.

[4] R. Gupta, H. Lee, J. Zhao, Y. Cao, A. Rastogi, and Y. Wu, “Show, don't tell: Demonstrations outperform descriptions for schema-guided task-oriented dialogue,” in *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2022.

[5] J. Zhao, R. Gupta, Y. Cao, D. Yu, M. Wang, H. Lee, A. Rastogi, I. Shafran, and Y. Wu, “Description-driven task-oriented dialog modeling,” *arXiv preprint arXiv:2201.08904*, 2022.

[6] J. Zhao, Y. Cao, R. Gupta, H. Lee, A. Rastogi, M. Wang, H. Soltau, I. Shafran, and Y. Wu, “Anytod: A programmable task-oriented dialog system,” *ArXiv*, vol. abs/2212.09939, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:254877048

[7] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, “Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension,” *arXiv preprint arXiv:1910.13461*, 2019.

[8] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, “Language models are unsupervised multitask learners,” *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[9] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, “Exploring the limits of transfer

learning with a unified text-to-text transformer," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5485–5551, 2020.

[10] E. Hosseini-Asl, B. McCann, C.-S. Wu, S. Yavuz, and R. Socher, "A simple language model for task-oriented dialogue," *Advances in Neural Information Processing Systems*, vol. 33, pp. 20 179–20 191, 2020.

[11] K. M. Yoo, H. Lee, F. Dernoncourt, T. Bui, W. Chang, and S.-g. Lee, "Variational hierarchical dialog autoencoder for dialog state tracking data augmentation," *arXiv preprint arXiv:2001.08604*, 2020.

[12] D. J. Shah, R. Gupta, A. A. Fayazi, and D. Hakkani-Tur, "Robust zero-shot cross-domain slot filling with example values," *arXiv preprint arXiv:1906.06870*, 2019.

[13] B. King and J. Flanigan, "Diverse retrieval-augmented in-context learning for dialogue state tracking," *arXiv preprint arXiv:2307.01453*, 2023.

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[15] E. Kapelonis, E. Georgiou, and A. Potamianos, "A multi-task bert model for schema-guided dialogue state tracking," *arXiv preprint arXiv:2207.00828*, 2022.

[16] Z. Chen, L. Chen, B. Chen, L. Qin, Y. Liu, S. Zhu, J.-G. Lou, and K. Yu, "Unidu: Towards a unified generative dialogue understanding framework," *arXiv preprint arXiv:2204.04637*, 2022.

[17] R. Su, J. Yang, T.-W. Wu, and B.-H. Juang, "Choice fusion as knowledge for zero-shot dialogue state tracking," in *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2023, pp. 1–5.

[18] Z. Lin, B. Liu, A. Madotto, S. Moon, P. Crook, Z. Zhou, Z. Wang, Z. Yu, E. Cho, R. Subba *et al.*, "Zero-shot dialogue state tracking via cross-task transfer," *arXiv preprint arXiv:2109.04655*, 2021.

[19] M. Moradshahi, S. J. Semnani, and M. S. Lam, "Zero and few-shot localization of task-oriented dialogue agents with a distilled representation," in *Conference of the European Chapter of the Association for Computational Linguistics*, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:257038848

[20] R. Goel, W. Ammar, A. Gupta, S. Vashishtha, M. Sano, F. Surani, M. Chang, H. Choe, D. Greene, K. He *et al.*, "Presto: A multilingual dataset for parsing realistic task-oriented dialogs," *arXiv preprint arXiv:2303.08954*, 2023.

[21] K. Sun, S. Moon, P. A. Crook, S. Roller, B. Silvert, B. Liu, Z. Wang, H. Liu, E. Cho, and C. Cardie, "Adding chit-chat to enhance task-oriented dialogues," in *North American Chapter of the Association for Computational Linguistics*, 2020.

[22] T. Young, F. Xing, V. Pandelea, J. Ni, and E. Cambria, "Fusing task-oriented and open-domain dialogues in conversational agents," in *AAAI Conference on Artificial Intelligence*, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:237453371

[23] Z. Chen, B. Liu, S. Moon, C. Sankar, P. A. Crook, and W. Y. Wang, "Ketod: Knowledge-enriched task-oriented dialogue," *ArXiv*, vol. abs/2205.05589, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:248693217

[24] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, E. Li, X. Wang, M. Dehghani, S. Brahma, A. Webson, S. S. Gu, Z. Dai, M. Suzgun, X. Chen, A. Chowdhery, D. Valter, S. Narang, G. Mishra, A. W. Yu, V. Zhao, Y. Huang, A. M. Dai, H. Yu, S. Petrov, E. H. hsin Chi, J. Dean, J. Devlin, A. Roberts, D. Zhou, Q. V. Le, and J. Wei, "Scaling instruction-finetuned language models," *ArXiv*, vol. abs/2210.11416, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:253018554

[25] P. Gupta, C. Jiao, Y.-T. Yeh, S. Mehri, M. Eskenazi, and J. P. Bigham, "Instructdial: improving zero and few-shot generalization in dialogue through instruction tuning," in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, 2022, pp. 505–525.

[26] Z. Lin, B. Liu, S. Moon, P. Crook, Z. Zhou, Z. Wang, Z. Yu, A. Madotto, E. Cho, and R. Subba, "Leveraging slot descriptions for zero-shot cross-domain dialogue state tracking," *arXiv preprint arXiv:2105.04222*, 2021.

[27] C.-H. Lee, H. Cheng, and M. Ostendorf, "Dialogue state tracking with a language model using schema-driven prompting," *arXiv preprint arXiv:2109.07506*, 2021.

[28] P. Budzianowski, T.-H. Wen, B.-H. Tseng, I. Casanueva, S. Ultes, O. Ramadan, and M. Gasic, "Multiwoz - a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling," in *Conference on Empirical Methods in Natural Language Processing*, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:52897360

[29] F. Ye, J. Manotumruksa, and E. Yilmaz, "Multiwoz 2.4: A multi-domain task-oriented dialogue dataset with essential annotation corrections to improve state tracking evaluation," in *SIGDIAL Conferences*, 2021.

[30] A. Rastogi, X. Zang, S. Sunkara, R. Gupta, and P. Khaitan, "Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset," in *AAAI Conference on Artificial Intelligence*, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:202565722

[31] C.-S. Wu, A. Madotto, E. Hosseini-Asl, C. Xiong, R. Socher, and P. Fung, "Transferable multi-domain state generator for task-oriented dialogue systems," *ArXiv*, vol. abs/1905.08743, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:160009896

[32] Y.-P. Ruan, Z. Ling, J.-C. Gu, and Q. LIU, "Fine-tuning bert for schema-guided zero-shot dialogue state tracking," *ArXiv*, vol. abs/2002.00181, 2020.

[33] S. Kim, S. Yang, G. Kim, and S.-W. Lee, "Efficient dialogue state tracking by selectively overwriting memory," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020.

[34] F. Ye, X. Wang, J. Huang, S. Li, S. Stern, and E. Yilmaz, "Metaassist: Robust dialogue state tracking with meta learning," in *Conference on Empirical Methods in Natural Language Processing*, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:253098815

[35] F. Ye, Y. Feng, and E. Yilmaz, "ASSIST: Towards label noise-robust dialogue state tracking," in *Findings of the Association for Computational Linguistics: ACL 2022*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 2719–2731.

[36] Y. Yao, Z. Li, and H. Zhao, "Beyond chain-of-thought, effective graph-of-thought reasoning in large language models," *arXiv preprint arXiv:2305.16582*, 2023.