Efficient Task-Oriented Dialogue State Tracking using Language Models

Chia-Hsuan Lee

A dissertation

submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

University of Washington
2024

Reading Committee:
Mari Ostendorf, Chair
Noah Smith
Hannaneh Hajishirzi

Program Authorized to Offer Degree:
Electrical and Computer Engineering

University of Washington

Abstract

Efficient Task-Oriented Dialogue State Tracking using Language Models

Chia-Hsuan Lee

Chair of the Supervisory Committee:

Mari Ostendorf
Electrical and Computer Engineering

In the digital era, humans depend on automated systems to accomplish a wide variety of tasks that used to require talking to a human agent, such as booking flights and hotels for their trips. This demand has driven the development of AI assistants that enable users to interact with databases and APIs through conversational interfaces. Training these AI assistants requires large amounts of data, and annotating conversational data is expensive. Therefore, developing models that can learn from small amounts of data is crucial. As the scale of large language models (LLM) driving AI assistants continues to grow, their computational requirements also increase significantly. Thus, the key challenges this thesis addresses are creating computation-efficient and data-efficient models that maintain high task performance.

A core component of a task-oriented AI assistant is Dialogue State Tracking (DST), which deciphers user intents from conversational histories, thereby mapping user goals to predefined schema for executing task-specific queries. In this thesis, we introduce our proposed solutions to the critical challenges in building computation-efficient and data-efficient dialogue state tracking systems. The initial approach concerns how to build effective DST systems with smaller LMs (SLM) that have lower computation requirements. To facilitate this line of research, we first introduce a prompt-based finetuning framework SDP-DST, that enriches SLM inputs with schema-specific textual information. This enables task-aware contextualization of the input conversations and significantly benefits SLM-based DST systems without losing generalization abilities. This schema-driven prompting framework advances the SOTA on a DST dataset despite using

fewer model parameters.

Second, learning efficiently from conversations is critical to facilitate building task-oriented agents on new services that have limited annotated data. We introduce a few-shot learning framework IC-DST, that leverages the in-context learning (ICL) abilities of LLM where models can make predictions given task instructions or a few exemplars. This framework leverages a prompt format with DST ontology descriptions and reformulates DST as a text-to-SQL task to improve ICL performance, advancing the state-of-the-art (SOTA) few-shot DST without any parameter updates.

To combine the benefits of the two previous approaches, we propose a novel routing framework ORCHESTRALLM, that simultaneously leverages an SLM and an LLM along with a router that dynamically assigns instances to either LM in inference time to save computing. Hypothesizing that conversational turns with similar semantic embeddings are of the same difficulty level, the router selects an appropriate LM expert based on embedding distances between the testing instance and instances in pre-stored LM expert pools. We demonstrate the effectiveness of the routing framework on two different multi-domain DST benchmarks under low-resource settings. The SLM can be trained with less data when combined with an LLM to handle more difficult cases. The proposed framework capitalizes on the expertise of different LMs, outperforming standalone systems while also achieving a substantial reduction of over 50% in computational costs.

Lastly, we propose an alternative strategy to achieve better data efficiency and computation efficiency using a self-correction system with only an SLM, capitalizing on recent technological advances that have made SLMs more powerful. LLMs have been developed to make corrections to their outputs through self-provided (or external) feedback when prompted, particularly in code and math reasoning tasks. While self-correction improves task performance, it requires multiple inference passes from the LLM, including feedback/verification outputs and final correction outputs. To enhance the efficiency of DST systems, we introduce a novel paradigm CORRECTIONLM, where we fine-tune SLMs to correct predictions using a small amount of annotated data, alleviating the need for feedback from LLMs. Evaluation results on two DST datasets reveal that SLMs can correct errors effectively.

Acknowledgements

I would like to express my deepest gratitude to my advisor, Mari Ostendorf. Her unwavering belief in me and her exceptional dedication to her students have been truly inspiring. She has taught me the importance of critical thinking in research and has guided me in articulating my ideas both in writing and public speaking.

I am also profoundly thankful to all of my committee members. Noah Smith has been instrumental in shaping my research mindset and generously providing resources. Hao Cheng, who has been an incredible mentor since my first day at UW, has supported me in every conceivable way and has greatly enhanced my technical skills. Hannaneh Hajishirzi has offered invaluable feedback on my thesis and has also guided me in the art of teaching. I am also grateful to Lucy Wang and Shane Steinert-Threlkeld, who served as GSRs, enriching my general and final exams with insightful discussions.

I feel fortunate to be a part of the TIAL lab and the UWNLP group, where I have been surrounded by supportive and inspiring individuals. My heartfelt thanks go to all the members. I extend special thanks to my collaborators and alumni in the UWNLP group, especially Tao Yu and Ellen Wu. Their invaluable advice and guidance during moments of uncertainty have made them exemplary figures in my research journey. I thank Akari Asai, Yizhong Wang, and Jungo Kasai, for making our workshop at NAACL an unforgettable experience. I also thank Yushi Hu, Weijia Shi, and Sewon Min for the fruitful research collaboration and discussions.

I am also deeply grateful to my first PhD internship mentors, Alex Polozov and Matthew Richardson, who showed immense patience with a novice like me and meticulously taught me various research skills. My thanks also go to my mentors at Google, Melvin Johnson, who provided me with ample space and resources to unleash my creativity, and Ankur Bapna, who taught me the passion for research and collaboration.

To my friends in Seattle and my PhD friends across the US, UK, and Taiwan, your unwavering psycho-

logical support and practical assistance during my studies and job search have been invaluable.

Finally, I am profoundly thankful to my parents and sister. Though my career path has diverged from their expectations, their unconditional love and support have given me the freedom to pursue my dreams.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In our daily lives, humans perform a wide variety of digital tasks, from managing calendars and booking air tickets to setting alarms. These tasks requires interactions with various applications, knowledge sources, and databases, requiring an intermediary to facilitate communication between humans and machines. In the modern context, this role is increasingly fulfilled by artificial intelligence (AI) assistants such as Google Gemini, Apple Intelligence, and OpenAI ChatGPT. To provide human users with requested information or execute digital tasks, the most critical module is for AI assistants to precisely interpret user intents in the context of a multi-turn dialogue. The salient information mentioned by users is mapped to the corresponding information in the backend database. This mapping process is also referred to as dialogue state tracking (DST). User goals are represented as slot values within a predefined schema (e.g. restaurant-bookpeople-2, train-destination-Seattle), which outlines the information needed to execute task-specific queries to the backend. Users might update their information requests or simply change topics to use different services. Therefore it is important for agents to be able to have strong context understanding abilities. Earlier task-oriented systems [Wen et al., 2017; Henderson et al., 2014; Shah et al., 2018a] were limited to single-domain dialogues, where the conversation topic was restricted to one domain (e.g., restaurant, movie). More recent works, however, have focused on multi-domain dialogues to better reflect real-world scenarios and build more practical conversational agents [Budzianowski et al., 2018; Rastogi et al., 2020b; Chen et al., 2021a].

In recent years, language models (LM) have obtained state-of-the-art (SOTA) performance on diverse generation and understanding tasks including bi-directional encoder style language models [Devlin et al.,

2019; Liu et al., 2019], auto-regressive language models[Radford et al., 2019; Brown et al., 2020; Touvron et al., 2023; Reid et al., 2024] and sequence-to-sequence language models [Raffel et al., 2020]. This thesis focuses on different strategies for using language models for DST. Our initial work leveraged a sequence-to-sequence LM and was among the early efforts which used small language models (SLMs). Now LMs are widely used to build DST systems. More recently, large language models (LLMs) have also been used in our work and that of others [Hu et al., 2022; King and Flanigan, 2023; Li et al., 2024]. We explore different configurations of SLMs and LLMs to advance the SOTA, while also improving model efficiency. As the field of LMs has rapidly evolved in recent years, the SLMs and LLMs used in our experiments have also advanced. We do not rely on the same SLM and LLM across all works; instead, we leverage more advanced models as they become available.

Annotating conversational data is both time-consuming and expensive, and the nature of rich in-domain knowledge within task-oriented dialogues further complicates the development of systems for new domains or services. This underscores the importance of creating data-efficient AI assistants. Moreover, as AI assistants gain popularity across diverse application scenarios, there is a growing demand for deploying computationally efficient solutions on edge devices or cloud services. Edge devices require systems that optimize model weight parameter usage, while cloud services focus on finding ways to minimize the computational demands of LLM.

1.1 Thesis Contributions

This thesis aims to improve data efficiency and computation efficiency for DST using LMs. We highlight our contributions in this section.

1.1.1 Data-Efficiency

Collecting and annotating turn-level dialogue states is both challenging and expensive [Budzianowski et al., 2018]. To overcome this issue, Previous studies have explored zero-shot or few-shot DST. Most prior few-shot learning methods for DST are based on finetuning smaller language models [Wu et al., 2019; Shin et al., 2022; Lin et al., 2021a] where "few shot" is defined as a small percentage of the original training set (1-10%). These systems need to be retrained when new slots or domains are added to the schema and

thus are less flexible. In this thesis, we propose the first successful few-shot LLM system for DST, IC-DST [Hu et al., 2022] that does not require retraining. IC-DST leverages the in-context learning abilities of LLM that can make predictions based on just a few task demonstrations selected from a small training set. To facilitate better use of tabular information of schema, we reformulate DST into a text-to-SQL problem. The effectiveness of IC-DST is shown on the public MultiWOZ benchmark [Budzianowski et al., 2018] in which it advanced SOTA and demonstrates strong zero-shot learning abilities.

1.1.2 Computation-Efficiency

Many previous approaches for Dialogue State Tracking (DST) employ domain-specific architectures with specialized classifiers, such as those designed to detect slot types [Ye et al., 2021; Chen et al., 2020; Wu et al., 2019; Kim et al., 2020; Lin et al., 2020]. These modules require extensive training efforts and demand more model parameters. In this thesis, we introduce SDP-DST [Lee et al., 2021], which demonstrates that training with schema-driven prompts can facilitate task-aware contextualization and significantly enhance the performance of language model-based dialogue systems while requiring fewer model parameters. Experimental results on the public MultiWOZ dataset show that we obtained then SOTA with an SLM when using the full MultiWOZ training set.

For data efficiency, LLMs have been utilized to build systems for a wide range of downstream tasks, including DST, but they require substantial computational resources. To reduce the computational demands of LLMs, hybrid strategies that combine SLMs with LLMs have emerged. One approach, known as cascade-based, routes a query to an LLM only if an SLM cannot handle it [Chen et al., 2023b; Madaan et al., 2023]. While being simple in design, this method introduces latency and computational redundancy by constantly querying SLMs. Another strategy uses binary classifiers to determine the most suitable LM for each task [Kag et al., 2022; Šakota et al., 2023]. However, this approach requires frequent retraining whenever new models are added, posing a significant limitation. In this thesis, we propose a new routing framework, ORCHESTRALLM [Lee et al., 2024], which seamlessly integrates an SLM and an LLM, coordinated by a retrieval-based router. During inference, this dynamic router directs instances to either the SLM or LLM based on their semantic embedding distances with the retrieved LM exemplars, leveraging the strengths of both models. Our evaluation on public MultiWOZ and SGD benchmarks demonstrates that ORCHES-

TRALLM outperforms LLM-based systems while also achieving computational cost savings of over 50%.

While routing helps save computational resources by activating the LLM less often, the involvement of an LLM still poses deployment challenges. To address this, we further explore achieving data-efficient and computation-efficient without using an LLM, focusing on few-shot SLMs. LLMs have shown remarkable abilities to reflect on predictions (whether their own or from other models) and make subsequent corrections. This approach has been studied across various tasks, such as programming and math reasoning [Shinn et al., 2024; Madaan et al., 2024]. Despite being useful, most correction methods depend on LLMs for feedback and sequential corrections, leading to significant computational challenges. To address this, previous methods have explored fine-tuning SLMs to mimic the feedback and corrections generated by LLMs [Zhang et al., 2024b; Yu et al., 2024a]. However, they still require using LLMs. In this thesis, we propose CORRECTIONLM, a method that fine-tunes SLMs to serve as correction models without any involvement of LLMs. We prompt SLMs with in-context exemplars and use the errors they make on the training set to train the SLMs. Experimental results show that CORRECTIONLM improves accuracy by over 40% relative to strong baselines. Additionally, CORRECTIONLM outperforms ORCHESTRALLM, achieving superior results with five times less data and ten times less computational resources on the public MultiWOZ benchmark.

1.2 Thesis Overview

To summarize, this thesis addresses two essential challenges in building task-oriented AI assistants: improving data efficiency and enhancing computational efficiency to better fulfill execution and information requests from human users. The remainder of this thesis is organized as follows: In Chapter 2, we provide a detailed background and relevant literature on task-oriented dialogue systems, focusing on the specific challenges that this thesis aims to address, such as the limitations of existing models in terms of data and computational efficiency. In Chapter 3, we introduce SDP-DST, an approach that leverages schema-driven prompts to train an SLM to predict slot values. This chapter specifically addresses the challenge of computational efficiency by reducing the need for extensive model parameters and specialized architectures, demonstrating that a small language model can achieve state-of-the-art performance with a well-structured approach. In Chapter 4, we present IC-DST, a framework designed to tackle the challenge of data efficiency by employing in-context learning with an LLM. In this chapter, we demonstrate how IC-DST can

effectively use a limited amount of annotated data, advancing the state-of-the-art. In Chapter 5, we introduce ORCHESTRALLM, a novel routing framework that combines the strengths of both an SLM and an LLM to optimize for both data and computational efficiency. This chapter shows how ORCHESTRALLM dynamically selects the appropriate model based on the specific characteristics of each dialogue turn, achieving superior few-shot performance while significantly reducing computational costs. In Chapter 6, we focus on further enhancing computational and data efficiency through CORRECTIONLM, a self-correction framework that fine-tunes an SLM to correct its own errors without the need for LLM involvements. This chapter demonstrates how CORRECTIONLM can outperform more computationally expensive models in few-shot settings by leveraging in-context learning and effective error correction strategies. Finally, in Chapter 7, we conclude the thesis by summarizing the contributions made in each chapter and proposing future directions for advancing task-oriented AI assistants.

Chapter 2

Background

In this chapter, we begin with an overview of language models in Section 2.1, as all the methods discussed in this thesis build upon them. Section 2.2 provides a detailed overview of the datasets used for DST, along with related work and evaluation methods, with an emphasis on the multi-domain scenarios that are central to this research. Finally, Section 2.3 introduce approaches to enhance computational efficiency, including model compression, sample-adaptive inference, and model switching, which are critical for optimizing the performance of DST systems within the constraints of limited resources.

2.1 Language Models

2.1.1 Architectures

Large-scale pretrained language models have obtained state-of-the-art performance on diverse generation and understanding tasks including bi-directional encoder style language models (BERT and RoBERTa) [Devlin et al., 2019; Liu et al., 2019], auto-regressive language models (GPT-2 and GPT-3) [Radford et al., 2019; Brown et al., 2020] and more flexible sequence-to-sequence language models (T5) [Raffel et al., 2020]. We use bi-directional language models for retrievers in IC-DST, ORCHESTRALLM, and CORRECTIONLM. Sequence-to-sequence language models are used in SDP-DST and for the SLM in ORCHESTRALLM. Auto-regressive language models are used in IC-DST, for the LLM in ORCHESTRALLM, and across all LMs in CORRECTIONLM.

2.1.2 Finetuning and Prompting

Pretraining involves training a language model on a vast dataset to learn general language patterns and representations. Finetuning takes this pretrained model and further trains it on a smaller, task-specific dataset to adapt it to a particular application. Devlin et al. [2019] demonstrate that large-scale pretraining can benefit a wide range of downstream tasks by subsequently finetuning with one additional output layer. All the LMs in this thesis are pretrained.

Another line of research explores freezing the pretrained model weights and injects trainable rank decomposition matrices into each layer of the model architecture [Liu et al., 2024; Dettmers et al., 2024] which reduces the total number of parameters to train and is therefore better suited for limited data scenarios. We use Qlora [Dettmers et al., 2024] to efficiently finetune a Llama3 [Dubey et al., 2024] 8B model in CORRECTIONLM.

Radford et al. [2019] show language models can perform zero-shot transfer to new tasks with natural language prompts without finetuning when they are trained on multiple tasks, inspiring research around prompting language models. Prompting has also been used to provide customized task information during finetuning and improves data efficiency in low data regimes [Schick and Schütze, 2021; Le Scao and Rush, 2021; Raffel et al., 2020]. Our prompt-based DST model, described in Chapter 3 is largely inspired by this line of work where we design textual prompts driven by schema information to enhance task contextualization of inputs. Instead of leveraging human-authored prompts, Shin et al. [2020] propose to automatically design prompts for different tasks. Li and Liang [2021] and Qin and Eisner [2021] show it is also possible to use continuous embeddings as prompts instead of using natural language discrete tokens.

Brown et al. [2020] introduce in-context learning (ICL), a few-shot prompting approach that leverages LLM to make predictions by putting a few task demonstrations and/or instruction into the prompt. This approach does not require any parameter tuning. Improving the quality of in-context examples [Gao et al., 2021] and taking in-context examples similar to the query [Liu et al., 2022] can improve ICL performance. We introduce a few-shot DST system based on ICL in Chapter 4. In-context examples can also be used to finetune small or moderate-sized LMs (in-context tuning) [Min et al., 2022; Chen et al., 2022] and can benefit cross-task generalization. Our self-correction system, described in Chapter 6, leverages in-context tuning to train an SLM to correct errors.

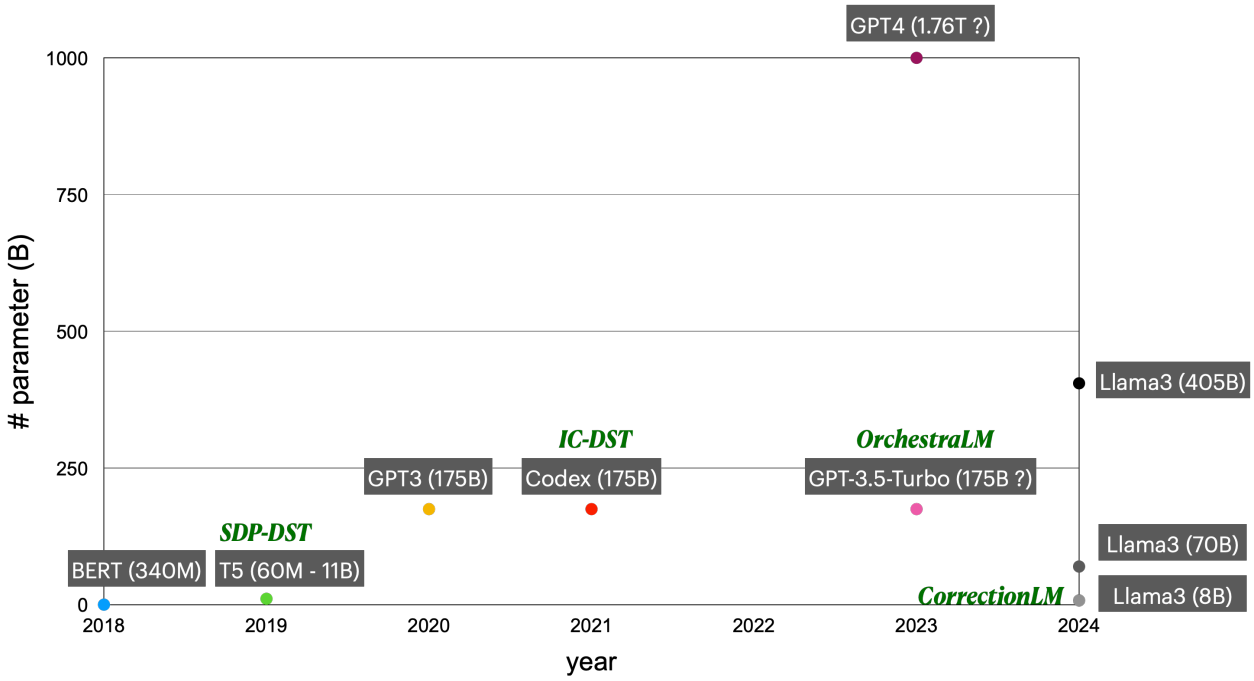

Figure 2.1: A timeline of language model development, illustrating the evolution in model size over the years and highlighting the specific models employed in this thesis. The figure traces the progression from early models like BERT and T5 to more recent, larger-scale models such as GPT-4 and Llama3. The models used in this research, including SDP-DST, IC-DST, ORCHESTRALLM, and CORRECTIONLM, are marked to emphasize their relative sizes and positions within the broader context of language model advancements. Sizes for GPT are not published and are indicated as estimates, with a '?'.

Wei et al. [2022b] introduce chain-of-thought prompting that involves guiding a language model to generate a step-by-step reasoning process to arrive at an answer, mimicking human problem-solving. When prompted, LLMs have demonstrated the ability to provide feedback on their own outputs and subsequently refine them based on that feedback [Shinn et al., 2024; Madaan et al., 2024; Huang et al., 2023; Saunders et al., 2022].

2.1.3 Evolution of Language Model Sizes

The definitions of small and large language models have been shifting as the landscape of language models has evolved in recent years. In our early work, the SLM we used had 60M parameters. In 2024, the state-of-the-art, open-source Llama3 model, for instance, is available in three distinct sizes: 8B, 70B, and 405B parameters. In the context of cutting-edge models, the 8B variant is considered relatively small but it is much bigger than earlier small models. For the purposes of this thesis, we categorize models based on relative

standards at the time of the work, but all models deemed large have more than 175B params and all small LMs have less than 10B params.

Throughout this thesis, various models are utilized according to their sizes and capabilities. T5-60M and T5-220M, classified as small models, are employed in SDP-DST. The Codex model serves as the backbone for IC-DST, while T5-60M, T5-220M, and GPT-3.5-Turbo are integrated within ORCHESTRALLM. For CORRECTIONLM, Llama3-8B is leveraged, reflecting the adaptability and evolution of model sizes within the broader context of this thesis.

2.2 Dialogue State Tracking

2.2.1 Early Task-Oriented Dialogue Systems

Young [2000] defined the framework of a task-oriented system, including a parser to convert user words into a structured representation in order to interact with back-end databases, later known as dialogue state tracking (DST). Following that, different approaches for DST have been proposed, including heuristic scores [Higashinaka et al., 2003], Bayesian networks [Williams and Young, 2007], and discriminative models [Bohus and Rudnicky, 2006].

Most of the previously mentioned works cannot be directly compared due to the lack of a common test bed. To mitigate such an issue, Williams et al. [2013] start the Dialogue State Tracking Challenge (DSTC). Wen et al. [2017] collect human-human conversations via crowd-sourcing by employing the Wizard-of-Oz framework [Kelley, 1984]. In order to diversify dialogue contents, Shah et al. [2018a] propose a machine-to-machine (M2M) approach to generate outlines for each turn and utilize crowd-sourcing to rewrite the outlines into natural language utterances. However, all of the above data resources focus on single-domain dialogues.

2.2.2 Multi-Domain Task-Oriented Dialogue Systems

In order to study in a more realistic setting and create practical TOD systems, Budzianowski et al. [2018] build upon [Wen et al., 2017] and collect a multi-domain human-human conversation dataset, MultiWOZ that spans over 8 domains and contains over 10k human-human written dialogues. It has been one of the

most popular benchmarks in the DST literature. After the publication of Budzianowski et al. [2018], many works improve the label qualities, e.g. MultiWOZ 2.1 [Eric et al., 2020] and MultiWOZ 2.4 [Ye et al., 2022]. Following a similar data collection paradigm as M2M [Shah et al., 2018a], Rastogi et al. [2020b] collect SGD, a dataset that possesses 16k multi-domain conversations spanning 41 domains and focuses on evaluating generalization abilities across unseen domains. 15 out of 21 domains in the test set are not present in the training set and 77% of the dialogue turns in the test set contain at least one domain not present in the training set. Chen et al. [2021a] collect a dataset that studies fine-grained dialogue acts with the Wizard-of-Oz framework. In this thesis, we utilize the MultiWOZ and SGD datasets due to their widespread adoption and the complexity of their schemas.

The multi-domain TOD datasets have spurred the development of many models. There are primarily two paradigms of DST systems. *Classification-based* models [Ye et al., 2021; Chen et al., 2020] formulate DST as a classification problem and pick the candidates from the oracle list of possible slot values. The assumption of full access to the schema prevents them from deploying to different domains. On the other hand, *generation-based* models predict slot values in an autoregressive manner, making it possible to generalize to unseen domains and values. Wu et al. [2019] produce the slot values by using a decoder with a copy mechanism over the joint distribution of the open vocabulary and the input utterances. Kim et al. [2020] consider dialogue states as stored memory and predict pre-defined operations for each slot, enabling efficient updates by predicting values only for slots identified as update status. Most of these models require task-specific designs. For example, Wu et al. [2019] require a slot gate predictor, and Kim et al. [2020] require a state operation predictor. A hybrid strategy has also been studied [Zhang et al., 2020], i.e. a classification module is used for categorical slots and a generative module is used for non-categorical slots.

The *Classification-based* and *generation-based* approaches to DST represent two major paradigms for handling the task. These paradigms are largely orthogonal to the underlying technology used, meaning that they can be implemented with or without leveraging LMs. While some of the earlier models relied heavily on task-specific designs and did not utilize LMs (e.g. [Wu et al., 2019]), others have integrated LMs to enhance performance and flexibility (e.g. [Kim et al., 2020]). This integration of LMs has led to a new wave of DST systems. In the following section, we delve into the evolution and impact of language models specifically within the context of DST.

2.2.3 Language Models for Dialogue State Tracking

The earliest systems finetuned from autoregressive pretrained LMs have been developed [Ham et al., 2020; Hosseini-Asl et al., 2020; Peng et al., 2021] did not require task-specific modules but required additional supervision as inputs. Both Ham et al. [2020] and Hosseini-Asl et al. [2020] require dialogue acts as inputs. Both Hosseini-Asl et al. [2020] and Peng et al. [2021] require DB search results as inputs. Our work, SDP-DST [Lee et al., 2021] is finetuned from a sequence-to-sequence language model and doesn't require task-specific modules.

Subsequent to the publication of SDP-DST [Lee et al., 2021], other systems that are finetuned from sequence-to-sequence pretrained LMs have been developed to build multi-tasking dialogue agents, such as state tracking, response generation, and intent detection. Bang et al. [2023] utilize adapters [Houlsby et al., 2019] to achieve efficient multi-task learning. Imrattanatrai and Fukuda [2023] use a separate encoder to embed schema and associated descriptions. Su et al. [2022] finetune the LM with additional task descriptions, leveraging the instruction following abilities from T5. These systems perform on par with SDP-DST on the MultiWOZ full training benchmark.

To reduce the need for labeled data in DST, few-shot methods based on fine-tuning pretrained LMs have been proposed [Wu et al., 2020b; Li et al., 2021; Su et al., 2022; Shin et al., 2022; Lin et al., 2021b; Xie et al., 2022] using 1-10% of the standard training set as noted earlier. However, these systems are less flexible as they require retraining when new slots or domains are added, and fine-tuning LMs is computationally expensive. To address these challenges, Xie et al. [2022] and Madotto et al. [2021] were the first to apply ICL for few-shot DST, but their systems underperformed compared to previous fine-tuning-based methods. Our work, IC-DST [Hu et al., 2022], is the first successful system to apply ICL to DST. King and Flanigan [2023] further improved this approach by reformulating ICL as a Python programming task and diversifying the retrieved in-context exemplars.

Another line of research focuses on large-scale pretraining, training LMs on diversified dialogue corpora [Wu et al., 2020a; Peng et al., 2022; He et al., 2022; Zhang et al., 2024a]. Building powerful pretrained LMs is orthogonal to the contributions of this thesis.

2.2.4 DST Evaluation

In the evaluation of Dialogue State Tracking (DST) systems, the metric most often employed is the Joint Goal Accuracy (DST JGA) [Henderson et al., 2014]. This metric assesses the correctness of the dialogue state at each turn and deems it accurate only if all slot values within every domain precisely match the ground-truth values including null-valued slots. The reported DST JGA is the average of the turn-level DST JGA scores. A limitation of DST JGA lies in the accumulative nature of dialogue state, which tends to disproportionately penalize early mistakes, as they can be challenging to rectify in subsequent turns. To address this issue, Flexible Goal Accuracy (FGA) [Dey et al., 2022] has been proposed, which introduces a partial penalty for inherited errors, providing a more forgiving evaluation metric.

A limitation of both DST JGA and FGA is that empty slots dominate the score for complex multi-domain schema. Average Goal Accuracy (AGA) [Rastogi et al., 2020b] focus solely on non-empty slots in the gold states, which, however, may overlook false positive predictions. In contrast, Relative Slot Accuracy (RSA) [Kim et al., 2022] take into account all non-empty slots present in both the predicted and gold states, offering a more comprehensive assessment. An alternative method for reducing emphasis on empty slots is to use an F1 measure.

For turn-level performance evaluation, Turn-Level Belief (TLB JGA) [Dey et al., 2022] has been introduced. TLB JGA assesses local predictions at each turn independently, without considering slot values from previous turns, providing insights into performance at a finer granularity. These various evaluation metrics cater to different aspects of DST system performance and can be employed depending on the specific research goals and requirements.

In this thesis, we primarily report dialogue-level JGA (denoted as DST JGA) to evaluate long-context understanding abilities of DST systems. We also report turn-level JGA (denoted as TLB JGA) in Chapter 5 and Chapter 6 to assess local context understanding abilities.

Additionally, to provide a more meaningful evaluation, we also report F1 score on dialogue-level (DST F1) and turn-level (TLB F1) in Chapter 6. F1 scores are computed on a set of slot values, providing credit for partial matches. In contrast, JGA assigns a score only when all slot values are correct.

2.3 Computation Efficiency

Pretrained language models have made remarkable strides in advancing the state of natural language processing across various tasks. Additionally, the process of scaling up these language models has demonstrated an expansive augmentation of their capabilities, as elucidated in recent research [Wei et al., 2022a]. Such approaches incur substantial costs in terms of computational resources, memory requirements, and storage capacity. Consequently, there has been a concerted research effort aimed at mitigating these challenges.

2.3.1 Model Compression

One broad class of approaches to reduce computation is through model compression. This work primarily falls into three major paradigms: pruning, distillation, and quantization.

Pruning strategies are primarily devised to reduce computational overhead by selecting and retaining a subnetwork within a larger model [Fan et al., 2020; Michel et al., 2019; Wang et al., 2020; Lagunas et al., 2021; Xia et al., 2022]. In contrast, distillation techniques entail the training of a compact student model, to impart the knowledge and performance of a larger teacher model [Sanh et al., 2019; Turc et al., 2019; Jiao et al., 2020]. Finally, quantization methods aim to diminish memory demands by representing model parameters with fewer bits, thereby trading off a degree of precision for enhanced efficiency [Shen et al., 2020; Dettmers et al., 2022, 2024]. We use QLoRA [Dettmers et al., 2024] to efficiently fine-tune and perform inference with a Llama3 [Dubey et al., 2024] 8B model in CORRECTIONLM.

Note that the aforementioned methods are characterized as "static" in nature, as they primarily focus on the optimization of fixed model architectures for each data point. The routing framework (ORCHESTRALLM) introduced in this thesis adopts a dynamic perspective. Our framework is inspired by the recognition that examples can vary in terms of their complexity, and a single model may either overperform or underperform depending on the difficulty level. The model compression techniques are complementary to ORCHESTRALLM and CORRECTIONLM.

2.3.2 Sample-Adaptive Inference

To facilitate more nuanced control during the inference phase, researchers have explored techniques that enable variable levels of computational resources to be allocated for processing different input samples.

Two predominant categories of approaches have emerged in this domain: early exiting and token dropping.

Early exiting strategies typically incorporate additional classifiers at intermediate layers within a model. These classifiers play a crucial role in determining whether a given input example should terminate its processing prematurely and abstain from propagating to subsequent layers [Liu et al., 2020; Xin et al., 2021; Zhou et al., 2020]. On the other hand, token-dropping techniques aim to dynamically reduce the length of the input sequence, thereby enhancing computational efficiency. This is achieved by selectively excluding certain tokens from the input sequence, which are subsequently not passed on to subsequent layers in the model [Goyal et al., 2020; Guan et al., 2022; Kim and Cho, 2021]. Salehi et al. [2023] also direct different samples to sub-networks with varying widths. Our proposed routing framework, ORCHESTRALLM, also embraces sample-adaptive inference. However, it distinguishes itself by leveraging not just a single model but a combination of models, aiming to optimize inference for various samples by dynamically routing them to the most suitable expert within the ensemble.

Another line of research also leverages an SLM to reduce the computational costs of LLM. In Leviathan et al. [2023], motivated by the idea that not all inference steps (tokens) are equally challenging, they sample generations from an SLM as speculative prefixes for the LLM, which then evaluates and either accepts or rejects these speculations. Their approach assumes that the LLM is always superior to the SLM, whereas ORCHESTRALLM strategically utilizes the unique strengths of both SLM and LLM, depending on the specific characteristics of each dialogue turn.

2.3.3 Model Switching

Language models come in various types and sizes, with larger models typically exhibiting enhanced capabilities. It is well-established that not all input examples pose the same level of complexity to these models. Consequently, there has been a growing body of research focusing on the concept of model switching, wherein input examples are intelligently routed between small and large models based on their individual complexity levels. Our work draws inspiration from this line of research.

For instance, Madaan et al. [2023] propose a methodology that leverages an external meta-verifier to ascertain the correctness of predictions made by a Small Language Model (SLM) and to determine whether an example warrants routing to a Large Language Model (LLM). In contrast, our approach does not necessitate

the use of additional verifiers.

Another set of related approaches, exemplified by the work of Šakota et al. [2023]; Kag et al. [2022], involves training binary classifiers to categorize examples as suitable for SLM or LLM processing. This approach requires the router to be trained on labeled data where language models have made predictions. In contrast, ORCHESTRALLM dispenses with the requirement for a task-specific classifier by incorporating a retrieval method.

Chapter 3

SDP-DST: Dialogue State Tracking with a Language Model using Schema-Driven Prompting

3.1 Introduction

This chapter proposes a new prompt-based finetuning framework that achieves effective DST with a SLM. There are two broad paradigms of DST models, *classification-based* and *generation-based* models, where the major difference is how the slot value is inferred. In classification-based models [Ye et al., 2021; Chen et al., 2020], the prediction of a slot value is restricted to a fixed set for each slot, and non-categorical slots are constrained to values observed in the training data. In contrast, generation-based models [Wu et al., 2019; Kim et al., 2020] decode slot values sequentially (token by token) based on the dialogue context, with the potential of recovering unseen values. The first generation-based DST systems built on large-scale pretrained neural language models (LM), achieving strong results without relying on domain-specific modules. Among them, the autoregressive model [Peng et al., 2021; Hosseini-Asl et al., 2020] uses a uni-directional encoder whereas the sequence-to-sequence model [**?**Heck et al., 2020] represents the dialogue context using a bi-directional encoder.

In this chapter, we follow a generation-based DST approach using a pre-trained sequence-to-sequence

model, but with the new strategy of adding task-specific prompts as input for sequence-to-sequence DST models, inspired by *prompt-based* finetuning [Radford et al., 2019; Brown et al., 2020]. Specifically, instead of generating domain and slot symbols in the decoder, we concatenate the dialogue context with domain and slot prompts as input to the encoder, where prompts are taken directly from the schema. We hypothesize that jointly encoding dialogue context and schema-specific textual information can further benefit a sequence-to-sequence DST model. This allows task-aware contextualization for more effectively guiding the decoder to generate slot values.

Although the domain and slot names typically have interpretable components, they often do not reflect standard written English, e.g. "*arriveby*" and "*ref*". Those custom meaning representations are typically abbreviated and/or under-specified, which creates a barrier for effectively utilizing the pretrained LMs.

To address this issue, we further incorporate natural language schema descriptions into prompting for DST, which include useful information to guide the decoder. For example, the description of "*ref*" is "*reference number of the hotel booking*"; the values of "*has_internet*" are "*yes*", "*no*", "*free*", and "*don't care*".

In short, SDP-DST advanced generation-based DST in two ways. First, candidate schema labels were jointly encoded with the dialogue context, providing a task-aware contextualization for initializing the decoder. Second, natural language descriptions of schema categories associated with database documentation were incorporated in encoding as prompts to the language model, allowing uniform handling of categorical and non-categorical slots. When implemented using a pretrained sequence-to-sequence model, this simple approach improved SOTA results on MultiWOZ 2.2, and performance was on par with SOTA on MultiWOZ 2.1 and M2M. In addition, our analyses provided empirical results that contributed towards understanding how schema description augmentation can effectively constrain the model prediction.

This chapter contains material that was originally published in [Lee et al., 2021]. I was the main contributor of this work, and discussed and wrote the paper with the other collaborators throughout the project.

3.2 Approach

In this section, we first set up the notations that are used throughout this chapter, and then review the generative DST with the sequence-to-sequence framework. Based on that, we formally introduce SDP-

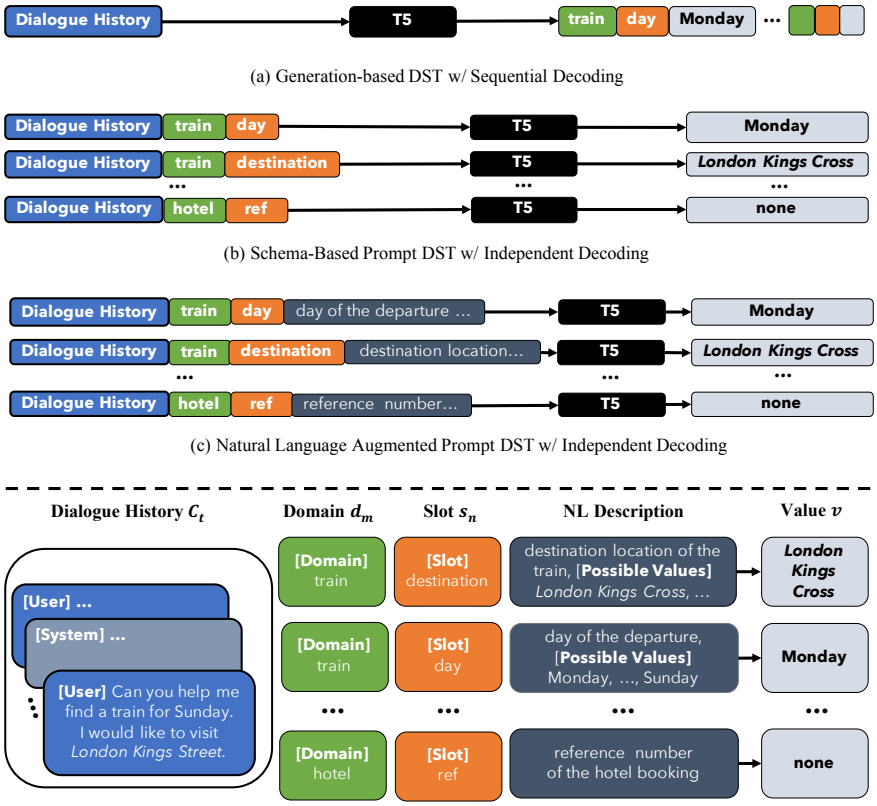

Figure 3.1: Overview of generative DST approaches for multi-domain scenario. The top three figures illustrate three different generative approaches considered in this chapter and the bottom figure includes specific examples for dialogue history, domain names, slot names, natural language descriptions (types, set of valid values, etc.) for slots. Sub-figure (b)(c) demonstrate two prompt-based DST models proposed, where method in (c) includes additional natural language description of slots considered for tracking. Domain descriptions are omitted for brevity.

DST and the corresponding backbone pretrained model.

Problem Definition

For task-oriented dialogues considered in this chapter, a dialogue consists of a sequence of utterances alternating between two parties, $U_1, A_1, ..., U_T, A_T$, where U and A represent the user utterance and the system response, respectively. In a turn t, the user provides a new utterance U_t and the system agent responds with utterance A_t. As shown in the bottom of Figure 5.1, at turn t, we denote the dialogue context as $C_t = \{U_1, A_1, \ldots, A_{t-1}, U_t\}$, which excludes the latest system response A_t. In this chapter, we assume a multi-domain scenario, in which case the schema contains M domains $\mathcal{D} = \{d_1, \ldots, d_M\}$ and N slots $\mathcal{S} = \{s_1, \ldots, s_N\}$ to track as examples illustrated in Figure 5.1. B_t, the dialogue state at turn t, is then

defined as a mapping from a pair (d_m, s_n) into values v. Here, we define $B_t(d_m, s_n) = \phi$, if (d_m, s_n) is not in the current dialogue state. In the given example of Figure 5.1, the pair (domain=*hotel*, slot=*ref*) is not in the dialogue state, and the value "*none*" is assigned.

3.2.1 Generation-based DST with the Sequence-to-sequence Model

There are primarily two decoding strategies for generation-based DST in the literature for inferring the dialogue state at a particular turn – sequential (a) and independent (b)(c) – both of which are explored in the paper as illustrated in Figure 5.1.

In the first case (top system (a) in Figure 5.1), the dialogue history C_t is taken as input to the encoder, and domain-slot-value triplets (d_m, s_n, v) are generated sequentially, where $B_t(d_m, s_n) \neq \phi$. This approach is adopted in many systems that leverage autoregressive LMs [Peng et al., 2021; Hosseini-Asl et al., 2020]. Despite being simple, this kind of sequential generation of multiple values is more likely to suffer from optimization issues with decoding long sequences resulting in lower performance. However, given its wide adoption in the literature, we still include this type of generative DST with the same backbone pretrained encoder-decoder Transformer model in our experiments. To partially address this issue, Lin et al. [2020] propose a domain-independent decoding where the decoder only has to generate a sequence of slot and value pairs within a specific given domain. Although their model leverages the same backbone model as ours, we empirically find that this form of strategy is still of limited effectiveness.

In the second case (middle two systems (b)(c) in Figure 5.1), the values for each domain-slot pair are generated independently, potentially in parallel. The domain and slot names (embedded as continuous representations) are either the initial hidden state of the decoder [Kim et al., 2020] or the first input of the decoder [Wu et al., 2019]. Values are either generated for all possible domain-slot (d_m, s_n) pairs with a possible value of "*none*" and/or there is a separate gating mechanism for domain-slot combinations not currently active. Since we are interested in enriching the input with task-specific information, we focus on extending the independent decoding modeling for our prompt-based DST.

3.2.2 SDP-DST

In this section, we formally present the flow of SDP-DST with an encoder-decoder architecture. Here, we are interested in an encoder-decoder model with a bi-directional encoder [Raffel et al., 2020; Lewis et al., 2020], in contrast with the uni-directional encoder used in autoregressive LMs [Radford et al., 2019; Brown et al., 2020].

The input of the prompt-based DST is made up of a dialogue context C_t (The sequence of user and system turns, U and A, respectively,) up to the current turn U_t) and a task-specific prompt. Here, we use two types of task-specific prompts, the domain-related prompt $X(d_m)$, and slot-related prompt $X(s_n)$, both of which are derived based on the given schema. We leave the discussion of two specific realizations of task-specific prompts to the later part of this section. Specifically, all sub-sequences are concatenated with special segment tokens, i.e., "`[user]` U_1 `[system]` $A_1 \ldots$ `[system]` A_{t-1} `[user]` U_t `[domain]` $X(d_m)$ `[slot]` $X(s_n)$", as input to the encoder, where `[user]`, `[system]`, `[domain]`, `[slot]` are special segment tokens for indicating the start of a specific user utterance, system utterance, domain-related prompt, and slot-related prompt, respectively.

Given this prompt-augmented input, the bi-directional encoder then outputs

$$H_t = \text{Encoder}(C_t, X(d_m), X(s_n)), \tag{3.1}$$

where $H_t \in \mathbb{R}^{L \times k}$ is the hidden states of the encoder, L is the input sequence length, and k is the encoder hidden size. Then, the decoder attends to the encoder hidden states and decodes the corresponding slot value $B_t(d_m, s_n)$:

$$B_t(d_m, s_n) = \text{Decoder}(H_t). \tag{3.2}$$

The overall learning objective of this generation processing is maximizing the log-likelihood of $B_t(d_m, s_n)$ given C_t, $X(d_m)$ and $X(s_n)$, that is

$$\sum_{(m,n)} \log P(B_t(d_m, s_n) | C_t, X(d_m), X(s_n)). \tag{3.3}$$

During inference, a greedy decoding procedure is directly applied, i.e., only the most likely token in the

given model vocabulary is predicted at each decoding step.

Schema-Based Prompt.

The first realization of task-specific prompt considered in this chapter is based on the domain and slot names as defined in the task-dependent schema. As shown in (b) of Figure 5.1, given the domain name *train* and the slot name *day*, the specific prompt is in the form of "`[domain]` *train* `[slot]` *day*". Different from [Lin et al., 2020; Wu et al., 2019] where the task-specific information is used in the decoder side, our symbol-based prompt as additional input to the bi-directional encoder can potentially achieve task-aware contextualizations. Observing that users often revise/repair their earlier requests in dialogues, we posit that the resulting encoded representations can be more effectively used by the decoder for generating corresponding slot values.

Natural Language Augmented Prompt.

One main drawback of symbol-based prompt is that those domain/slot names contain limited information that can be utilized by pretrained LMs. In other words, those symbols from the custom schema are typically under-specified and unlikely to appear in corpus for LM pretraining. Fortunately, documentation is commonly available for real-world databases, and it is a rich resource for domain knowledge that allows dialogue systems to better understand the meanings of the abbreviated domain and slot names. The documentation includes but is not limited to domain/slot descriptions and the list of possible values for categorical slots. In this chapter, we experiment with a simple approach that augments the input by incorporating the domain description after the domain name and the slot description (with the sequence of values, if any) following the slot name, as illustrated in the system (c) in Figure 5.1.

Backbone Sequence-to-sequence Model

Our prompt-based DST model is initialized with weights from a pretrained LM in an encoder-decoder fashion. In this chapter, we use the Text-to-Text Transformer (T5) [Raffel et al., 2020] as our backbone model. T5 is an encoder-decoder Transformer with relative position encodings [Shaw et al., 2018]. We refer interested readers to the original paper for more details.

Dataset	MWOZ 2.2	MWOZ 2.1	M2M
# Domains	8	8	2
# Dialogues	10438	10438	3008
# Total Turns	143004	143048	27120
Avg. Turns per Dial.	13.70	13.70	9.01
Avg. Toks per Turn	13.23	13.18	8.28
# Cat. Slots	21	0	0
# Non-Cat. Slots	40	37	12
Domain Desc.	Y	N	N
Slot Desc.	Y	Y	N
Value Set	Y	N	N

Table 3.1: Experiment data summary. The numbers are computed on all splits of the datasets. `MWOZ` stands for MultiWOZ. `Cat. Slots` and `Non-Cat. Slots` stand for categorical slots and non-categorical slots, respectively. The rows `Domain Desc.` and `Slot Desc.` indicate whether the corresponding dataset has natural language description for domains and slots, respectively. The row `Value Set` incates whether the corresponding dataset provides possible value set for categorical slots.

3.3 Experiments

3.3.1 Datasets

Table 5.1 summarizes the statistics of the datasets used in our experiments.

MultiWOZ [Budzianowski et al., 2018] is a multi-domain task-oriented dialogue dataset that contains over 10K dialogues across 8 domains. It is a collection of human-human written conversations and has been one of the most popular benchmarks in the DST literature. Since its initial release, many erroneous annotations and user utterances have been identified and fixed in subsequent versions, i.e., MultiWOZ 2.1 [Eric et al., 2019] and MultiWOZ 2.2 [Zang et al., 2020]. In addition, MultiWOZ 2.1 provides 2-3 descriptions for every slot in the dataset. We randomly sample one of them and use the same descriptions for every experiment. The original dataset does not have domain descriptions and possible values so these are omitted in the corresponding experiments. MultiWOZ 2.2 further provides descriptions of domain and slot as well as possible values for categorical slots.

Machines Talking To Machines (M2M) [Shah et al., 2018b] is a framework that combines simulation and online crowdsourcing. Templates of each dialogue are first generated and then online workers rewrite the conversations to make them human-readable while preserving the meaning. It provides 3,000 dialogues spanning 2 domains. The restaurant domain is denoted as `Sim-R` and the movie domain is denoted as

Sim-M. Since there are no descriptions provided in the corpus, we take existing descriptions from other corpora that have the same slots. Specifically, descriptions for the restaurant domain are taken from MultiWOZ 2.2, whereas descriptions for the movie domain are taken from SGD [Rastogi et al., 2020b]. All slots in M2M are covered. Since all slots are non-categorical, the descriptions do not include the possible values.

Evaluation Metric. The standard dialogue state tracking joint goal accuracy (DST JGA) is used as the evaluation metric. For MultiWOZ 2.1 and 2.2, we use the official evaluation script from the DSTC8 challenge [Rastogi et al., 2020a].[1] For M2M, we adopt the above evaluation scripts with simple modifications.

3.3.2 Implementation Details and Schema Descriptions

The backbone models we use for finetuning are T5-small (60M parameters) and T5-base(220M parameters). We use the pretrained checkpoint from *transformers* library[2]. For T5-small, we train the model with a batch size of 4, a learning rate of 5e-5 for 3 epochs. For T5-base, we train the model with a batch size of 64, a learning rate of $5e{-}4$ for 2 epochs. Both models are trained using Adam[Loshchilov and Hutter, 2019]. We don't use any text or label normalization scripts like [Wu et al., 2019; Hosseini-Asl et al., 2020].

For MultiWOZ 2.1 and 2.2, following many previous works[Wu et al., 2019], since *police* and *hospital* domains only appear in the training set, we exclude them in all our experiments.

Descriptions We show the descriptions of M2M and MultiWOZ 2.1in Table3.2 and Table3.3

3.3.3 MultiWOZ 2.2: Fully Annotated Natural Language Augmented Prompt

We present the evaluation results on MultiWOZ 2.2 in Table 3.4. The following baseline models are considered: TRADE [Wu et al., 2019], DS-DST [Zhang et al., 2020] and Seq2Seq-DU [Feng et al., 2020]. Similar to ours, the decoding strategy of TRADE is independent. However, the sum of domain and slot embeddings are the first input of the decoder, which makes their dialogue history representation not task-aware contextualized. The sequential decoding strategy is worse than the independent decoding strategy by over 5% with both T5-small and T5-base. Even with T5-small (almost half the model size of BERT-base which is used in most previous benchmark models), our system achieves the SOTA performance using the

[1] https://github.com/google-research/google-research/tree/master/schema_guided_dst#evaluation-on-multiwoz-21

[2] https://huggingface.co/t5-small ,https://huggingface.co/t5-base

Table 3.2: Domain and slot descriptions of M2M used in our experiments. The descriptions of the movie domain is taken from [Rastogi et al., 2020a] and the descriptions of the restaurant domain is taken from [Zang et al., 2020].

Sim-M			
Domain	**Domain Description**	**Slot**	**Slot Description**
Movie	A go-to provider for finding movies, searching for show times and booking tickets	theatre_name	the name of the theatre where the movie is playing
		movie	name of the movie
		date	date of the show booking
		time	time of the show booking
		num_people	number of people to purchase tickets for
Sim-R			
Domain	**Domain Description**	**Slot**	**Slot Description**
Restaurant	find places to dine and whet your appetite	price_range	price budget for the restaurant
		location	the location or area of the restaurant
		restaurant_name	the name of the restaurant
		category	the cuisine of the restaurant you are looking for
		num_people	how many people for the restaurant reservation
		date	date of the restaurant booking
		time	time of the restaurant booking

independent decoding. As expected, T5-base systems outperform T5-small systems. With the augmentation of descriptions, we improve the overall JGA by over 1% in both T5-small and T5-base.

3.3.4 MultiWOZ 2.1: Partially Annotated Natural Language Augmented Prompt

Different from MultiWOZ 2.2 studied in the previous section, MultiWOZ 2.1 only contains natural language descriptions for slots but not domains. In addition, there is no possible slot value information.

The evaluation results on MultiWOZ 2.1 are shown in Table 3.5, where we compare with TRADE [Wu et al., 2019], MinTL [Lin et al., 2020], SST [Chen et al., 2020], TripPy [Heck et al., 2020], Simple-TOD [Hosseini-Asl et al., 2020], SOLOIST [Peng et al., 2021] and TripPy+SCORE [Yu et al., 2020]. Note that both SOLOIST and TripPY+SCORE use external dialogue datasets to finetune their models.

As expected, we observe that T5-base models perform consistently better than T5-small models. Moreover, using descriptions consistently improves the performance of both models. All our models outperform baselines that do not use extra dialogue data. It is worth noting that comparing with MinTL (T5-small), our model is better by over 4% even without descriptions. Further, our T5-small system is even better than MinTL built on BART-LARGE [Lewis et al., 2020] which has substantially more parameters. Similar to ours, MinTL leverages a sequence-to-sequence LM. One difference is that their domain information is fed only to the decoder while our approaches enables task-aware contextualization by prompting the LMs with domain and slot information on the encoder side. Another difference is that they jointly learn DST

Table 3.3: The randomly sampled descriptions of MultiWOZ 2.1 used in all our experiments.

MultiWOZ 2.1		
Domain	**Slot**	**Slot Description**
taxi	leaveat	what time you want the taxi to leave your departure location by
taxi	destination	destination of taxi
taxi	departure	what place do you want to meet the taxi
taxi	arriveby	when you want the taxi to drop you off at your destination
restaurant	book people	number of people booking the restaurant
restaurant	book day	what day of the week to book the table at the restaurant
restaurant	book time	time of the restaurant booking
restaurant	food	food type for the restaurant
restaurant	pricerange	price budget for the restaurant
restaurant	name	name of the restaurant
restaurant	area	preferred location of restaurant
train	destination	destination of the train
train	day	what day you want to take the train
train	departure	departure location of the train
train	arriveby	what time you want the train to arrive at your destination station by
train	book people	number of people booking for train
train	leaveat	when you want to arrive at your destination by train
hotel	pricerange	preferred cost of the hotel
hotel	type	type of hotel building
hotel	parking	parking facility at the hotel
hotel	book stay	length of stay at the hotel
hotel	book day	day of the hotel booking
hotel	book people	how many people are staying at the hotel
hotel	area	rough location of the hotel
hotel	stars	rating of the hotel out of five stars
hotel	internet	whether the hotel has internet
hotel	name	which hotel are you looking for
attraction	type	type of attraction or point of interest
attraction	area	area or place of the attraction
attraction	name	name of the attraction

together with dialogue response generation, which provides more supervision signals. Therefore, the better performance of our systems implies that schema-driven prompting is effective.

3.3.5 M2M: Borrowed Natural Language Augmented Prompt

Table 3.6 shows the evaluation results on M2M. In this case, all natural language descriptions are directly borrowed from dialogue datasets that are annotated in a different manner. We achieve the SOTA performance on Sim-Restaurant and Sim-Movie+Restaurant while being comparable on Sim-Movie. The improvements of descriptions are only evident on the restaurant domain. The lack of improvement from slot descriptions for the movie domain may be because the slot descriptions do not add much beyond the slot name (compared

Models	Pretrained-Model/ # Para.	DST JGA
TRADE	N	48.6
DS-DST	BERT-base / (110M)	51.7
Seq2Seq-DU	BERT-base / (110M)	54.4
Sequential	T5-small / (60M)	48.9
Sequential	T5-base / (220M)	51.2
Independent	T5-small / (60M)	55.2
w. desc	T5-small / (60M)	56.3
Independent	T5-base / (220M)	56.7
***w.* desc**	T5-base / (220M)	**57.6**

Table 3.4: Results on MultiWOZ 2.2. All numbers are reported in dialogue state tracking joint goal accuracy (DST JGA)(%). *w.* desc means the model is trained with the description. # Para. stands for the number of model parameters.

to "category" for the restaurant domain) or that it has many slots that generalize across domains (e.g. date, time, number of people).

3.4 Analysis

3.4.1 Breakdown Evaluation for MultiWOZ

In Table 3.7, we follow the categorization provided in [Zang et al., 2020] and show the breakdown evaluation of categorical and non-categorical slots on MultiWOZ 2.2. We compute JGA on the collection of categorical slots and non-categorical slots separately for each turn and then average over turns. As we can see, the breakdown accuracy scores for both categorical and non-categorical slots are pretty consistent with the overall JGA. For both T5-small and T5-base models, models with sequential decoding perform worse than the corresponding models with independent decoding for both categorical and non-categorical slots. In particular, the independent decoding models achieve more pronounced improvement in categorical slots indicating that the task-specific prompt is very helpful for guiding the decoder to predict valid values. When comparing models using natural language description with those not, we observe performance gains for both types of slots for T-base but only non-categorical slots for T5-small. It is likely that the smaller size of T5 has limited representation capability to effectively utilize the additional textual description information regarding types and possible values.

Models	Pretrained-Model / # Para.	DST JGA
TRADE	N	45.60
MinTL	T5-small / (60M)	50.95
MinTL	BART-large / (406M)	53.62
SST	N	55.23
TripPy	BERT-base / (110M)	55.29
Simple-TOD[3]	GPT2 / (117M)	55.72
*SOLOIST	GPT-2 / (117M)	56.85
***TripPy + SCORE**	ROBERTA-large / (355M)	**60.48**
Independent	T5-small / (60M)	55.37
w. desc	T5-small / (60M)	56.12
Independent	T5-base / (220M)	56.39
***w.* desc**	T5-base / (220M)	**56.66**

Table 3.5: Results on MultiWOZ 2.1. All numbers are reported in dialogue state tracking joint goal accuracy (DST JGA)(%). *w.* desc means the model is trained with the description. * means extra dialogue data is used to finetune the language model. # Para. stands for the number of model parameters.

Models	Sim-M	Sim-R	Sim-M+R
[Rastogi et al., 2017]	96.8	94.4	–
[Rastogi et al., 2018]	50.4	87.1	73.8
[Chao and Lane, 2019]	80.1	89.6	–
[Heck et al., 2020]	**83.5**	90.0	–
SDP-DST	83.3	89.6	**88.0**
w. desc	81.0	**90.6**	86.4

Table 3.6: Results on M2M. All numbers are reported in dialogue state tracking joint goal accuracy (DST JGA)(%). [Rastogi et al., 2017] should be considered as a kind of oracle upper bound performance because target slot value is guaranteed to be in the candidate list and considered by the model.

3.4.2 Ablation Study on Schema Descriptions

To understand what parts of the schema descriptions are most important, we experiment with three kinds of description combinations on MultiWOZ 2.2 using the T5-small configuration: **(i)** excludes the list of possible values for categorical slots **(ii)** excludes slot descriptions **(iii)** excludes domain descriptions. For **(i)**, there is an 0.4% point drop in JGA, validating that value sets can successfully constrain the model output, as we illustrate in Table 3.8. For **(ii)**, there is a 0.8% point drop in JGA. And for **(iii)**, there is a 0.1% point drop in JGA. This shows that slot descriptions are the most important part of the schema prompts and domain descriptions are relatively less effective. This is probably due to the fact that there are 61 slots in MultiWOZ 2.2 but only 8 domains. Also, the domain names are all self-contained single words.

Models	DST JGA	CAT	NON-CAT
Sequential (T5-small)	48.9	61.3	69.0
Sequential (T5-base)	51.2	62.9	70.9
Independent (T5-small)	55.2	71.4	75.2
w. desc	56.3	71.1	76.2
w. only slot desc	55.2	70.4	75.8
w. only domain desc	54.3	70.1	75.4
w. only slot + domain desc	55.9	71.2	76
Independent (T5-base)	56.7	71.6	76.3
w. desc	57.6	72.4	76.8

Table 3.7: Slot type breakdown results on the test set of MultiWOZ 2.2. All numbers are reported in dialogue state tracking joint goal accuracy (DST JGA) (%). CAT and NON-CAT correspond to categorical slots DST JGA and non-categorical slots DST JGA, respectively. *w.* desc indicates that the model is trained with the full description.

3.4.3 The Effectiveness of Natural Language Augmented Prompt

In order to understand the benefit of natural language augmented prompt, we focus on analyzing the examples where the description augmented model correctly tracks the dialogue state while the unaugmented one fails. Based on our analysis of T5-base model on MultiWOZ 2.2, the most common errors are either misses of gold slots or over-predictions of irrelevant slots (82.8% of all errors). The remaining error cases are correct slot predictions with wrong slot values (17.2%).

We provide representative examples for which the description augmented system correctly tracks the dialogue states but not the unaugmented one in Table 3.8. In the first example, the phrases in the dialogue history are partially matched to the slot description of *arriveby* making it easier for the description-augmented system to detect the mention of the correct slot. For the second example, the type information in the description implicitly guides the model to focus on time-related information leading the correct output of the normalized time expression, 16:45. In contrast, the model without descriptions only generates the partial answer 4:45, ignoring PM. Lastly, "London Kings Street" is a typographical error in this case. By utilizing the provided possible values included in the slot descriptions, the model is able to generate the correct slot value without spelling error, demonstrating that the natural language augmented prompt can successfully constrain the model output and potentially provides robustness to the dialogue state tracking system.

Database *Train*	**Slot Descriptions ‖ Possible Values**
arriveby	**arrival time of the train**
destination	destination of the train ‖ Birmingham New Street, **London Kings Cross**, ..., Stevenage
Dialogue History	... *[SYS]* The earliest being 19:09 and arriving by 20:54. Would that work for you? *[USR]* Yes, I think the **20:54 arrival time** should work.
no desc.	*(train, day, friday) (train, departure, leicester) (train, destination, cambridge) (train, leaveat, 19:00)*
desc.	***(train, arriveby, 20:54)*** *(train, day, friday) (train, departure, leicester) (train, destination, cambridge) (train, leaveat, 19:00)*
Dialogue History	*[USER]* I need to find a train going to Leicester that arrives by **4:45 PM**. Do you know of one?
no desc.	*(train, arriveby, 04:45) (train, destination, leicester)*
desc.	***(train, arriveby, 16:45)*** *(train, destination, leicester)*
Dialogue History	*[USER]* Can you help me find a train for Sunday. I would like to visit **London Kings Street**.
no desc.	*(train, destination, London Kings Street) (train, day, Sunday)*
desc.	***(train, destination, London Kings Cross)*** *(train, day, Sunday)*

Table 3.8: Examples for `train` domain dialogues where the description-augmented ("desc.") model make the correct state predictions but the unaugmented models ("no desc.") fails. The correctly predicted triplets are in bold.

	53.33%: Annotation Errors
Dialogue History	...*[SYSTEM]* Out of the 21 restaurant choices, one is the **Yippee Noodle Bar which is moderately priced in the centre of town**. Would you like to make a reservation? *[USER]* **That sounds great**, what is the postcode?
Gold	*()*
desc. Prediction	*(restaurant, area, centre) (restaurant, pricerange, moderate) (restaurant, name, yippee noodle bar*
	20.00%: Unable to Capture System Information
Dialogue History	... *[SYSTEM]* There is TR6679. **It leaves at 19:35 and arrives at 19:52**. Is that good for you? *[USER]* **Sounds good**. May I have the travel time and ticket price, please?
Gold	*(train, arriveby, 19:52) (train, leaveat, 19:35)*
desc. Prediction	*()*
	16.66%: Unable to Mention Slot Provided by User
Dialogue History	... *[USER]* Do you happen to know if there is a nightclub in the centre? *[SYSTEM]* Yes, we have FIVE nightclubs in the centre of town. Is there a particular one you're looking for? *[USER]* **I don't care which one you recommend**, but can you tell me the entrance fee and address?
Gold	*(attraction, area, centre) (attraction, type, nightclub)* **(attraction, name, dontcare)**
desc. Prediction	*(attraction, area, centre) (attraction, type, nightclub)*
	10.00%: Incorrect Value Reference
Dialogue History	*[USER]* Hi can you help me find a very nice Italian restaurant near the centre of cambridge? *[SYSTEM]* Please specify your price range. *[USER]* **It does not matter**.
Gold	*(restaurant, area, centre) (restaurant, food, italian)* ***(restaurant, pricerange, dontcare)***
desc. Prediction	*(restaurant, area, centre) (restaurant, food, italian)* ***(restaurant, pricerange, expensive)***

Table 3.9: The most common error types of our best model(t5-base *w/ desc.*) and corresponding examples.

3.4.4 Error Analysis of Natural Language Augmented Prompt-based DST

Here, we further carry out error analyses into the natural language augmented prompt-based T5-base model on MultiWOZ 2.2. As shown in Table 3.9, we randomly sample 50 turns and categorize them into different types. In summary, there are four types of errors: **(i)** The most common error type is annotation error in which the model prediction is actually correct, which is similar to the findings of [Zhou and Small, 2019]. **(ii)** 20% of the errors come from model failing to capture information provided by the system.[4] **(iii)** 16.66% of the errors are caused by the model misses of at least one gold slot. **(iv)** 10% of the errors are correct slot predictions with the wrong corresponding values. In general, most errors are likely caused by the lack of explicit modeling of user-system interactions.

3.5 Summary

In this work, we proposed a simple but effective task-oriented dialogue system based on large-scale pre-trained LM. We showed that, by reformulating the dialogue state tracking task as prompting knowledge from LM, our model can benefit from the knowledge-rich sequence to sequence T5 model. Based on our experiments, the proposed natural language augmented prompt-based DST model improved SOTA on MultiWOZ 2.2 and had comparable performance on MultiWOZ 2.1 and M2M to recent SOTA models. Moreover, our analyses provided evidence that the natural language prompt is effectively utilized to constrain the model prediction.

[4]There is label inconsistency in the MultiWoZ as pointed out by [Zhou and Small, 2019]. If the user confirms the booking or gives a positive response, then the dialogue states in the previous system utterance should be grounded. However, this rule is not always followed in the dataset construction. So to some extent, this type of error is inevitable.

Chapter 4

IC-DST: In-Context Learning for Few-Shot Dialogue State Tracking

4.1 Introduction

In this chapter, we propose an in-context learning framework for DST building on LLM. Collecting and annotating turn-level dialogue states is notoriously hard and expensive [Budzianowski et al., 2018]. Also, in commercial applications, it is common to extend the schema and incorporate new domains. Thus, it is important to develop DST learning strategies that are flexible and scalable, in addition to requiring less data.

Previous studies have explored zero/few-shot DST, but with some limitations. Most few-shot methods are based on finetuning pretrained language models [Wu et al., 2020b; Li et al., 2021; Su et al., 2022; Shin et al., 2022; Lin et al., 2021b; Xie et al., 2022]. Such systems are less flexible, since they need to be retrained when new slots or domains are added, and finetuning large LMs is computationally expensive. Most zero-shot methods have involved domain-transfer approaches [Hosseini-Asl et al., 2020; Lin et al., 2021b,a], which have not yielded good performance.

To address the above challenges, we propose the IC-DST model to solve the DST problem with the in-context learning (ICL) paradigm [Brown et al., 2020], in which a large language model makes predictions based on the task instruction and/or examples in the prompt. In few-shot settings, the prompt contains exemplars that are retrieved from a small set of labeled training data. A motivation behind this framework

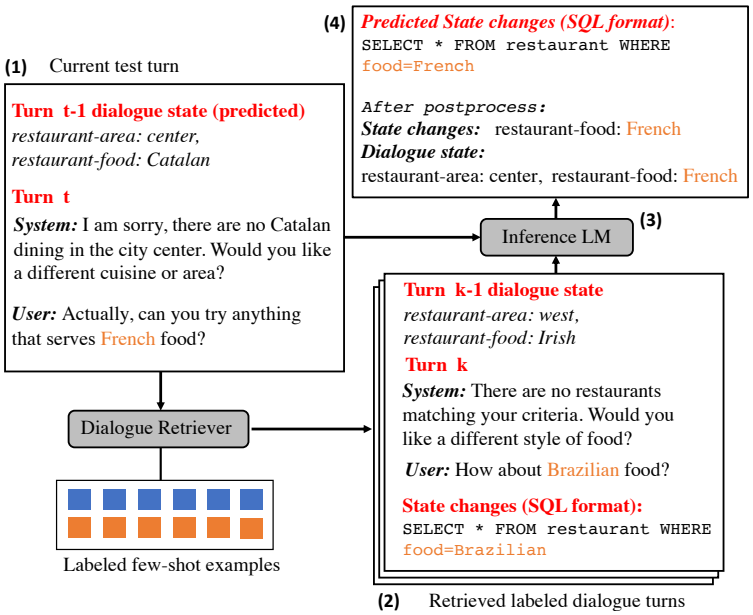

Figure 4.1: Illustration of DST task and IC-DST approach. The task is to track the slot values associated with a user request up to the current turn (dialogue state). In few-shot settings, given a test turn (1), IC-DST first retrieves a few most similar turns from the labeled dialogues as examples (2). The task schema (not shown in the figure), examples, and the test dialogue turn are concatenated in the prompt to a LM (e.g. GPT3) (3) to produce the current turn dialogue state changes as a SQL query (4).

is that it requires no finetuning (i.e., no parameter updates), which makes systems flexible in that they can handle queries in a new domain via the exemplar retrieval process *without re-training*. This enables developers to quickly prototype systems in new domains and rapidly leverage new collected data. ICL has been used successfully in semantic parsing [Rajkumar et al., 2022; Pasupat et al., 2021; Rubin et al., 2022], especially in few-shot scenarios. However, these studies focus on sentence-level tasks. ICL has been explored for DST [Madotto et al., 2021; Xie et al., 2022], but the performance fell short of pretraining and domain-transfer approaches to few/zero-shot learning. DST involves long, two-party dialogue histories with grounding in a structured ontology. We believe these challenges cause the poor ICL performance on DST tasks in previous work.

To address these challenges, we explore in-context learning with three novel contributions. First, we reformulate DST as a text-to-SQL task, including a tabular description of the ontology in the prompt. This is a better match to the knowledge-grounded scenario, and it takes advantage of large language models pre-trained with code: Codex [Chen et al., 2021b], GPT-Neo [Black et al., 2021], and CodeGen [Nijkamp et al., 2022]. Second, we use the dialogue state in representing context, rather than the full conversation history,

which is more efficient and better suited to domain changes. Lastly, in the few-shot scenario, we propose a new approach to learning a similarity score for selecting in-context examples that is trained to match similarity based on dialogue state changes. The IC-DST approach, which incorporates these advances, achieves a new state of the art on MultiWOZ few-shot settings, i.e. when using 1–10% training data. A further contribution is an extensive analysis demonstrating impact from each innovation.

In summary, IC-DST makes the following contributions. To our knowledge, we are the first to successfully apply in-context learning for DST, building on a text-to-SQL approach. To extend in-context learning to dialogues, we introduce an efficient representation for the dialogue history and a new objective for dialogue retriever design. Our system achieves a new state of the art on MultiWOZ in few-shot settings. We provide insights into how in-context learning works for dialogue, including the importance of good in-context examples and the LM's ability to generalize beyond examples.

This chapter includes material originally published in [Hu et al., 2022]. As the second main contributor to this work, I collaborated closely with the other authors throughout the project. My contributions included establishing the initial version of the workable framework, proposing the idea of reformulating DST as a text-to-SQL task, and conducting a subset of experiments.

4.2 DST System Design

4.2.1 DST Task Framing

Notation A task-oriented dialogue consists of a sequence of utterances alternating between the user and the system, $A_1, U_1, ..., A_T, U_T$, where A and U represent the system and user utterances, respectively. The task of DST is to predict the dialogue state y_t at each turn t,[1] given the dialogue context $C_t = [A_1, U_1, \cdots, A_t, U_t]$, where y_t is a set of slot-value pairs:

$$y_t = \{(s_i^t, v_i^t); i = 1, \ldots, n_t\}.$$

The set of possible slots s_i is given in a pre-defined schema. The schema can contain multiple domains, where a "domain" corresponds to a backend capability such as hotel or restaurant booking. Each domain is

[1]For brevity, "turn" is used to mean a pair of system and user turns (A_t, U_t), associated with state update intervals.

associated with a set of slots; for example, the 'hotel' domain has slots 'hotel-name', 'hotel-price_range,' 'hotel-area', etc. Each observed slot is associated with a value, which may have pre-defined categories (e.g., 'hotel-price_range' may be 'cheap,' 'moderate,' or 'expensive') or open (e.g. 'hotel-name'). We focus on a multi-domain scenario in this work, in which dialogue states may contain slots from multiple domains.

One popular way to generate the dialogue states for the current turn is finetuning auto-regressive language models [Hosseini-Asl et al., 2020; Peng et al., 2021]. For each turn, the model takes the dialogue context C_t as input, and generates a sequence of slot-value pairs (s_i, v_i) sequentially. Equivalently, one can generate a sequence of slot-value pair dialogue state changes.

Dialogue states vs. state changes Dialogues can be lengthy and complex, resulting in dialogue states that can include several slots and values, which means that coverage of the possible states is sparse for few-shot learning. However, the dialogue state change from one turn to the next typically involves a small number of slots. For that reason, we use state *changes* at each turn as a label for prediction. The concept of state changes is illustrated in Figure 5.1. Possible state changes include slot addition, slot deletion, and slot value change. For example, in the current test turn of Figure 5.1, the user asks for Catalan food in turn $t-1$, and changes it to French food in turn t. The state change updates the dialogue state by replacing 'Catalan' with 'French'. Specifically, given the previous turn dialogue state y_{t-1} and the predicted current turn state changes c_t, we update the dialogue state by first copying y_{t-1} to y_t, and then executing add, delete and change operations according to each slot-value pair (s_i, v_i) in c_t. Our analysis in Section 4.5 shows that using state changes leads to substantial improvements.

DST as Text-to-SQL Here we propose a new representation for dialogue states: SQL. This is inspired by the fact that dialogue states are used to determine how to query backend databases for the information users need. Our representation follows three rules: (1) each domain is defined as a table and each slot is defined as a column; (2) all the slots and values are in the WHERE clause; and (3) for turns with multiple domains, we rename each domain to $d_1, ..., d_m$. Using the SQL state representation with a generative LM, DST becomes a Text-to-SQL problem. This approach is facilitated by language models pretrained with code (Codex and GPT-Neo) as SQL is closer to the code used for pre-training.

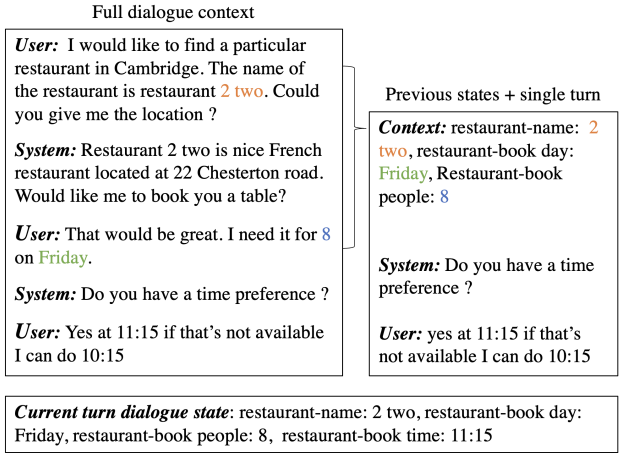

Figure 4.2: Illustration of the dialogue context representation: the full dialogue context C_{t-1} before the current turn is replaced by the associated dialogue state y_{t-1}.

Dialogue context representation Previous work generally represents dialogue context by concatenating all the system and user utterances $A_1, U_1, \cdots, A_t, U_t$ [Lee et al., 2021; Lin et al., 2021b; Peng et al., 2021]. However, real world dialogues can be lengthy, and there is a length limit for current large language models (2048 tokens for GPT-3, 4096 tokens for Codex). It is not practical to represent the full dialogue context for multiple exemplars in the prompt. A simple solution is to just include the N recent turns in the dialogue history [Lei et al., 2018; Budzianowski and Vulić, 2019; Wu et al., 2021]. We adopt a new approach that takes advantage of the fact that the dialogue state is a summary of the dialogue history, as shown in Figure 4.2. Specifically, we represent dialogue context by $[y_{t-1}, A_t, U_t]$, in which y_{t-1} is the accumulated dialogue state after user turn $t-1$.

4.2.2 In-Context Learning

In-context learning is an alternative to finetuning that keeps pretrained language model parameters fixed [**?**]. The language model takes a prompt P that contains task descriptions, in-context examples, and the test instance as input, and predicts the label by capturing the patterns in the context. ICL has two advantages over finetuning. First, it avoids the need for repeated finetuning when the schema is updated or new examples are added. This is particularly important for large models like GPT-3, since finetuning at this scale is extremely expensive. Second, by simply adding/removing training examples, in-context learning enables us to quickly manipulate the model's predictions and correct mistakes without re-training.

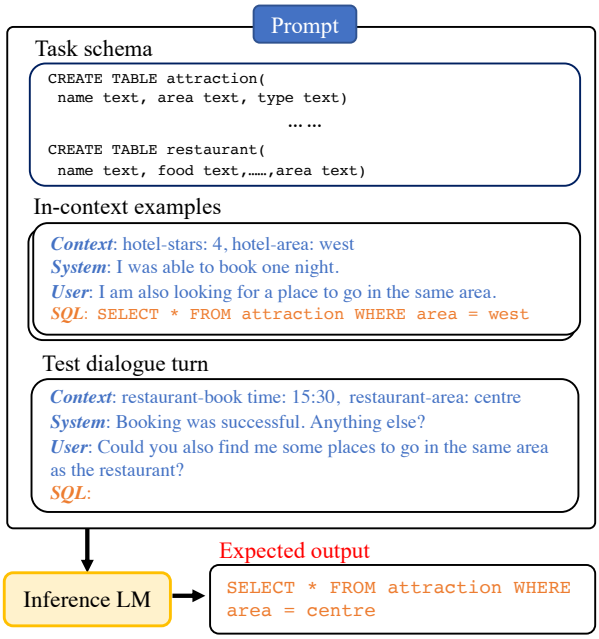

Figure 4.3: The prompt contains the task schema, in-context examples, and the current test dialogue turn.

An overview of our IC-DST system is shown in Figure 5.1 for the few-shot setting. The details of our prompt are shown in Figure 4.3. Examples of prompts are shown in Appendix A.1.1. The task description is the schema associated with the task ontology, and a retriever is used to select labeled example turns from the training data.

Schema prompting We use an SQL table for each domain to represent the dialogue schema in the prompt. Each table includes a row of slot names followed by three rows of example values associated with each slot, as illustrated in Figure 4.4. Slots like “restaurant-name” or "restaurant-book time" typically have many possible values. Thus, for these slots, we only list a few example values. In our experiments, we create SQL tables for all domains and concatenate them to be part of our input.

In-context examples In the few-shot scenario, a retriever takes the dialogue context as input (either $[C_{t-1}, A_t, U_t]$ or $[y_{t-1}, A_t, U_t]$) and retrieves similar example contexts from the labeled training set. Advantages of using the dialogue state y_{t-1} rather than the full history C_{t-1} are that it is shorter (allowing for more examples) and it leads to a more effective retrieval similarity score.

```
CREATE TABLE taxi(
destination text, departure text,
leaveat text, arriveby text)
/*
3 example rows:
SELECT * FROM taxi LIMIT 3;
destination     departure    leavat     arriveby
avalon          allenbell    14:45      15:30
camboats        la mimosa    dontcare   15:45
curry garden    golden wok   11:45      dontcare
*/
```

Figure 4.4: The SQL table for the taxi domain schema. We follow the table prompt from Rajkumar et al. [2022].

4.2.3 Dialogue Retriever

In few-shot settings, the success of in-context learning relies heavily on the quality of the context examples. Typically, this is achieved through semantic retrieval using the testing input as the query. Previous studies have focused on building sentence-level retrievers, but our work extends this to retrieving entire dialogue histories.

One approach involves using a pretrained embedding model with a cosine similarity score as the retriever, which does not require any DST-specific data. For each test example, we retrieve the k training examples that are the nearest neighbors of the test context based on the cosine similarity of their embeddings. We experimented with RoBERTa, SBERT, and BM25, finding that SBERT produced the best results.

Our goal is to retrieve example dialogue contexts that are relevant to the predicted state change of a test sample. Therefore, we also finetuned SBERT on the few-shot examples to better align the retriever with the objective of predicting state changes. We defined the similarity between state changes using slot and slot-value pair similarities, and trained the embedding model using a contrastive loss to ensure high similarity for positive example pairs and low similarity for negative example pairs. This finetuned retriever improved the relevance of retrieved examples, thereby enhancing the overall performance of the DST system.

For more details about the dialogue retriever, please refer to Hu et al. [2022].

Model	MultiWOZ 2.1 1%	5%	10%	100%	MultiWOZ 2.4 1%	5%	10%	100%
Baselines (finetuned)								
TRADE [Wu et al., 2020b]	12.58	31.17	36.18	46.00	-	-	-	55.05
SDP-DST [Lee et al., 2021]	32.11	43.14	46.92	56.66	-	-	-	-
DS2 - BART [Shin et al., 2022]	28.25	37.71	40.29	46.86	30.55	42.53	41.73	46.14
DS2 - T5 [Shin et al., 2022]	33.76	44.20	45.38	52.32	36.76	49.89	51.05	57.85
In-Context Learning								
IC-DST GPT-Neo 2.7B	16.70	26.90	31.65	39.18	17.36	29.62	34.38	45.32
IC-DST CodeGen 2.7B	20.72	29.62	33.81	39.93	21.87	33.16	37.45	45.71
IC-DST Codex-davinci	**43.13**	**47.08**	**48.67**	50.65	**48.35**	**55.43**	**56.88**	62.43

Table 4.1: Multi-domain DST JGA evaluated on MultiWoz 2.1 and 2.4 using 1%, 5%, 10%, and 100% of the training set. The average results of 3 runs are reported. DS2-T5 [Shin et al., 2022] is the previous state-of-the-art few-shot DST model on MultiWOZ.

4.3 Experiments

4.3.1 Datasets

MultiWOZ [Budzianowski et al., 2018] is a multi-domain human-human written dialogue dataset that contains over 10K dialogues across 8 domains. The labels and utterances have been refined in subsequent versions, e.g., MultiWOZ 2.1 [Eric et al., 2019] and MultiWOZ 2.4 [Ye et al., 2022]. MultiWOZ 2.4 is built on top of the 2.1 version and made substantial changes to the validation and testing sets. It can be viewed as a cleaner version of MultiWOZ 2.1 that better reflects model performance. In general, DST results are higher on 2.4 when compared to 2.1.[2] Following the previous work [Wu et al., 2019], we use 5 domains: hotel, taxi, attraction, restaurant, and train. There are 8438 dialogues in the training set, and 1000 dialogues in the dev and test set. On average, there are 13.46 turns per dialogue and 13.13 tokens per turn. For preprocessing, we use the scripts of Ye et al. [2022]. This script mainly fixes typos and standardizes the formatting. All data are downloadable from Ye et al. [2022].

4.3.2 Baselines

TRADE [Wu et al., 2019] An encoder-decoder framework is applied to the DST problem, enabling generalization to unseen values and domains. This was the first work to explore cross-domain transfer in DST. Different from IC-DST, TRADE has to make a prediction for each domain and slot pair in separate passes.

[2]Many prior few-shot DST studies report results on MultiWOZ 2.1; we include this older version for direct comparisons.

SDP-DST [Lee et al., 2021] In SDP-DST, schema information is used as a prompt to query a sequence-to-sequence language model (e.g., T5). It achieves SOTA on MultiWOZ 2.2. Similar to TRADE, the value for each domain and slot pair is predicted in a separate pass.

TransferQA [Lin et al., 2021a] TransferQA reformulated DST as QA problem. It is the state-of-the-art model for zero-shot DST. The model is pretrained with a large amount of QA data. At inference time, the model predicts slot values by taking synthesized extractive questions as input.

DS2 [Shin et al., 2022] In DS2, DST is reformulated as a dialogue summarization problem. Sequence-to-sequence language models are trained with synthetic summary templates. The dialogue states can be recovered by reversing the template generation rules. This is by far the strongest few-shot model in the literature, outperforming recent few-shot models like T5-DST [Lin et al., 2021b]. However, different from IC-DST, this model still requires finetuning on DST labels.

4.3.3 Experimental Details

Language models **GPT-3** [Brown et al., 2020] is a language model with 175B parameters pretrained on a large web corpus. It demonstrates strong zero-shot results on language modeling benchmarks. Its successor, **Codex** [Chen et al., 2021b], is pretrained using open-source code from Github.[3] This enables interesting applications such as code completion. In our initial studies, as in [Shin and Van Durme, 2022], Codex substantially outperforms GPT-3; therefore, we use Codex for the following experiments. In this paper, we use **Codex-Davinci**.[4] In addition, we report results using **GPT-Neo** (2.7B) [Black et al., 2021] and **CodeGen** (2.7B) [Nijkamp et al., 2022]. They are both pretrained on Pile [Gao et al., 2020] and open-source code. For more model details, please refer to Appendix A.1.2.

Few-shot setting We follow the multi-domain scenario from Wu et al. [2020b], where 1%, 5%, and 10% of training data are sampled as the selection pool. The retriever is finetuned on the selection pool and does not see any other DST data.

[3] `https://openai.com/blog/openai-codex/`

[4] In particular, we use code-davinci-002 engine. Some papers mention that the model size of Codex-Davinci is 175B, but OpenAI does not officially confirm that.

Evaluation The standard dialogue state tracking joint goal accuracy (DST JGA) is used as the evaluation metric. To be consistent with prior work [Wu et al., 2019], we report all-domain DST JGA on few-shot settings. We also report the F_1 on slot-value pairs for analysis.

4.4 Results

Few-shot DST on MultiWOZ Table 4.1 shows the result on few-shot settings and full-shot settings of our IC-DST compared with several previous baselines on MultiWOZ 2.1 and 2.4. As discussed in Section 4.3.1, MultiWOZ 2.4 is a clean version of MultiWOZ 2.1 and therefore the performance is better. Our system achieves state-of-the-art performance for 1%, 5%, and 10% few-shot learning settings using Codex, outperforming previous works that require model finetuning. When given more data as retrieval candidates, our systems improve. GPT-Neo and Codegen have a similar trend but are generally worse than Codex and other baselines. This suggests that the size of language model matters when deploying ICL.

The prior ICL DST systems did not report on the standard few-shot configurations, so were not included as baselines. However, our system represents a significant advance over these systems as well, substantially outperforming both UnifiedSKG [Xie et al., 2022] (43.1% vs. 23.5% when using about 80 labeled dialogues) and Few-Shot Bot [Madotto et al., 2021] (50.6% vs. 13.9%) on MultiWOZ 2.1.

4.5 Analysis

To better understand the effectiveness of our proposed methods, we provide detailed analysis in this section. All ablation experiments are conducted on 100 random MultiWOZ 2.4 development set dialogues in the 5% few-shot setting.

Dialogue context representation	Example label (retrieval objective)	MW 2.4 DST JGA
full context	dialogue state	45.0
prev. state + single turn	dialogue state	47.9
full context	state changes	52.0
single turn	state changes	55.8
prev. state + single turn	state changes	58.7

Table 4.2: Comparison of dialogue context representation in prompt examples and different retrieval objectives, with 5% of the training data and Codex. The retrieval objectives correspond to F_1 over all slot-value pairs in the dialogue state vs. just state changes.

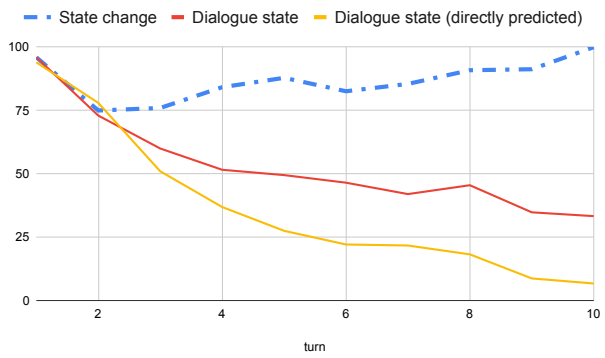

Figure 4.5: DST JGA of each turn. The blue line is the JGA of state changes (TLB JGA) predicted by the system on Table 4.2 row 5 (our IC-DST setting). The red line is the JGA of dialogue states (DST JGA) produced by accumulating predicted state changes. The yellow line is the JGA of dialogue states (DST JGA) predicted by the system on Table 4.2 row 2.

Representation of dialogue context Table 4.2 compares approaches to representing dialogue context in the examples in the prompt, with different retriever fine-tuning objectives. For each setting, we train a retriever with the given dialogue context representation and retrieval objective for a fair comparison. We experiment with representing the dialogue history by: **(1)** concatenating the whole dialogue history, **(2)** only the latest turn (one system utterance and one user utterance), and **(3)** the previous dialogue states and the latest turn. For both of the retrieval objectives, representing the dialogue context by the previous dialogue state and the current turn gives the best performance.

Dialogue states vs. state changes We also explore the benefit of using state change vs. full dialogue state labels (c_t vs. y_t). Note that example labels also act as the basis for the similarity objective in training the retriever. Table 4.2 shows that using state changes gives substantial improvement.

We further compare these two kinds of example labels by investigating the JGA on each turn, illustrated in Figure 4.5. Because states have increasing numbers of slots with more turns, JGA (red and yellow lines) of the full state decreases for later turns. However, all turns have a relatively small number of state changes, so the state change JGA (blue line) remains high throughout the dialogue. As a result, the JGA of the full state benefits from using predicted state change updates (red line, Table 4.2 row 5) as compared to predicting the full state (yellow line, Table 4.2 row 2).

	Traditional	Text-to-SQL
GPT-Neo 2.7B	10.7	33.2
CodeGen 2.7B	17.6	30.6
Codex-Davinci	49.1	58.7

Table 4.3: Comparison of DST generation target formulation using 5% training data and Codex. The traditional format corresponds to the representation in SimpleTOD [Hosseini-Asl et al., 2020].

Effect of DST as Text-to-SQL Table 4.3 shows the performance of our ICL framework given different input-output formats. We follow SimpleTOD [Hosseini-Asl et al., 2020] as the traditional format to formulate the input-output. More specifically, the exemplar labels and the generation target are slot-value pairs. Also, we rewrite the dialogue schema in the same format in the prompt to replace the SQL tables. The prompt in the setting is shown in Appendix A.1.1. By reformulating DST as a text-to-SQL task, and leveraging large language models pretrained with code, ICL can make better use of the structured knowledge associated with the task schema. As shown in Table 4.3, GPT-Neo, CodeGEN, and Codex all perform better with text-to-SQL format. Note that the performance of GPT-Neo with the traditional format is much worse than with the text-to-SQL format, possibly due to its much smaller model size compared to Codex. It is easier for GPT-Neo to work with SQL, rather than learning a slot value pair format.

Error analysis In examining a subset of IC-DST errors, we identified three common types, as shown in Table 4.4. The first type of error is caused by a *noisy training example*, such as a missing slot. ICL is sensitive to noisy training data because the inference LM only sees a few examples in the prompt during prediction. In this example, there is a missing slot 'attraction-type' in the example label. The second type of error is *retrieval limitations*. In this case, the retrieved samples are not good exemplars, because they lack some slots that should be predicted in the test instance. This could be due to sparse annotated data (which impacts all few-shot learning methods) or a retriever error. In this example, the inference LM is mimicing the example, only predicting slot 'attraction-name' and missing the slot 'hotel-pricerange'. The third type of error is *failure to generalize from examples*, which happens when the model fails to learn from examples. In this example, the in-context example contains the notion of slot value 'don't care'. However, the LM fails to generalize this notion and misses the slot value pair 'hotel-area: don't care'.

Error type I	**Noisy training example**
Example	[user] I need some information about churchill college.
Example label	*attraction-name: churchill college*
Test turn	[user] I am hearing some good things about queens college, can you give me some basic info on them?
Prediction	*attraction-name: queens college*
Gold	*attraction-name: queens college, attraction-type: college*
Error type II	**Retrieval limitations**
Example	[system] I found the Scudamores Punting co. and the Cambridge Punter. Which would you prefer? [user] I like the Cambridge Punter better. Can you give the phone number and postcode for them?
Example label	*attraction-name: Cambridge Punter*
Test turn	[system] There are 2 in the centre of town. Scudamores Punting co., and the Cambridge Punter. Would either of those interest you? [user] could you give me the address for the Cambridge Punter, please? I also need a place to stay , preferably somewhere cheap.
Prediction	*attraction-name: Cambridge Punter*
Gold	*attraction-name: Cambridge Punter, hotel-pricerange: cheap*
Error type III	**Failure to generalize from examples**
Example	[user] I would like some information on places to stay in Cambridge. I prefer a guesthouse that includes free WiFi, parking does not matter.
Example label	*hotel-internet: yes, hotel-type: guest house, hotel-parking: don't care*
Test turn	[system] What area of town would you like to be in? [user] It doesn't matter. I just want it to be a cheap guesthouse with WiFi included.
Prediction	*hotel-internet: yes, hotel-type: guest house*
Gold	*hotel-internet: yes, hotel-type: guest house, hotel-pricerange: cheap, hotel-area: don't care*

Table 4.4: The most common error types of IC-DST Codex and their corresponding most similar examples. The missed slots in gold state changes are marked in red.

4.6 Summary

We successfully apply in-context learning for dialogue state tracking by introducing a new approach to representing dialogue context, a novel objective for retriever training, and by reformulating DST into a text-to-SQL task. On MultiWOZ, our system achieves a new state-of-the-art in a few-shot setting. In particular, the text-to-SQL formulation gives 18% relative improvements compared to the conventional format. Our analyses show that each innovation benefits performance. We also study in detail the contribution of each design decision. Future work may apply this in-context learning framework to a wider range of dialogue tasks.

Chapter 5

ORCHESTRALLM: Efficient Orchestration of Language Models for Dialogue State Tracking

5.1 Introduction

In this chapter, we propose a new routing framework that simultaneously leverages an SLM and an LLM to achieve effective and efficient DST. LLMs have become versatile tools capable of tackling a wide range of tasks with only a few training examples. However, their expanding sizes have brought escalating computational demands. In contrast, more efficient SLMs often require a substantial amount of fine-tuning data to become truly effective. This work addresses scenarios where only limited task-specific data is available, making fine-tuned SLMs less dependable. Our objective is to develop a routing framework that orchestrates SLMs and LLMs, enhancing task performance while reducing computational costs.

Fine-tuned SLMs have been used in DST for a few years, including both autoregressive LMs [Ham et al., 2020; Hosseini-Asl et al., 2020; Peng et al., 2021] and sequence-to-sequence LMs [Lee et al., 2021; Su et al., 2022; Bang et al., 2023; Imrattanatrai and Fukuda, 2023; Wang et al., 2023]. LLMs have been used for few-shot in-context learning in DST [Xie et al., 2022; Hudeček and Dušek, 2023; Hu et al., 2022; King and Flanigan, 2023] where LLMs are prompted with human-authored task descriptions or in-context

exemplars. In our work, we seek to take advantage of both SLMs and LLMs given a limited amount of training data, assuming that SLMs will still be useful in some contexts.

Strategies that leverage both SLMs and LLMs have been developed to mitigate the computational demands of LLMs. Cascade-based approaches direct a query to an LLM when it cannot be resolved by an SLM [Chen et al., 2023b; Madaan et al., 2023]. These approaches introduce latency and computational redundancy since they consistently query SLMs. Other approaches use binary classifiers to predict the most appropriate LM to utilize [Kag et al., 2022; Šakota et al., 2023]. A limitation of the classifier-based approaches is the necessity for retraining when introducing new models.

In this chapter, we propose a dynamic routing framework, **ORCHESTRALLM** (illustrated in Figure 5.1), that leverages small (fine-tuned) and large LM experts. Hypothesizing that examples with similar semantic embeddings are of the same difficulty level, we select an appropriate expert based on embedding distances between the testing instance and instances in expert pools. The expert pools contain examples representing the types of dialogue contexts where the different LMs provide more reliable answers. After retrieving the top k nearest examples, an expert is selected based on the majority vote. Unlike cascade-based and classifier-based approaches, the proposed framework eliminates the need for router training, though hand-labeled data is needed for creating the expert pools. In addition, the retriever can be fine-tuned with target task labels or expert information to achieve more efficient and accurate routing.

In summary, the key contribution of this chapter is the introduction of a novel switching model designed to reduce the computational costs associated with LLMs while simultaneously enhancing performance. Experimental results on two different multi-domain DST benchmarks (MultiWOZ [Budzianowski et al., 2018; Ye et al., 2022] and SGD [Rastogi et al., 2020b]) demonstrate that ORCHESTRALLM capitalizes on the proficiencies of different experts, outperforming LLM systems while also achieving a substantial reduction of over 50% in computational costs.

This chapter contains material that was originally published in [Lee et al., 2024]. I was the main contributor of this work, and discussed and wrote the paper with the other collaborators throughout the project.

5.2 Problem Definition and Backbone Models

The problem definition is the same as in the previous chapter. Given context $C_t = A_1, U_1, \cdots, A_t, U_t$, the goal is to predict the dialog state:

$$DST_t = y_t = \{(s_i^t, v_i^t); i = 1, \ldots, n_t\}.$$

which is implemented by predicting the turn level belief TLB_t (the set of new or updated slot-value pairs provided in u_t) and then aggregating DST_{t-1} with TLB_t to get DST_t.

In the literature, task-specific SLM-based DST models are typically fine-tuned with full-parameter updates while DST models using LLMs are realized via few-shot in-context learning. We discuss the two different DST models considered below.

LLM DST. We develop IC-DST 2.0 based on [Hu et al., 2022] with the following enhancements: (1) we use GPT-3.5-Turbo as the backbone language model instead of Codex, (2) for SGD [Rastogi et al., 2020b], we incorporate three turn pairs instead of one to handle more complex dialogues, and (3) also for SGD [Rastogi et al., 2020b], to avoid domain confusion, we change the prompt order and move the schema table from the start of the prompt to just before the inference target instance. As in Chapter 4, dialogue states of previous turns are used as a summary of the context history which allows using more exemplars which is crucial for ICL performance. Concretely, given the schema table, K in-context exemplars, dialogue states of the previous turn, and the input instance (most recent agent-user utterance pairs), the LLM outputs (for MultiWOZ)

$$TLB_t = \text{LLM}(T, E_{1:K}, DST_{t-1}, A_t, U_t)^1 \tag{5.1}$$

where T is the schema table for all domains in MultiWOZ, and each exemplar $E_i = (DST_{\tau-1}, A_\tau, U_\tau, TLB_\tau)$ consists of previous dialogue states as context, one pair of turn change, and associated outputs.

For the SGD dataset, the schema table T is tailored to include only the domains relevant to the target instance, and the order of elements is slightly adjusted. The LLM outputs are given by:

[1]The elements appear in the same sequential order as they are presented in the prompt.

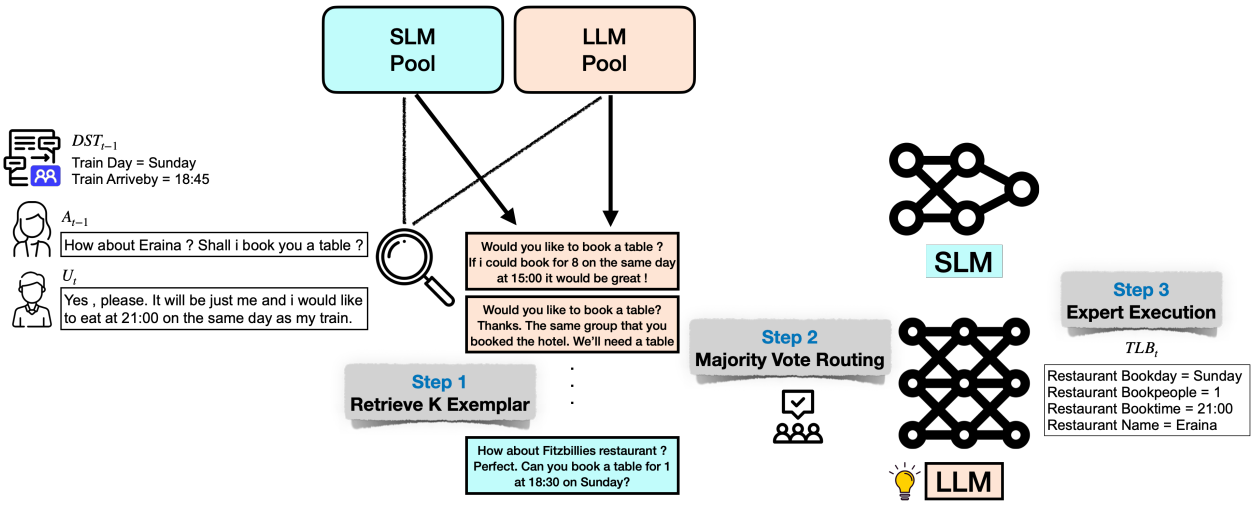

Figure 5.1: Illustration of ORCHESTRALLM. LMs are orchestrated by a retrieval-based dynamic router. During inference, the testing instance queries the expert pools to retrieve top k similar examples. Subsequently, a LM expert is selected based on the majority vote.

$$TLB_t = \text{LLM}(E_{1:K}, T, DST_{t-1}, A_{t-2}, U_{t-2}, A_{t-1}, U_{t-1}, A_t, U_t) \tag{5.2}$$

SLM DST. Here, we develop a prompt-based DST model (denoted as *Prompt-DST*) with SLM (T5 [Raffel et al., 2020]). The input of Prompt-DST is similar to IC-DST, except that the in-context exemplars are excluded. Specifically, given the schema prompt-augmented input, the model outputs

$$TLB_t = \text{SLM}(T, DST_{t-1}, A_t, U_t). \tag{5.3}$$

Here, the model is trained using the learning objective by maximizing the log-likelihood of slot values $v_t(d_m, s_n)$ for the current TLB, i.e.

$$\max \log P(TLB_t | T, DST_{t-1}, A_t, U_t). \tag{5.4}$$

During inference, a greedy decoding procedure is directly applied, i.e.only the most likely token in the given model vocabulary is predicted at each decoding step.

5.3 Routing Approach

Here, we present our approach for routing with ORCHESTRALLM applied to the DST task. The overall framework is illustrated in Figure 5.1. We denote different DST models as **experts**. Given a new input instance (the triplet (DST_{t-1}, A_t, U_t)), ORCHESTRALLM first computes its semantic embedding, compares it with exemplar embeddings of triplets from each **expert pool** using a cosine distance, and retrieves the top-K exemplars. The router assigns the input to an expert based on majority vote. While our approach draws inspiration from the work of Jang et al. [2023], it is important to note that their approach primarily focuses on optimizing task performance in zero-shot task transfer scenarios, whereas our emphasis lies in improving computational efficiency within the few-shot learning setting.

5.3.1 Expert Pool Construction

For each dialogue in a small held-out set, the SLM and LLM experts are used to predict the TLB at each user turn (TLB_t) individually. If both experts correctly predict the TLB, the instance triplet is included in the SLM pool. When only one expert correctly predicts the TLB, the instance is assigned to that expert's pool. Instances that are not correctly predicted are not used in either pool.

5.3.2 Triplet Representation Learning

Similar to recent work on dense retrieval [Karpukhin et al., 2020], the retriever uses a bi-encoder architecture, which encodes dialogues with labels and predictions into embedding space. Throughout the work, SenBERT [Reimers and Gurevych, 2019] is used as the backbone embedding model. The bi-encoder is fine-tuned using a small set of dialogues, the same as that used to construct the expert pools. We use a contrastive loss such that the similarity between a positive example pair is high and the similarity between a negative example pair is low. Three different methods for constructing positive and negative pairs are explored: task-aware, expert-aware, and their combination.

Task-Aware Supervision identifies positive and negative instance pairs for training by first computing pairwise similarity for each sample in the hold-out set. Then, the l highest and lowest scoring pairs are used as positive and negative examples, respectively. The similarity function leverages the gold annotations of the hold-out set dialogues. Given two instances, a and b, the similarity is a weighted combination of the

slot-value similarity of the previous state (DST) and the current TLB:

$$Sim_{TLB} + \frac{1}{2} Sim_{DST}.$$

Let $TLB_x = \{(s_1^x, v_1^x), \cdots, (s_m^x, v_m^x)\}$ be the TLB of instance x. Following **?**, the slot-value pair similarity is

$$F_{\textit{slot-value}} = F(\{(s_1^a, v_1^a), \cdots, (s_m^a, v_m^a)\}, \{(s_1^b, v_1^b), \cdots, (s_n^b, v_n^b)\}).$$

and the slot similarity is

$$F_{slot} = F(\{s_1^a, \cdots, s_m^a\}, \{s_1^b, \cdots, s_n^b\}).$$

where F is the standard definition of F1 score i.e.$F = \frac{2PR}{P+R}$, in which P is precision, and R is recall. The similarity score between TLB_a and TLB_b is

$$Sim(TLB_a, TLB_b) = F_{\textit{slot-value}} + F_{slot} - 1$$

The context history similarity Sim_{DST} is defined in the same way.

Expert-Aware Supervision first groups instances in the hold-out set according to which expert gave the most accurate prediction. (For ties, the SLM is chosen.) We then compute pairwise triplet similarities using an off-the-shelf embedder (e.g.SenBERT). The l highest scoring pairs with the same expert label are positive examples, and the l lowest scoring pairs with different expert label are negative examples.

Task+Expert-Aware Supervision simply pools both sets of positive and negative pairs.

Note that task-aware supervision is agnostic to what experts are used in routing, so the embedding model need not be retrained as experts are added or updated. Expert-aware supervision will require updating the embedding model if the experts change. In all cases, the expert pools will need to be updated with changes to the experts.

Dataset	MultiWOZ	SGD
# Domains	8	41
# Dialogues	8438	16142
# Total Turns	113556	329964
Avg. Turns per Dial.	13.46	20.44
Avg. Toks per Turn	13.13	9.75
# Slots	24	214
# Slot Values	4510	14139

Table 5.1: Experiment data summary. The numbers are computed on training splits of the datasets.

5.4 Experiments

5.4.1 Datasets

We use two datasets detailed below for experiments. A summary of DST datasets is reported in Table 5.1.

MultiWOZ [Budzianowski et al., 2018] is a multi-domain task-oriented dialogue dataset that contains over 10K human-human written dialogues across 8 domains and has been one of the most popular benchmarks in the DST literature. After the publication of Budzianowski et al. [2018], many works improve the label qualities, e.g. MultiWOZ 2.1 [Eric et al., 2020] and MultiWOZ 2.4 [Ye et al., 2022]. We experiment using the most recent version, MultiWOZ 2.4.

SGD [Rastogi et al., 2020b] is a task-oriented dialogue dataset that contains over 16k multi-domain conversations spanning 41 domains, featuring out-of-domain evaluation. 15 out of 21 domains in the test set are not present in the training set and 77% of the dialogue turns in the test set contain at least one domain not present in the training set.

5.4.2 Experimental Setting

In this work, we consider a **low-resource** set up for DST. Following the multi-domain experiment setting from TRADE [Wu et al., 2019], we randomly sample 5% of training data from MultiWOZ and SGD respectively for training the expert models.

Model and Hyperparameter Setting. For PROMPT-DST, we use T5-base and T5-large as the backbone model for MWOZ and SGD respectively, as the latter is more complex in terms of schema and more dialogue

turns. For IC-DST 2.0, we use ChatGPT as the backbone model[2] with 10 in-context exemplars. We initialize the routing retriever from SenBERT (all-mpnet-base-v2). We run inference on 100 dialogues randomly sampled from validation sets of MWOZ and SGD as the held-out sets. The same 100 dialogues are used to train the retriever. For all experiments, $l = 25$ is used for the positive and negative examples for contrastive learning. During inference, we randomly sample 100 turns from the held-out sets to serve as SLM pool and LLM pool respectively for MWOZ experiments and 300 turns for SGD experiments.[3] We leave it for future work to create novel strategies for expert pool instance selection. We use $k = 10$ for the majority vote and break the tie by favoring SLM.

5.4.3 Evaluation

Accuracy

Conventionally, DST systems are evaluated by joint goal accuracy (JGA) on accumulated dialogue states [Henderson et al., 2014]. This metric assesses the correctness of the dialogue state at each turn and deems it accurate only if all slot values within every domain precisely match the ground-truth values. It is difficult to accurately assess how well a system performs on single turns with DST JGA. Therefore we also report turn-level belief (TLB JGA) [Dey et al., 2022].

Efficiency

Floating-point operations per Second (FLOPs) represent the number of floating-point arithmetic operations (additions and multiplications) a model performs in one pass. FLOPs are often used to estimate the computational cost or workload required for training or inference. Training a large model requires a significant number of backward passes, which are more expensive than forward passes, yet inference is a continuous process that happens whenever the model is in use, thus accruing more cost over time. NVIDIA [Leopold, 2019] and Amazon [Barr, 2019] report around 90% of the ML workload is inference processing in their cloud services. Therefore, we choose to report FLOPs for inference time usage.

We estimate the aggregate computational cost, measured in TeraFLOPs, required for performing inference across the entire testing dataset. It is important to note that IC-DST 2.0 relies on ChatGPT, a model

[2]Accessed: August–October 2023, Version: gpt-3.5-turbo-0301.

[3]We also experimented with 50 turns for MWOZ and observed less than 1% accuracy degradation.

that is not publicly accessible, thus precluding a direct evaluation of its computational efficiency. Based on prevailing conjecture within the public domain, ChatGPT is presumed to be a fine-tuned iteration of the GPT-3 model with a substantial parameter count of 175 billion [Brown et al., 2020]. To estimate the computational requirements, we conduct FLOPs measurements on the GPT-2 [Radford et al., 2019] model and subsequently scale these measurements in accordance with the parameter size differential between GPT-2 and ChatGPT. The computational cost of the retriever, measured in FLOPs, for each turn instance, is approximately 0.02 TeraFLOPs. This computational load becomes negligible when considered in conjunction with ChatGPT in the ORCHESTRALLM. ChatGPT requires approximately 3000 TeraFLOPs for each turn instance.

5.4.4 Routing Baselines

Classification-Based Routing

We compare our routing framework with existing classification-based approaches to model switching, such as those proposed by Šakota et al. [2023] and Kag et al. [2022]. These existing approaches typically train a binary classifier to serve as the router. We train BERT [Devlin et al., 2019] (Bert-base-cased) with the expert labels in the hold-out set of dialogues with a binary classification objective to do routing as a baseline.

Cascade-Based Routing

Cascade-based approaches Chen et al. [2023b]; Madaan et al. [2023] typically query a SLM and redirect the instance to a LLM if the smaller language model is not confident enough. We choose to utilize the normalized sequence level probability of SLM output as the confidence measure. We tune the probability threshold on the hold-out-set and use the threshold to determine whether to redirect the instance to LLM during inference.

5.4.5 Results

MultiWOZ

We demonstrate the MultiWOZ experiments in a low-resource setting in Table 5.2. PROMPT-DST and IC-DST 2.0 perform inference on another 100 dialogues from the validation set, documenting the turns each expert specializes in. We randomly select 100 turns from these dialogues for each expert to serve as expert

Models	Router	Assignment Ratio	TeraFLOPs	TLB JGA	DST JGA
DST Baselines					
PROMPT-DST	N/A	N/A	272	73.43	46.06
IC-DST 2.0	N/A	N/A	22 M	78.21	49.68
DS2 - T5 [Shin et al., 2022]∗	N/A	N/A	N/A	N/A	49.89
RefPyDST [King and Flanigan, 2023]	N/A	N/A	N/A	N/A	62.30
Routing Baselines					
Prompt-DST & IC-DST 2.0	Oracle	73% Prompt-DST	5.94 M	88.07	65.39
Prompt-DST & IC-DST 2.0	Classification-Based	91% Prompt-DST	1.98 M	77.60	47.58
Prompt-DST & IC-DST 2.0	Cascade-Based	13 % Prompt-DST	19.14 M	80.40	51.46
Our Retrieval-Based Routing DST					
ORCHESTRALLM	SenBERT	60% Prompt-DST	8.8 M	80.74	50.19
ORCHESTRALLM	Task-Aware	55% Prompt-DST	9.9 M	82.43	52.53
ORCHESTRALLM	Expert-Aware	78% Prompt-DST	4.8 M	81.02	50.65
ORCHESTRALLM	Task+Expert-Aware	62% Prompt-DST	8.3 M	**82.46**	**52.68**

Table 5.2: Results on MultiWOZ 2.4 using 5% training data. The TeraFLOPs are computed on inference passes on the entire testing set. We report the percentage of turns routed to PROMPT-DST in the assignment ratio column. ∗ marks numbers reported in Hu et al. [2022].

pools for dynamic routing.

As expected, IC-DST 2.0 outperforms PROMPT-DST in the few-shot setting, indicating that the LLM is more generalizable than the fine-tuned SLM. The BERT-based classification router struggles to effectively harness the capabilities of both models. To establish an upper performance bound for the learned router, we introduce the oracle router, which aggregates predictions from both LLM and SLM when either model is correct, with a preference for SLM whenever available. Even with a vanilla SenBERT as a retriever, ORCHESTRALLM outperforms IC-DST 2.0 while saving 60% calls to LLM, demonstrating the effectiveness of our proposed framework. Further finetuning the retriever with the proposed task-aware contrastive examples routes examples more effectively and improves DST JGA around 3% compared to IC-DST 2.0. With additional expert-aware training of the retriever, we can further save around 7% traffic to LLM with superior performance compared with IC-DST 2.0. In spite of its compact size, PROMPT-DST is finetuned to align with specific in-domain knowledge and task-specific artifacts (e.g.schema constraints and customized labeling strategies). Conversely, IC-DST 2.0 is enriched with an extensive repertoire of knowledge acquired during the pretraining phase of LLM, endowing it with contextual reasoning capabilities and an enhanced grasp of common-sense knowledge (Section 5.5.2). Since these two models are complementary, an effective integration can surpass the performance of the IC-DST 2.0 model.

Models	Router	Assignment Ratio	TeraFLOPs	TLB JGA	DST JGA
DST Baselines					
PROMPT-DST	N/A	N/A	8882	62.21	28.38
IC-DST 2.0	N/A	N/A	121 M	63.86	33.15
Routing Baselines					
Prompt-DST & IC-DST 2.0	Oracle	62% Prompt-DST	45.98 M	77.48	47.50
Prompt-DST & IC-DST 2.0	Classification-Based	38% Prompt-DST	75.02 M	66.94	31.86
Prompt-DST & IC-DST 2.0	Cascade-Based	7.9% Prompt-DST	111.34 M	64.17	32.75
Our Retrieval-Based Routing DST					
ORCHESTRALLM	SenBERT	50% Prompt-DST	60.50 M	65.97	32.75
ORCHESTRALLM	Task-Aware	55% Prompt-DST	54.45 M	67.25	32.78
ORCHESTRALLM	Expert-Aware	54% Prompt-DST	55.66 M	67.34	32.95
ORCHESTRALLM	Task+Expert-Aware	57% Prompt-DST	52.03 M	**68.09**	**33.07**

Table 5.3: Results on SGD using 5% training data. The TeraFLOPs are computed on inference passes on the entire testing set. We report the percentage of turns routed to Prompt-DST in the assignment ratio column.

The state-of-the-art few-shot system for MultiWOZ is RefPyDST [King and Flanigan, 2023]. Similar to IC-DST, RefPyDST utilizes in-context learning with Codex, which is no longer accessible to the general public. Moreover, RefPyDST generates a candidate set by performing five inference passes with Codex and selecting the best one using the scoring strategy from [Holtzman et al., 2021]. This method significantly increases computational costs. The fact that the oracle router gives a system that outperforms RefPyDST suggests that high performance can be achieved with lower cost with improvements to routing.

SGD

To evaluate our system under out-of-domain scenarios, we show experimental results in a low-resource setting on SGD in Table 5.3. We use 5% of dialogues in the training set for finetuning PROMPT-DST and the retriever of IC-DST 2.0. PROMPT-DST and IC-DST 2.0 performed inference on another 100 dialogues in the validation set to serve as expert pool. We randomly select 300 turns from each expert to serve as expert pools for dynamic routing.

As we observe in MultiWOZ, incorporating an off-the-shelf SenBERT as the router improves the TLB score and also saves around 50% of computes. Finetuning SenBERT with the task-aware objective improves efficiency by 5% and increases both the TLB and DST scores. With the additional expert-aware supervision, more turns are routed to SLM and improves TLB score. This setting outperforms IC-DST 2.0 by over 4% TLB JGA and saves 57% FLOPs, demonstrating that our router is universal enough to support cross-domain

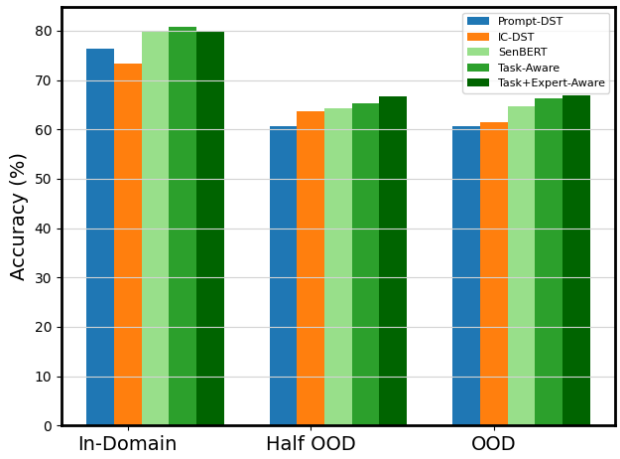

Figure 5.2: Cross-domain generalization results on SGD. We denote *In-Domain* when all of the testing domains are in the training set and denote *OOD* when all of the testing domains are not in the training set. For all other dialogues, we categorize them as *Half OOD*. We report TLB JGA for all settings. Green bars indicate ORCHESTRALLM with different retrievers.

assignment and successfully improves system accuracy.

5.5 Analysis

5.5.1 Cross-Domain Generalization

Out-of-Domain (OOD) in SGD To assess the effectiveness of ORCHESTRALLM in generalizing to unseen domains, we present breakdown results on SGD in Figure 5.2. First, we observe that PROMPT-DST performs better than IC-DST 2.0 on in-domain dialogues but lags behind IC-DST 2.0 on all other types of dialogues. This suggests that the generalization ability of Large Language Models (LLMs) is superior to Smaller Language Models (SLMs). All variants of ORCHESTRALLM outperform IC-DST 2.0 in OOD scenarios, demonstrating the router's capability to effectively dispatch instances even when they are out of the domain.

5.5.2 Specialty of SLM and LLM

To better understand the complementary nature of the LMs, we inspected MultiWOZ examples to identify their specialties. We provide representative examples from the expert pools in Table 5.4. One common mistake made by LLM is failing to adhere to the schema. In this example, LLM simply copies the text

Example from SLM pool	
DST of Previous Turn	*restaurant-area: centre*
Test turn	[system] Do you have a cuisine or price range in mind? [user] Yes, something in the affordable price range. Also, do any of them serve Singaporean food ?
SLM Prediction	*restaurant-food: Singaporean, restaurant-pricerange: cheap*
LLM Prediction	*restaurant-food: Singaporean, restaurant-pricerange: affordable*
Example from LLM Pool	
DST of Previous Turn	*hotel-name=Alpha Milton guest house*
Test turn	[system] Would you like to book a room? [user] That would be a massive help if you can do that for me! It's me and my mum and we'll be there for 2 nights.
SLM Prediction	*hotel-bookstay: 2, hotel-bookpeople: 1*
LLM Prediction	*hotel-bookstay: 2, hotel-bookpeople: 2*
DST of Previous Turn	*restaurant-name=Cocum, restaurant-area: west*
Test turn	[system] Can I be of any further assistance today? [user] Yes, I am also looking for a 3-star hotel located in the same area as the restaurant.
SLM Prediction	*hotel-stars: 3, hotel-area: centre*
LLM Prediction	*hotel-stars: 3, hotel-area: west*

Table 5.4: Representative MultiWOZ examples from SLM and LLM pool. Red color text indicates the errors made by LMs.

("affordable") from the turn as a DST prediction, while SLM is capable of grounding the value in the schema-specific format ("cheap"). However, we identify two strengths that LLM possesses over SLM. Firstly, it excels in handling common-sense knowledge, for example, it can infer the correct number of guests staying at the guest house from the context ("me and my mum"). Secondly, it demonstrates proficiency in long-context reasoning. When there is a reference to previous context across domains, LLM consistently makes the correct inference, while SLM often overlooks the context and produces random values.

Additional insights can be gained by looking at the slot accuracy for categorical vs. non-categorical slots that are in-domain vs. OOD. Table 5.5 shows examples for 2 randomly selected domains of each type. The OOD categorical slots pose the most significant challenge for the SLM system. The SLM system struggles to comprehend two new concepts, "Buses_3-additional_language" and "Buses_3-category", but it demonstrates high accuracy in predicting "number_passengers," a concept it has already been trained on. The SLM is also good at dates, a concept that is represented in many domains, and ORCHESTRALLM uses the SLM more often for dates. Both LMs struggle with the alarm domain, which is unlike the booking domains that dominate training.

It is noteworthy that ORCHESTRALLM demonstrates proficiency in handling location concepts and can surpass both standalone LLM and SLM systems in this regard (e.g., "Buses_3-to_city," "Hotels_2-where_to," "Travel_1-location"). Additionally, the router appears capable of making accurate routing deci-

Slot	CAT	OOD	Dual	SLM	LLM	SLM% in Dual
Alarm_1-new_alarm_time	N	Y	35.1%	25.5%	43.5%	23.4%
Alarm_1-new_alarm_name	N	Y	34.6%	32.1%	47.4%	38.5%
Buses_3-from_city	N	Y	86.0%	91.5%	76.7%	47.0%
Buses_3-to_city	N	Y	78.2%	67.9%	72.2%	40.1%
Buses_3-departure_date	N	Y	83.8%	96.9%	75.3%	46.2%
Buses_3-departure_time	N	Y	59.7%	76.4%	60.9%	30.1%
Buses_3-additional_luggage	Y	Y	31.8%	4.1%	34.9%	15.0%
Buses_3-num_passengers	Y	Y	88.3%	87.9%	87.4%	38.6%
Buses_3-category	Y	Y	39.9%	1.6%	70.2%	3.4%
Hotels_2-where_to	N	N	81.9%	47.8%	74.4%	46.4%
Hotels_2-number_of_adults	Y	N	96.9%	96.6%	96.9%	34.5%
Hotels_2-check_in_date	N	N	74.5%	90.0%	62.9%	45.6%
Hotels_2-check_out_date	N	N	70.8%	87.9%	67.1%	44.7%
Hotels_2-rating	N	N	98.6%	96.3%	99.1%	23.8%
Hotels_2-has_laundry_service	Y	N	94.4%	73.5%	95.1%	32.3%
Travel_1-location	N	N	80.0%	56.7%	76.5%	32.0%
Travel_1-category	Y	N	72.8%	60.9%	71.0%	28.3%
Travel_1-free_entry	Y	N	94.4%	90.1%	96.5%	18.6%
Travel_1-good_for_kids	Y	N	95.6%	90.6%	96.1%	22.3%

Table 5.5: System performance (TLB accuracy) and assignment ratio (SLM%) on SGD test set by slots. Dual corresponds to ORCHESTRALLM. We denote CAT as categorical and OOD as out-of-domain. We also report the assignment ratio of the ORCHESTRALLM in "SLM% in Dual".

Models	***Assignment***	**TLB JGA**
T5-base	N/A	73.43
T5-3B	N/A	78.77
ORCHESTRALLM (T5-base+T5-3B)	61%	81.09

Table 5.6: MultiWOZ routing results between T5-base and T5-3B (a new LM) using an off-the-shelf SenBERT. We denote the % of testing turns routed to T5-base as *Assignment*.

sions even when dealing with OOD slots featuring new concepts (e.g., "Buses_3-additional_luggage" and "Buses_3-category").

5.5.3 Routing a New LM

To demonstrate the flexibility of ORCHESTRALLM, we provide routing results of when integrating a new LM, specifically T5-3b finetuned with Prompt-DST method on MultiWOZ 5% training data. We apply ORCHESTRALLM with an off-the-shelf SenBERT to route between T5-base and T5-3b. The results displayed in Table 5.6 underscore the adaptability of ORCHESTRALLM in effectively routing examples with a newly introduced LM, all without requiring any additional training of the retriever.

5.6 Summary

We introduce ORCHESTRALLM, a routing framework that seamlessly integrates a SLM and a LLM, orchestrated by a retrieval-based router. During inference, a dynamic router guides instances to either LM based on their semantic embedding distances with the retrieved LM exemplars, leveraging the expertise of both SLM and LLM. Our evaluation on DST demonstrates that ORCHESTRALLM outperforms LLM-based systems while also achieving computational cost savings of over 50%. This research represents a significant step towards efficient collaboration of language models, particularly in a multi-turn human-computer interaction system such as task-oriented dialogue.

Chapter 6

CORRECTIONLM: Self-Corrections with SLM for Dialogue State Tracking

6.1 Introduction

As noted in Chapter 2, what constitutes an SLM has been increasing in size due to technology improvements, with increasing ability to do in-context learning. In this chapter, we take advantage of this with Llama3 [Dubey et al., 2024], which with 8B params can be considered small relative to other frontier models such as GPT-4 and GPT-3.5.

Recently, large language models (LLMs) have demonstrated strong reasoning abilities by providing feedback on their own outputs and subsequently refining them based on that feedback [Shinn et al., 2024; Madaan et al., 2024; Huang et al., 2023; Saunders et al., 2022]. This is especially true for tasks requiring multi-step reasoning (code and math reasoning tasks). However, these capabilities of generating feedback and refinement are less commonly observed in small language models (SLMs) [Saunders et al., 2022; Ye et al., 2023].

To enable the abilities of self-critique and self-refinement of SLMs, previous research has focused on distilling knowledge from LLMs. Typically, this involves fine-tuning SLMs on improvement demonstrations generated by LLMs [Shridhar et al., 2023; Ye et al., 2023] and has proven to improve the self-improvement abilities of SLM.

However, [Yu et al., 2024b] observe that naively training SLMs on LLM improvement demonstrations can hurt task performance, since SLMs may have different error modes, and learning from LLM mistakes may be less beneficial. To address this, they propose generating reasoning trajectories with SLMs and using LLMs to provide feedback and refinement before fine-tuning the SLMs on these edited trajectories. However, this approach still heavily relies on LLM involvement.

In this chapter, we introduce a novel self-improvement framework for DST that finetunes an SLM and makes corrections using in-context exemplars, without involving any LLMs. This approach achieves superior data efficiency and computational efficiency compared to ORCHESTRALLM. As discussed in Chapter 2, the definition of an SLM has evolved due to technological advancements, with even larger SLMs now capable of in-context learning. In this chapter, we leverage this progress by utilizing Llama3, which, with 8 billion parameters, is considered small relative to frontier models such as GPT-4.

In this chapter, we target low-resource settings and use only 5% of the training set for experiments. We first randomly sample a few examples as in-context learning prompts for the SLM to generate dialogue state predictions for the remaining data. This step is intended to capture the errors that SLMs are prone to during in-context learning inference. We then finetune the SLM on the conversations along with self-generated predictions and gold labels, resulting in an SLM capable of making corrections. Unlike prior work, our method does not rely on LLMs to provide feedback or generate predictions. Additionally, our method does not require an external verifier (such as execution environment [Chen et al., 2023a] or heuristic filter [Shinn et al., 2024]), as the model internally decides whether to make changes to the initial predictions.

I was the main contributor to this work, and discussed and wrote the chapter with other collaborators throughout the project.

6.2 Correction Approach with SLM

The problem definition is the same as in the previous chapter. Given context $C_t = A_1, U_1, \cdots, A_t, U_t$, the goal is to predict the dialog state:

$$DST_t = y_t = \{(s_i^t, v_i^t); i = 1, \ldots, n_t\}.$$

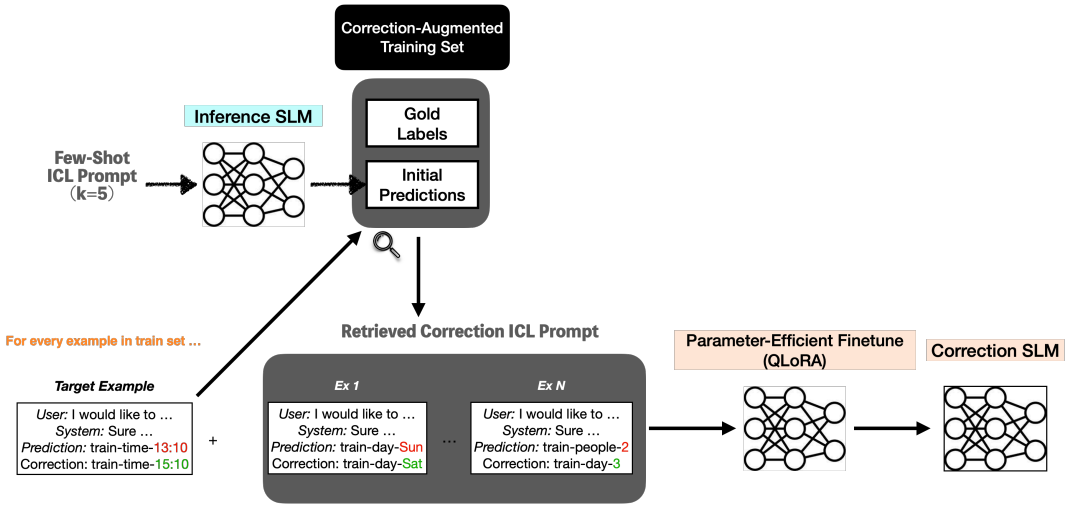

Figure 6.1: Illustration of the training stage of CORRECTIONLM. The first step is to prompt SLM with a few fixed in-context exemplars to produce predictions for each example in the training set. The second step is to finetune SLM to generate gold label given correction-augmented in-context exemplars and the target input and initial target prediction from the first step.

which is implemented by predicting the turn level belief TLB_t (the set of new or updated slot-value pairs provided in u_t) and then aggregating y_{t-1} with TLB_t to get y_t.

Our method focuses on a low-resource setting where only 5% of the training set is used and no external models or tools are available. We choose Llama3 8B as the backbone model and use a in-context learning framework, IC-DST 2.0 developed for DST in the Chapter 5.[1]

There are two stages of the framework: the training stage is illustrated in Figure 6.1, and the inference stage is illustrated in training stage is illustrated in Figure 6.2. We next explain each of the two stages:

6.2.1 Training

Acquire Prediction Demonstration on Train Set The first step of the training is to obtain SLM predictions using ICL in order to provide supervision signals for the correction training. We randomly sample 5 examples as task demonstrations and prompt Llama 3 with IC-DST 2.0 to obtain predictions for every example in the training set. Concretely, for MultiWOZ, given the schema table, 5 in-context exemplars, dialogue states of the previous turn, and the input instance (most recent agent-user utterance pair), the inference_SLM outputs

[1]Our proposed method is model-agnostic and algorithm-agnostic. Other LMs or DST frameworks can be used.

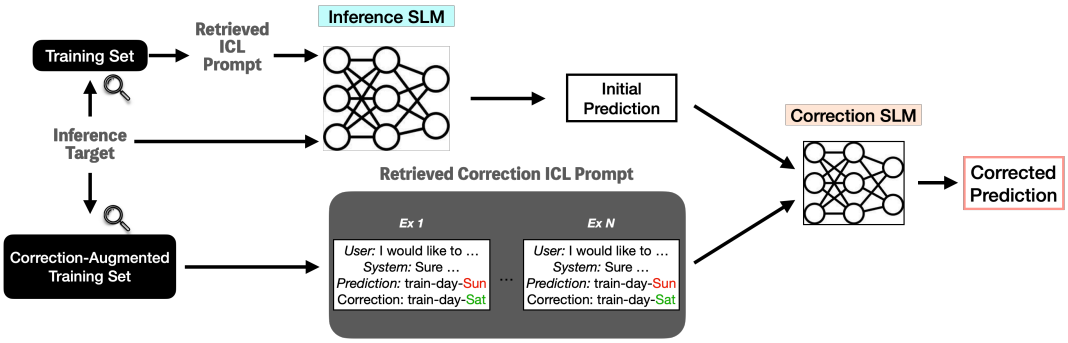

Figure 6.2: Illustration of the inference and correction stage of CORRECTIONLM. The first step is to perform vanilla ICL on the testing set using in-context exemplars from the training set. The second step is to prompt the finetuned SLM with correction-augmented in-context exemplars to make corrections on the initial predictions.

$$TLB_t = \text{inference\_SLM}(T, E_{1:K}, DST_{t-1}, A_t, U_t)^2 \tag{6.1}$$

where T is the schema table for all domains in MultiWOZ and each exemplar $E_i = (DST_{\tau-1}, A_\tau, U_\tau, TLB_\tau)$ consists of previous dialogue states as context, pairs of turn changes, and associated outputs.

For the SGD dataset, which involves more complex dialogues, we use three most recent turn pairs instead of one. The schema table T is tailored to include only the domains relevant to the target instance, and the order of elements is slightly adjusted. The SLM outputs are given by:

$$TLB_t = \text{inference\_SLM}(E_{1:K}, T, DST_{t-1}, A_{t-2}, U_{t-2}, A_{t-1}, U_{t-1}, A_t, U_t) \tag{6.2}$$

Finetune SLM through In-Context Tuning Our initial experiments reveal that models tuned using in-context examples demonstrate significantly stronger correction capabilities compared to those fine-tuned without such examples. This is particularly evident when the correction targets involve domains not present in the training data. To address this, we have enhanced the prompt format in IC-DST 2.0,[3] creating a sequence of examples that includes the previous dialogue states as context, the relevant turn pairs, the initial SLM prediction, and the corresponding gold dialogue state label. For each example in the training set, a fine-tuned dialogue retriever is employed to select few-shot examples from the training data to construct the

[2]The elements appear in the same sequential order as they are presented in the prompt.

[3]For the SGD dataset, we additionally append the schema table to each exemplar to enable the correction model to fully capture the mappings between the schema and the extracted dialogue states.

sequence.

Specifically, for MultiWOZ, given the schema table T for the target domains, a set of in-context examples $G_{1:k}$, along with the dialogue states from the previous turn DST_{t-1} and the most recent agent-user utterance pair, the initial prediction for the turn change H_t is represented as follows:

$$(T, G_{1:K}, DST_{t-1}, A_t, U_t, H_t, TLB_t) \tag{6.3}$$

where T is the schema table for all domains in MultiWOZ, and each exemplar $G_i = (DST_{\tau-1}, A_\tau, U_\tau, H_\tau, TLB_\tau)$ consists of previous dialogue states as context, one pair of turn change, initial prediction, and associated gold outputs.

For the SGD dataset, the sequence is represented as follows:

$$(G_{1:K}, T, DST_{t-1}, A_{t-2}, U_{t-2}, A_{t-1}, U_{t-1}, A_t, U_t, H_t, TLB_t) \tag{6.4}$$

In this case, each exemplar $G_i = (T, DST_{\tau-1}, A_{\tau-2}, U_{\tau-2}, A_{\tau-1}, U_{\tau-1}, A_\tau, U_\tau, H_\tau, TLB_\tau)$ in the SGD dataset includes the schema table T for its respective domains to enable the correction model to fully capture the mappings between the schema and the extracted dialogue states.

This approach ensures that the model effectively leverages in-context examples, improving its ability to generalize and correct across various domains. We train the SLM on the whole sequence with a standard cross-entropy loss. In order to be parameter-efficient, the finetuning process is done with QLoRA [Dettmers et al., 2024], where the parameters in the SLM are quantized before training and only the light-weighted inserted adapters are updated during the training process.

6.2.2 Inference

First Pass To generate the initial predictions for the testing set, we utilize IC-DST 2.0. We use the training set as the retrieval pool to select the 10 examples most similar to the inference instance, which are then used to construct the prompt for the inference SLM. The prompts used in this step are consistent with those in the first step of the training stage, with the primary difference being the inclusion of 10 examples and the retrieval process necessary for selecting the most relevant examples from the pool.

Second Pass For the second pass, we construct a correction prompt by retrieving k examples from the training set, following the IC-DST 2.0 framework. Each retrieved example includes the initial predictions from the first step of the training stage. We then prompt the correction-tuned SLM to refine the initial predictions made in the first pass.

Specifically, for MultiWOZ, given the schema table for the target domains T, the in-context examples, dialogue states from the previous turn, and the input instance (the most recent j agent-user utterance pairs), the initial prediction for the turn change H_t, the correction_SLM outputs

$$TLB_t = \text{correction\_SLM} = (T, G_{1:K}, DST_{t-1}, A_t, U_t, H_t) \tag{6.5}$$

where T is the schema table for all domains in MultiWOZ, and each exemplar $G_i = (DST_{\tau-1}, A_\tau, U_\tau, H_\tau, TLB_\tau)$.

For the SGD dataset, given the schema table for the target domains T, the in-context examples, dialogue states from the previous turn, and the input instance (the most recent j agent-user utterance pairs), the initial prediction for the turn change H_t, the correction_SLM outputs

$$TLB_t = \text{correction\_SLM} = (G_{1:K}, T, DST_{t-1}, A_t, U_t, H_t) \tag{6.6}$$

In this case, each exemplar $G_i = (T, DST_{\tau-1}, A_{\tau-2}, U_{\tau-2}, A_{\tau-1}, U_{\tau-1}, A_\tau, U_\tau, H_\tau, TLB_\tau)$ in the SGD dataset includes the schema table T for its respective domains.

6.3 Experiments

6.3.1 Datasets and Evaluation

As in the previous chapter, we experiment with MultiWOZ 2.4 [Ye et al., 2022], which is the lastest version of MultiWOZ and SGD, which has more complicated dialogues and larger schema. We consider low-resource settings in which we use 5% training set as the training data, the same subset used in Chapter 5.

We report both turn-level scores (denoted as TLB) and dialogue-level scores (denoted as DST). For both types of scores, we report JGA following previous works. Since JGA assigns a score only when all slot values are correct, we also report F1 scores to reflect the degree of correctness by matching subsets of the

slot values.

6.3.2 Experimental Setting

We use LLama3 8B[4] as the backbone SLM. We also use GPT-4o as the LLM for comparisons. We choose IC-DST 2.0 as the in-context learning DST framework. We use k=10 examples for training and inference with correction_SLM for MultiWOZ and k=3 for SGD[5]. The dialogue retriever following [Hu et al., 2022] is initialized with SenBERT (all-mpnet-base-v2) and finetuned with the 5% training set. Both vanilla in-context learning and correction-augmented in-context learning use the same retriever.

6.3.3 Baselines

Single Pass ICL

We compare to a baseline that uses the same first pass inference model without any corrections.

In-Context Correction Baseline (non-finetuned)

We build a baseline by prompting the same non-finetuned SLM with the inputs from equation 6.1, equation 6.2, equation 6.5 and equation 6.6. except that the prompt format for the CORRECTIONLM by includes a task description to instruct the LM to make corrections.

Correction Baseline (Finetuned)

To demonstrate the effectiveness of correction through in-context examples, we build a baseline by finetuning SLM with prompts from equation 6.3 and equation 6.4 excluding the in-context examples $G_{1:k}$. After finetuning, the model is then prompted with the first-pass and second-pass prompts during the inference stage, again excluding the in-context examples.

[4]https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct

[5]This is due to the memory constraint during training Llama. SGD has more turns and schema tables therefore less examples can be fit into the prompt.

Inference Model on Train	Inference Model on Test	Correction Model	DST JGA/F1	TLB JGA/F1
Single Pass ICL Baselines				
N/A	Llama3-8B	N/A	40.10 / 87.11	72.32 / 81.43
N/A	GPT-4o	N/A	47.96 / 90.32	81.69 / 87.23
In-Context Correction Baselines (Non-Finetuned)				
Llama3-8B	Llama3-8B	Llama3-8B	41.54 / 88.27	71.19 / 82.61
GPT-4o	GPT-4o	GPT-4o	55.60 / 92.64	84.35 / 89.83
Correction Baseline (Finetuned)				
Llama3-8B	Llama3-8B	Correction-Tuned Llama	52.42 / 91.37	81.73 / 88.45
CORRECTIONLM				
Llama3-8B	Llama3-8B	Correction-Tuned Llama	56.20 / 92.97	83.44 / 89.71

Table 6.1: Results on MultiWOZ 2.4. We use 5% training set as training data. We report in-context correction baseline with both Llama3-8B and GPT-4o. Our proposed correction model was in-context finetuned with self-generated errors and gold labels.

6.4 Results and Analysis

6.4.1 Impact of Correction Finetuning

The results for MultiWOZ are presented in Table 6.1. When using in-context demonstrations with predictions and gold labels, a non-finetuned Llama shows limited correction capabilities. In contrast, GPT-4o proves to be highly effective in correcting mistakes. Fine-tuning with CORRECTIONLM, Llama demonstrates a further substantial improvement when training with in-context exemplars, achieving over 40% relative gain in DST JGA and a 15% relative gain in TLB JGA compared to IC-DST 2.0 baseline. Here, these results even surpass those of GPT-4o (inference and correction by GPT-4o), highlighting the effectiveness of our proposed framework. The results for the SGD dataset are shown in Table 6.2. Here, the non-finetuned Llama performs reasonably well in self-correction compared to its performance on MultiWOZ. We hypothesize that this is due to the significant improvements in handling challenging out-of-domain examples by leveraging semantically relevant in-context examples $G_{1:k}$ with corrections. Once again, fine-tuning Llama with our proposed CORRECTIONLM yields significant additional performance gains.

6.4.2 Comparisons with ORCHESTRALLM

We compare CORRECTIONLM to previous state-of-the-art low resource works. The MultiWOZ results are shown in table 6.3. As we discussed in the previous chapter, RefPyDST is the best system in few-shot settings, but their system requires sampling five candidate dialogue states which significantly increases the

Inference Model on Train	Inference Model on Test	Correction Model	DST JGA/F1	TLB JGA/F1
Single Pass ICL Baseline				
N/A	Llama3-8B	N/A	14.58 / 53.58	51.86 / 59.62
In-Context Correction Baseline (non-finetuned)				
Llama3-8B	Llama3-8B	Llama3-8B	23.17 / 72.96	54.99 / 66.02
CORRECTIONLM				
Llama3-8B	Llama3-8B	Correction-tuned Llama	34.14 / 80.56	73.08 / 81.20

Table 6.2: Results on SGD. We use 5% training set as training data. We report in-context correction baseline with both Llama3-8B. Our proposed correction model was in-context finetuned with self-generated errors and gold labels.

System	Backbone LMs	Training Data	TeraFLOPs	DST JGA	TLB JGA
Previous State-of-the-art					
IC-DST 2.0	GPT-3.5-Turbo (175B)	5%	22M	49.68	78.21
DS2 [Shin et al., 2022]	T5 (220M)	5%	N/A	49.89	N/A
ORCHESTRALLM-SenBERT	T5 (60M), GPT-3.5-Turbo (175B)	5%	8.8M	50.19	80.74
ORCHESTRALLM-best∗	T5 (60M), GPT-3.5-Turbo (175B)	5%	8.3M	52.68	82.46
RefPyDST	Codex (175B)	5%	110M (?)	62.30	N/A
Ours					
CORRECTIONLM	Llama3 (8B)	5%	1.7M	56.20	83.44

Table 6.3: Full results on MWOZ 2.4 comparing CORRECTIONLM to the previous state-of-the-art on low-resource DST: IC-DST 2.0, DS2, ORCHESTRALLM, and RefPyDST. The best setting in ORCHESTRALLM uses both task-aware and expert-aware objectives to finetune the retriever as described in the previous chapter. ? marks the estimated TeraFLOPs for RefPyDS by assuming prompting a 175B GPT3-style model with 4096 tokens (same as IC-DST 2.0) five times per turn.

computational costs. Furthermore, Codex is not an available API model anymore. Compared to ORCHESTRALLM using 5% training data, we see that CORRECTIONLM with only 5% training data can achieve better performance in all metrics and requires much less computation with the trade-off of having higher latency associated with two inference passes. Similar trends for SGD can be observed in Table 6.4.

6.4.3 Finetuned SLM can Correct LLM errors

To evaluate whether an SLM can correct errors produced by a more advanced LLM, we present MultiWOZ results for correcting GPT-4o outputs in Table 6.5. Due to the high cost of GPT-4o, we use 1% training data and 10% testing data. When fine-tuned on GPT-4o predictions, is able to make corrections at a level comparable to GPT-4o itself. Even when Llama is fine-tuned on its own predictions, it still produces high-quality corrections, giving a result similar to the GPT-4o model. Although the small test set size means the

System	Backbone LMs	Training Data	TeraFLOPs	DST JGA	TLB JGA
IC-DST 2.0	GPT-3.5-Turbo (175B)	5%	121M	33.15	63.86
ORCHESTRALLM	T5 (220M), GPT-3.5-Turbo (175B)	5%	60.50M	32.75	65.97
CORRECTIONLM	Llama3 (8B)	5%	9.3M	34.14	73.08

Table 6.4: Full results on SGD comparing CORRECTIONLM to the previous state-of-the-art on low-resource DST: IC-DST 2.0, ORCHESTRALLM.

Inference Model on Train	Inference Model on Test	Correction Model	DST JGA/F1	TLB JGA/F1
Single Pass ICL Baselines				
N/A	GPT-4o	N/A	42.40 / 89.40	70.17 / 75.01
In-Context Correction Baselines (non-finetuned)				
GPT-4o	GPT-4o	Llama3 8B	42.81 / 87.55	75.10 / 85.71
GPT-4o	GPT-4o	GPT-4o	55.26 / 92.46	83.99 / 90.11
CORRECTIONLM				
GPT-4o	GPT-4o	Finetuned on GPT-4o errors	55.51 / 92.79	84.67 / 90.26
GPT-4o	GPT-4o	Finetuned on Llama errors	54.71 / 92.65	82.90 / 90.90

Table 6.5: GPT4o correction Results on MultiWOZ 2.4. We use 1% training set as training data and 10% testing set as testing data. For the cost of GPT-4o, these diagnostic experiments are only run on 10% of the testing data. The second to the last row is a Llama3 8B model finetuned with GPT-4o generations and the last row is a Llama3 8B model finetuned with Llama generations.

differences are not significant, this indicates there is potential for developing a correction model for LLMs using SLM supervision, offering a much more cost-effective alternative to using LLMs for the same purpose. In addition, it will be important to explore this question with more difficult tasks. While JGA is still low, it is a pessimistic measure. Looking at DST F1 scores, all correction models do well on MultiWOZ (93% F1). On SGD, the correction LM DST F1 is lower (81%) but the benefits of correction are much greater.

6.5 Summary

In this chapter, we propose CORRECTIONLM, a self-correction framework that efficiently finetunes a SLM with correction-augmented in-context exemplars. Experimental results show that CORRECTIONLM can make successful refinements over a strong LLM-based in-context learning baseline and improves 40% relative DST JGA for MultiWOZ. When compared to the baseline SLM, the improvements are much greater, particularly for the more challenging SGD data set. CORRECTIONLM also outperformed ORCHESTRALLM while further reducing computational cost.

Chapter 7

Conclusion

7.1 Summary and Contributions

In this thesis, we propose solutions to address key challenges towards building a task-oriented agent that can extract users' task-execution needs given limited annotated data and minimizing computation costs. Our contributions to DST and to the two challenges we target are summarized as follows.

7.1.1 Language Model for DST

Many fine-tuning-based DST systems employ task-specific modules, complicating model training and increasing the demand for model parameters. In this thesis, we first introduced SDP-DST [Lee et al., 2021], a prompt-based finetuning framework built upon SLMs that alleviates the need for additional task-specific architectures (Chapter 3). By deriving prompts from the schema, this framework facilitates task-aware contextualization, enhancing the performance of language model-based systems while requiring fewer model parameters. SDP-DST set a new state-of-the-art benchmark on the MultiWOZ dataset. This work was among the first prompting language model approaches to DST. It also represented a more computationally efficient alternative to the large language models that would soon be applied instead.

7.1.2 Data Efficiency

We introduced IC-DST, an in-context learning framework based on LLMs to address the dependency of DST models on large amounts of annotated conversations (Chapter 4). IC-DST leverages recent developments in ICL where relevant examples are retrieved from a small training set to serve as examples, and achieved good results for the DST task by introducing a new representation of dialogue context and output prediction format. Experimental results demonstrate that IC-DST advanced the state-of-the-art in few-shot settings on the MultiWOZ dataset. As LLMs continue to improve, some aspects of the model have changed, but the basic structure and prompting framework continue to be useful.

7.1.3 Balancing Data and Computational Efficiency

In SDP-DST, we developed a computation-efficient DST system using the full training dataset. In IC-DST, we focused on building a data-efficient DST system with a small amount of training data. Building on these approaches, we proposed methods that utilize only a small amount of training data while also employing smaller LMs. We first proposed ORCHESTRALLM, a novel routing framework that leverages both SLMs and LLMs (Chapter 5). During inference, based on their semantic embedding distances to retrieve LM exemplars, a dynamic router directs instances to either the SLM or LLM, thus optimizing the strengths of both models. Evaluations on the MultiWOZ and SGD datasets in a limited data setting demonstrated that ORCHESTRALLM outperformed LLM-based systems and reduces computational costs by over 50%.

While LLMs have proven effective in correcting erroneous predictions, they typically consume significant computational resources. To mitigate this, we introduce CORRECTIONLM, a refinement framework that efficiently finetunes an SLM to make corrections using in-context exemplars during inference (Chapter 6), taking advantage of advances in SLMs. Unlike previous approaches, CORRECTIONLM does not require an LLM to provide textual feedback or produce improvement demonstrations. Experimental results on the MultiWOZ and SGD dataset in a limited data setting indicate that CORRECTIONLM improves JGA scores by 40% relative over a strong in-context learning baseline adapted from IC-DST. CORRECTIONLM improved performance over ORCHESTRALLM with a substantial further reduction in computation, though at the cost of added latency associated with the two-pass inference process.

7.2 Future Directions

Broader scope of application.

In this thesis, we propose solutions for building effective DST systems tailored for virtual assistant conversations, typically connecting with backend databases. However, the potential applications of DST systems extend far beyond virtual assistants. For instance, users may need to interact with local file systems, online functional APIs, or other specialized knowledge bases. Expanding the scope of DST systems to address these diverse scenarios presents research opportunities. Future work could focus on developing dialogue agents capable of interacting seamlessly with a wide variety of knowledge sources and tools, thereby enabling users to accomplish a broader range of tasks.

The routing framework developed in Chapter 5 has the potential to benefit other applications that involve structured prediction tasks, offering an efficient and effective approach. Similarly, the correction framework developed in Chapter 6 could be extended to other tasks beyond DST, providing a method for improving accuracy and performance in a variety of applications where correcting model predictions is critical.

Adapting to Evolving Language Models.

As the definitions of 'small' and 'large' language models evolve, earlier contributions may lose relevance as today's small models surpass the capabilities of what were once considered large models. This shift requires continual reassessment and adaptation of methods to leverage increasingly advanced language models. Future research must remain flexible, updating strategies for data efficiency, computational efficiency, and deployment to ensure that task-oriented dialogue systems take advantage of a rapidly changing landscape.

Robust Agents.

In real-world applications, schemas are subject to change over time due to a variety of reasons, such as evolving set of entities and entity attributes and the introduction of new services. Therefore, it is crucial to build robust and flexible agents that can transfer knowledge across different contexts without the need for extensive retraining. In Chapter 4, we show that IC-DST still has room for improvement in few-shot settings on the MultiWOZ dataset. In Chapter 5, experimental results on the SGD dataset reveal that IC-

DST 2.0 based on LLM performs relatively worse when there is a mismatch between the training and testing domains. This indicates that effective knowledge transfer to new services remains a critical challenge for DST systems. Future studies could focus on building stronger, more transferable systems that can maintain high performance across diverse and evolving domains.

Continuous Improvement Through Online Learning.

In real-world development, dialogue agents should be continuously updated to better serve users. One of the most crucial skills for these agents is the ability to learn from their previous mistakes. In Chapter 6, we demonstrate that finetuning an SLM enables it to refine its predictions. This approach could be extended to an online learning setup, where errors from new conversations are leveraged to iteratively improve the dialogue agents. By implementing an online learning framework, agents can continuously adapt and enhance their performance, ensuring they remain effective in dynamic and changing environments.

Personalized agents.

In the context of task-oriented dialogue state tracking, the development of personalized agents represents an important future direction. Personalized agents can adapt to individual users' feedback, objectives, and habitual tool usage, thereby creating a more tailored and pleasant user experience. The agents must dynamically refine its strategies based on positive or negative feedback from the environment and the interaction with the user. This adaptability is crucial for handling tasks that are new or heavily dependent on user-specific preferences. Furthermore, the agents should store and utilize experiences, including the user's preferences and habits, to support context references across multiple conversations and services. Future research can pave the way for personalized dialogue agents that not only fulfill users' task-oriented needs but also enhance their overall experience through personalized interactions.

Responsible task-oriented agents.

Task-oriented agents have the potential for misuse if not properly monitored and equipped with robust guardrails. Ensuring that these agents operate responsibly is critical, especially in areas such as healthcare and financial services, where safety and ethical considerations are paramount. Agents must learn to balance

user requests with their own programmed principles to prevent harmful or unethical actions. For example, in healthcare, an agent should never provide medical advice beyond its certification, and in financial services, it should avoid actions that could lead to fraud or financial loss.

Future research should focus on developing advanced monitoring systems and ethical frameworks that enable task-oriented agents to make responsible decisions autonomously. This includes implementing real-time feedback mechanisms that alert human supervisors to potential issues, ethical reasoning capabilities that allow agents to evaluate the implications of their actions, and strict compliance with safety protocols, such as data privacy regulations and industry-specific standards. By integrating these elements, we can create task-oriented agents that perform effectively while upholding the highest standards of safety and ethics.

Bibliography

Namo Bang, Jeehyun Lee, and Myoung-Wan Koo. 2023. Task-optimized adapters for an end-to-end task-oriented dialogue system. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7355–7369.

Jeff Barr. 2019. Amazon EC2 update. Blog Post. Available at: `https://aws.amazon.com/blogs/aws/amazon-ec2-update-inf1-instances-with-aws-inferentia-chips-for-high-performance-cost-effective-inferencing/`.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow.

Dan Bohus and Alexander Rudnicky. 2006. A" k hypothteses+ other" belief updating model. In *Proc. AAAI Workshop on Statistical and Empirical Approaches to Spoken Dialogue Systems, 2006*.

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

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

Paweł Budzianowski and Ivan Vulić. 2019. Hello, it's GPT-2 - how can I help you? towards the use of

pretrained language models for task-oriented dialogue systems. *Proceedings of the 3rd Workshop on Neural Generation and Translation*.

Guan-Lin Chao and Ian Lane. 2019. BERT-DST: Scalable End-to-End Dialogue State Tracking with Bidirectional Encoder Representations from Transformer. In *Proc. Interspeech 2019*, pages 1468–1472.

Bei Chen, Fengji Zhang, Anh Nguyen, Daoguang Zan, Zeqi Lin, Jian-Guang Lou, and Weizhu Chen. 2023a. CodeT: Code generation with generated tests. In *International Conference on Learning Representations*.

Derek Chen, Howard Chen, Yi Yang, Alexander Lin, and Zhou Yu. 2021a. Action-based conversations dataset: A corpus for building more in-depth task-oriented dialogue systems. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3002–3017.

Lingjiao Chen, Matei Zaharia, and James Zou. 2023b. FrugalGPT: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*.

Lu Chen, Boer Lv, Chi Wang, Su Zhu, Bowen Tan, and Kai Yu. 2020. Schema-guided multi-domain dialogue state tracking with graph attention neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7521–7528.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021b. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Yanda Chen, Ruiqi Zhong, Sheng Zha, George Karypis, and He He. 2022. Meta-learning via language model in-context tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 719–730.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. GPT3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*, 35:30318–30332.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2024. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Suvodip Dey, Ramamohan Kummara, and Maunendra Desarkar. 2022. Towards fair evaluation of dialogue state tracking by flexible incorporation of turn-level performances. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 318–324.

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

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyag Gao, and Dilek Hakkani-Tur. 2019. Multiwoz 2.1: Multi-domain dialogue state corrections and state tracking baselines. *arXiv preprint arXiv:1907.01669*.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, Adarsh Kumar, Anuj Goyal, Peter Ku, and Dilek Hakkani-Tur. 2020. Multiwoz 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 422–428.

Angela Fan, Edouard Grave, and Armand Joulin. 2020. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations*.

Yue Feng, Yang Wang, and Hang Li. 2020. A sequence-to-sequence approach to dialogue state tracking. *arXiv preprint arXiv:2011.09553*.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830.

Saurabh Goyal, Anamitra Roy Choudhury, Saurabh Raje, Venkatesan Chakaravarthy, Yogish Sabharwal, and Ashish Verma. 2020. Power-BERT: Accelerating BERT inference via progressive word-vector elimination. In *International Conference on Machine Learning*, pages 3690–3699. PMLR.

Yue Guan, Zhengyi Li, Jingwen Leng, Zhouhan Lin, and Minyi Guo. 2022. Transkimmer: Transformer learns to layer-wise skim. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7275–7286.

Donghoon Ham, Jeong-Gwan Lee, Youngsoo Jang, and Kee-Eung Kim. 2020. End-to-end neural pipeline for goal-oriented dialogue systems using gpt-2. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 583–592.

Wanwei He, Yinpei Dai, Min Yang, Jian Sun, Fei Huang, Luo Si, and Yongbin Li. 2022. Unified dialog model pre-training for task-oriented dialog understanding and generation. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 187–200.

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

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014. The second dialog state tracking challenge. In *Proceedings of the 15th annual meeting of the special interest group on discourse and dialogue (SIGDIAL)*, pages 263–272.

Ryuichiro Higashinaka, Mikio Nakano, and Kiyoaki Aikawa. 2003. Corpus-based discourse understanding

in spoken dialogue systems. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 240–247.

Ari Holtzman, Peter West, Vered Shwartz, Yejin Choi, and Luke Zettlemoyer. 2021. Surface form competition: Why the highest probability answer isn't always right. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7038–7051.

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

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

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

Jiaxin Huang, Shixiang Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2023. Large language models can self-improve. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1051–1068.

Vojtěch Hudeček and Ondřej Dušek. 2023. Are large language models all you need for task-oriented dialogue? In *Proceedings of the 24th Meeting of the Special Interest Group on Discourse and Dialogue*, pages 216–228.

Wiradee Imrattanatrai and Ken Fukuda. 2023. End-to-end task-oriented dialogue systems based on schema. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10148–10161.

Joel Jang, Seungone Kim, Seonghyeon Ye, Doyoung Kim, Lajanugen Logeswaran, Moontae Lee, Kyungjae Lee, and Minjoon Seo. 2023. Exploring the benefits of training expert language models over instruction tuning. In *International Conference on Machine Learning*, pages 14702–14729. PMLR.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174.

Anil Kag, Igor Fedorov, Aditya Gangrade, Paul Whatmough, and Venkatesh Saligrama. 2022. Efficient edge inference by selective query. In *The Eleventh International Conference on Learning Representations*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781.

John F Kelley. 1984. An iterative design methodology for user-friendly natural language office information applications. *ACM Transactions on Information Systems (TOIS)*, 2(1):26–41.

Gyuwan Kim and Kyunghyun Cho. 2021. Length-adaptive transformer: Train once with length drop, use anytime with search. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6501–6511.

Sungdong Kim, Sohee Yang, Gyuwan Kim, and Sang-Woo Lee. 2020. Efficient dialogue state tracking by selectively overwriting memory. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 567–582.

Takyoung Kim, Hoonsang Yoon, Yukyung Lee, Pilsung Kang, and Misuk Kim. 2022. Mismatch between multi-turn dialogue and its evaluation metric in dialogue state tracking. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 297–309.

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

François Lagunas, Ella Charlaix, Victor Sanh, and Alexander M Rush. 2021. Block pruning for faster transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10619–10629.

Teven Le Scao and Alexander M Rush. 2021. How many data points is a prompt worth? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2627–2636.

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

Chia-Hsuan Lee, Hao Cheng, and Mari Ostendorf. 2024. Orchestrallm: Efficient orchestration of language models for dialogue state tracking. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1434–1445.

Wenqiang Lei, Xisen Jin, Min-Yen Kan, Zhaochun Ren, Xiangnan He, and Dawei Yin. 2018. Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1437–1447.

George Leopold. 2019. AWS to offer NVIDIA's T4 GPUs for AI inferencing. `https://www.hpcwire.com/2019/03/19/aws-upgrades-its-gpu-backed-ai-inference-platform/`. Accessed: 2021-06-01.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Shuyang Li, Jin Cao, Mukund Sridhar, Henghui Zhu, Shang-Wen Li, Wael Hamza, and Julian McAuley. 2021. Zero-shot generalization in dialog state tracking through generative question answering. *Proceed-*

ings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597.

Zekun Li, Zhiyu Zoey Chen, Mike Ross, Patrick Huber, Seungwhan Moon, Zhaojiang Lin, Xin Luna Dong, Adithya Sagar, Xifeng Yan, and Paul A Crook. 2024. Large language models as zero-shot dialogue state tracker through function calling. *arXiv preprint arXiv:2402.10466*.

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

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

Zhaojiang Lin, Andrea Madotto, Genta Indra Winata, and Pascale Fung. 2020. Mintl: Minimalist transfer learning for task-oriented dialogue systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3391–3405.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, William B Dolan, Lawrence Carin, and Weizhu Chen. 2022. What makes good in-context examples for GPT-3? In *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 100–114.

Weijie Liu, Peng Zhou, Zhiruo Wang, Zhe Zhao, Haotang Deng, and Qi Ju. 2020. Fastbert: a self-distilling bert with adaptive inference time. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6035–6044.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Zequan Liu, Jiawen Lyn, Wei Zhu, and Xing Tian. 2024. Alora: Allocating low-rank adaptation for fine-tuning large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 622–641.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Aman Madaan, Pranjal Aggarwal, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, et al. 2023. Automix: Automatically mixing language models. *arXiv preprint arXiv:2310.12963*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2024. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36.

Andrea Madotto, Zhaojiang Lin, Genta Indra Winata, and Pascale Fung. 2021. Few-shot bot: Prompt-based learning for dialogue systems. *arXiv preprint arXiv:2110.08118*.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? *Advances in neural information processing systems*, 32.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2022. Metaicl: Learning to learn in context. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2791–2809.

Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. 2022. A conversational paradigm for program synthesis. *arXiv preprint arXiv:2203.13474*.

Panupong Pasupat, Yuan Zhang, and Kelvin Guu. 2021. Controllable semantic parsing via retrieval augmentation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7683–7698.

Baolin Peng, Michel Galley, Pengcheng He, Chris Brockett, Lars Liden, Elnaz Nouri, Zhou Yu, Bill Dolan, and Jianfeng Gao. 2022. Godel: Large-scale pre-training for goal-directed dialog. *arXiv preprint arXiv:2206.11309*.

Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao. 2021. Soloist: Building task bots at scale with transfer learning and machine teaching. *Transactions of the Association for Computational Linguistics*, 9:807–824.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying lms with mixtures of soft prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5203–5212.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

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

Nitarshan Rajkumar, Raymond Li, and Dzmitry Bahdanau. 2022. Evaluating the text-to-SQL capabilities of large language models. *arXiv*, abs/2204.00498.

Abhinav Rastogi, Raghav Gupta, and Dilek Hakkani-Tur. 2018. Multi-task learning for joint language understanding and dialogue state tracking. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 376–384.

Abhinav Rastogi, Dilek Hakkani-Tür, and Larry Heck. 2017. Scalable multi-domain dialogue state tracking. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 561–568. IEEE.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020a. Schema-guided dialogue state tracking task at DSTC8. *arXiv preprint arXiv:2002.01359*.

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

Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2655–2671.

Marija Šakota, Maxime Peyrard, and Robert West. 2023. Fly-swat or cannon? cost-effective language model choice via meta-modeling. *arXiv preprint arXiv:2308.06077*.

Mohammadreza Salehi, Sachin Mehta, Aditya Kusupati, Ali Farhadi, and Hannaneh Hajishirzi. 2023. Sharcs: Efficient transformers through routing with dynamic width sub-networks. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10519–10532.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. 2022. Self-critiquing models for assisting human evaluators. *arXiv preprint arXiv:2206.05802*.

Timo Schick and Hinrich Schütze. 2021. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269.

Pararth Shah, Dilek Hakkani-Tur, Bing Liu, and Gökhan Tür. 2018a. Bootstrapping a neural conversational agent with dialogue self-play, crowdsourcing and on-line reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 41–51.

Pararth Shah, Dilek Hakkani-Tür, Gokhan Tür, Abhinav Rastogi, Ankur Bapna, Neha Nayak, and Larry Heck. 2018b. Building a conversational agent overnight with dialogue self-play. *arXiv preprint arXiv:1801.04871*.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468.

Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. 2020. Q-BERT: Hessian based ultra low precision quantization of BERT. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8815–8821.

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

Richard Shin and Benjamin Van Durme. 2022. Few-shot semantic parsing with language models trained on code. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5417–5425.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt:

Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. 2023. Distilling reasoning capabilities into smaller language models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7059–7073.

Yixuan Su, Lei Shu, Elman Mansimov, Arshit Gupta, Deng Cai, Yi-An Lai, and Yi Zhang. 2022. Multi-task pre-training for plug-and-play task-oriented dialogue system. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4661–4676.

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

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962*.

Qingyue Wang, Liang Ding, Yanan Cao, Yibing Zhan, Zheng Lin, Shi Wang, Dacheng Tao, and Li Guo. 2023. Divide, conquer, and combine: Mixture of semantic-independent experts for zero-shot dialogue state tracking. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2048–2061.

Ziheng Wang, Jeremy Wohlwend, and Tao Lei. 2020. Structured pruning of large language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6151–6162.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy

Liang, Jeff Dean, and William Fedus. 2022a. Emergent abilities of large language models. *Transactions on Machine Learning Research*. Survey Certification.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022b. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gasic, Lina M Rojas Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 438–449.

Jason D Williams, Antoine Raux, Deepak Ramachandran, and Alan W Black. 2013. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413.

Jason D Williams and Steve Young. 2007. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422.

Chien-Sheng Wu, Steven CH Hoi, Richard Socher, and Caiming Xiong. 2020a. TOD-BERT: Pre-trained natural language understanding for task-oriented dialogue. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 917–929.

Chien-Sheng Wu, Steven CH Hoi, and Caiming Xiong. 2020b. Improving limited labeled dialogue state tracking with self-supervision. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4462–4472.

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

Zeqiu Wu, Bo-Ru Lu, Hannaneh Hajishirzi, and Mari Ostendorf. 2021. DIALKI: Knowledge identification in conversational systems through dialogue-document contextualization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1852–1863.

Mengzhou Xia, Zexuan Zhong, and Danqi Chen. 2022. Structured pruning learns compact and accurate models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1513–1528.

Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I Wang, et al. 2022. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 602–631.

Ji Xin, Raphael Tang, Yaoliang Yu, and Jimmy Lin. 2021. Berxit: Early exiting for bert with better fine-tuning and extension to regression. In *Proceedings of the 16th conference of the European chapter of the association for computational linguistics: Main Volume*, pages 91–104.

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

Fanghua Ye, Jarana Manotumruksa, Qiang Zhang, Shenghui Li, and Emine Yilmaz. 2021. Slot self-attentive dialogue state tracking. In *Proceedings of the Web Conference 2021*, pages 1598–1608.

Seonghyeon Ye, Yongrae Jo, Doyoung Kim, Sungdong Kim, Hyeonbin Hwang, and Minjoon Seo. 2023. SelFee: Iterative self-revising llm empowered by self-feedback generation. Blog post. Available at: `https://kaistai.github.io/SelFee/`.

Steve J Young. 2000. Probabilistic methods in spoken–dialogue systems. *Philosophical Transactions of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 358(1769):1389–1402.

Tao Yu, Rui Zhang, Alex Polozov, Christopher Meek, and Ahmed Hassan Awadallah. 2020. Score: Pre-training for context representation in conversational semantic parsing. In *International Conference on Learning Representations*.

Xiao Yu, Baolin Peng, Michel Galley, Jianfeng Gao, and Zhou Yu. 2024a. Teaching language models to self-improve through interactive demonstrations. In *Proceedings of the 2024 Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 5127–5149.

Xiao Yu, Baolin Peng, Michel Galley, Jianfeng Gao, and Zhou Yu. 2024b. Teaching language models to self-improve through interactive demonstrations. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5127–5149, Mexico City, Mexico. Association for Computational Linguistics.

Xiaoxue Zang, Abhinav Rastogi, Srinivas Sunkara, Raghav Gupta, Jianguo Zhang, and Jindong Chen. 2020. Multiwoz 2.2: A dialogue dataset with additional annotation corrections and state tracking baselines. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 109–117.

Jianguo Zhang, Kazuma Hashimoto, Chien-Sheng Wu, Yao Wang, S Yu Philip, Richard Socher, and Caiming Xiong. 2020. Find or classify? dual strategy for slot-value predictions on multi-domain dialog state tracking. In *Proceedings of the Ninth Joint Conference on Lexical and Computational Semantics*, pages 154–167.

Jianguo Zhang, Kun Qian, Zhiwei Liu, Shelby Heinecke, Rui Meng, Ye Liu, Zhou Yu, Huan Wang, Silvio Savarese, and Caiming Xiong. 2024a. Dialogstudio: Towards richest and most diverse unified dataset collection for conversational ai. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 2299–2315.

Yunxiang Zhang, Muhammad Khalifa, Lajanugen Logeswaran, Jaekyeom Kim, Moontae Lee, Honglak Lee, and Lu Wang. 2024b. Small language models need strong verifiers to self-correct reasoning. *arXiv preprint arXiv:2404.17140*.

Li Zhou and Kevin Small. 2019. Multi-domain dialogue state tracking as dynamic knowledge graph enhanced question answering. *arXiv preprint arXiv:1911.06192*.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian McAuley, Ke Xu, and Furu Wei. 2020. Bert loses patience:

Fast and robust inference with early exit. *Advances in Neural Information Processing Systems*, 33:18330–18341.

Appendix A

Appendix for IC-DST

A.1 Implementation Details and Prompt Examples for IC-DST

A.1.1 Prompt Examples

Prompt in Few-Shot Settings

Below is the full version of the prompt for few shot settings. Here we are including 5 examples that are retrieved by our finetuned retriever from a 5% subset of MultiWOZ training set. Notice that the test instance is preceded by "Example #6" but the label needs to be completed by the LM. The completion of Codex is at the end.

```
CREATE TABLE hotel(
  name text,
  pricerange text CHECK (pricerange IN (dontcare, cheap, moderate, expensive)),
  type text CHECK (type IN (hotel, guest house)),
  parking text CHECK (parking IN (dontcare, yes, no)),
  book_stay int,
  book_day text,
  book_people int,
  area text CHECK (area IN (dontcare, centre, east, north, south, west)),
  stars int CHECK (stars IN (dontcare, 0, 1, 2, 3, 4, 5)),
  internet text CHECK (internet IN (dontcare, yes, no))
)
/*
4 example rows:
SELECT * FROM hotel LIMIT 4;
name  pricerange  type  parking book_stay book_day  book_people area  stars internet
a and b guest house moderate  guest house  dontcare  3 friday  5 east  4 yes
ashley hotel  expensive hotel yes 2 thursday  5 north 5 yes
el shaddia guest house  cheap guest house  yes 5 friday  2 centre  dontcare  no
```

```
express by holiday inn cambridge  dontcare  guest house yes 3 monday  2 east  dontcare  no
*/

CREATE TABLE train(
  destination text,
  departure text,
  day text,
  book_people int,
  leaveat text,
  arriveby text
)
/*
3 example rows:
SELECT * FROM train LIMIT 3;
destination departure day book_people leaveat arriveby
london kings cross  cambridge monday  6 dontcare 05:51
cambridge stansted airport  dontcare  1 20:24 20:52
peterborough  cambridge saturday  2  12:06  12:56
*/

CREATE TABLE attraction(
  name text,
  area text CHECK (area IN (dontcare, centre, east, north, south, west)),
  type text,
)
/*
4 example rows:
SELECT * FROM attraction LIMIT 4;
name area type
abbey pool and astroturf pitch  centre  swimming pool
adc theatre centre  theatre
all saints church dontcare  architecture
castle galleries  centre  museum
*/

CREATE TABLE restaurant(
  name text,
  food text,
  pricerange text CHECK (pricerange IN (dontcare, cheap, moderate, expensive)),
  area text CHECK (area IN (centre, east, north, south, west)),
  book_time text,
  book_day text,
  book_people int
)
/*
5 example rows:
SELECT * FROM restaurant LIMIT 5;
name  food  pricerange  area  book_time book_day  book_people
pizza hut city centre italian dontcare centre  13:30 wednesday 7
the missing sock  international moderate  east  dontcare dontcare  2
golden wok chinese moderate north 17:11 friday 4
cambridge chop house  dontcare  expensive  center 08:43 monday  5
darrys cookhouse and wine shop  modern european expensive center  11:20 saturday  8
*/
```

```
CREATE TABLE taxi(
  destination text,
  departure text,
  leaveat text,
  arriveby text
)
/*
3 example rows:
SELECT * FROM taxi LIMIT 3;
destination departure leaveat arriveby
copper kettle royal spice 14:45 15:30
magdalene college  university arms hotel dontcare  15:45
lovell lodge  da vinci pizzeria 11:45 dontcare
*/

-- Using valid SQLite, answer the following multi-turn conversational questions for the tables provided above.

Example #1
[context] attraction-area: centre, attraction-type: museum, train-departure: cambridge, train-day: friday, train-arrive_by_time:
     12:45, train-book people: 6, train-destination: leicester
[system] i recommend castle galleries located at unit su43 , grande arcade , saint andrews street . their phone number is
     01223307402 . is there anything else i can help you with ?
Q: [user] excellent , can you give me the postcode ?
SQL: SELECT * FROM attraction WHERE name = castle galleries;


Example #2
[context] attraction-type: museum, restaurant-book day: wednesday, restaurant-book people: 7, restaurant-name: loch fyne,
     restaurant-book time: 16:30, attraction-area: west
[system] i would suggest cafe jello gallery located at cafe jello gallery , 13 magdalene street . they have free entry .
Q: [user] okay great ! what is their phone number please ?
SQL: SELECT * FROM attraction WHERE name = cafe jello gallery;


Example #3
[context] attraction-area: centre, attraction-type: museum
[system] the broughton house gallery is in the centre , and it has no entrance fee .
Q: [user] may i have the telephone number please ?
SQL: SELECT * FROM attraction WHERE name = broughton house gallery;


Example #4
[context] train-arrive_by_time: 21:30, train-destination: leicester, train-day: thursday, train-departure: cambridge
[system] how many tickets please ?
Q: [user] i do not need to make the reservation now . thank you though . i would like the address for cambridge contemporary art
     please .
SQL: SELECT * FROM attraction WHERE name = cambridge contemporary art;


Example #5
[context] attraction-area: east
[system] i like the cambridge artworks it s a museum at 5 greens road and it has free admission .
Q: [user] that sounds , good , what is the postcode ?
SQL: SELECT * FROM attraction WHERE name = cambridge artworks;
```

```
Example #6
[context] attraction-area: east
[system] how about cambridge artworks ? it s a museum on the east side of town , and they have no entrance fee .
Q: [user] that sounds great . what s their address and postcode ?
SQL: SELECT * FROM
--------------------Prompt Ends here!---------------------
LM completion:  attraction WHERE name = cambridge artworks
```

Prompt Example for Traditional Format

Below is the full version of the traditional format prompt following Hosseini-Asl et al. [2020] for few shot settings. The ontology is listed in a similar way to the dialogue states representation. Here we are including 5 examples that are retrieved by our finetuned retriever from a 5% subset of MultiWOZ training set. Notice that the test instance is preceded by "Example #6" but the label needs to be completed by the LM. The completion of Codex is at the end.

```
hotel-name: a and b guest house, ashley hotel, el shaddia guest house, etc.
hotel-pricerange: dontcare, cheap, moderate, expensive
hotel-type: hotel, guest house
hotel-parking: dontcare, yes, no
hotel-book_stay: 1, 2, 3, etc.
hotel-book_day: monday, tuesday, etc.
hotel-book_people: 1, 2, 3, etc.
hotel-area: dontcare, centre, east, north, south, west
hotel-stars: dontcare, 0, 1, 2, 3, 4, 5
hotel-internet: dontcare, yes, no

train-destination: london kings cross, cambridge, peterborough, etc.
train-departure: cambridge, stansted airport, etc.
train-day: monday, saturday, etc.
train-book_people: 1, 2, 3, etc.
train-leaveat: 20:24, 12:06, etc.
train-arriveby: 05:51, 20:52, etc.

attraction-name: abbey pool and astroturf pitch, adc theatre, all saints church, castle galleries, etc.
attraction-area: dontcare, centre, east, north, south, west
attraction-type: architecture, boat, church, cinema, college, concert hall, entertainment, hotspot, multiple sports, museum, nightclub, park, special,

restaurant-name: pizza hut city centre, the missing sock, golden wok, cambridge chop house, darrys cookhouse and wine shop, etc.
restaurant-food: italian, international, chinese, dontcare, modern european, etc.
restaurant-pricerange: dontcare, cheap, moderate, expensive
restaurant-area: centre, east, north, south, west
restaurant-book_time: 13:30, 17:11, etc.
restaurant-book_day: wednesday, friday, etc.
restaurant-book_people: 1, 2, 3, etc.

taxi-destination: copper kettle, magdalene college, lovell lodge
taxi-departure: royal spice, university arms hotel, da vinci pizzeria
```

```
taxi-leaveat: 14:45, 11:15, etc.
taxi-arriveby: 15:30, 12:45, etc.

-- answer the following multi-turn conversational questions for the ontology provided above.

Example #1
[context]
[system]
Q: [user] find me a restaurant that serves belgian food in the centre
A: restaurant-area: centre, restaurant-food: belgian;


Example #2
[context]
[system]
Q: [user] i need a place to dine on indian food . centre of the town please .
A: restaurant-area: centre, restaurant-food: indian;


Example #3
[context] restaurant-food: italian, restaurant-area: centre
[system] sure , did you want someone in a certain price range ?
Q: [user] no , price does not matter .
A: restaurant-pricerange: dontcare;


Example #4
[context]
[system]
Q: [user] hello , i am looking for a venetian restaurant in the centre of town .
A: restaurant-area: centre, restaurant-food: venetian;


Example #5
[context]
[system]
Q: [user] hello ! i would like to get some italian food , somewhere in the center of town .
A: restaurant-food: italian, restaurant-area: centre;


Example #6
[context]
[system]
Q: [user] i am looking for a italian restaurant centre .
A:
-----------------------Prompt Ends here!--------------------
LM completion:  restaurant-area: centre, restaurant-food: italian
```

A.1.2 Computing Details

For zero-shot inference and few-shot in-context learning, we do not need any training. We use one NVIDIA A10G graphics card for GPT-Neo and CodeGen models. Inference of one turn takes about 2 seconds for both models. For Codex, we use the OpenAI API. On average each turn takes about 4 seconds for inference.

For retriever finetuning, we train all our retrievers on one single NVIDIA A10G graphics card. For 5% few-shot scenario, each epoch of training takes 10 minutes on average with our given hyperparameters, and we train for 10 epochs. For 100% full-shot scenario, we train for 2 epochs, which takes 6 hours. The training time is linear to the data size.

Only retriever finetuning requires hyperparameter tuning. Hyperparameters are given in our code, and the best configuration is provided in the experiment section. They are chosen by manual tuning. The development evaluation metrics is the average F_1 score of retrieved examples, which has been described in the paper. For learning rates, we experimented over 1×10^{-5} and 2×10^{-5}. For contrastive learning, we scale the number of positive/negative examples for each data point linearly with the selection pool size. When using 1% of labeled data, the number of positive/negative examples is 2. For 5% of data, the number is 10. We also experimented with doubling the number of positive/negative examples. All few-shot test numbers are the average of three evaluation runs, and all retrievers are trained just once with each hyperparameter configuration.