In-Context Learning for Few-Shot Dialogue State Tracking

Yushi Hu♠ **Chia-Hsuan Lee**♠ **Tianbao Xie**◇
Tao Yu◇♠ **Noah A. Smith**♠† **Mari Ostendorf**♠

♠University of Washington
◇University of Hong Kong
†Allen Institute for Artificial Intelligence
{yushihu,chiahlee,ostendor}@washington.edu

Abstract

Collecting and annotating task-oriented dialogues is time-consuming and costly; thus, zero and few shot learning could greatly benefit dialogue state tracking (DST). In this work, we propose an in-context learning (ICL) framework for zero-shot and few-shot learning DST, where a large pre-trained language model (LM) takes a test instance and a few exemplars as input, and directly decodes the dialogue state *without any parameter updates*. To better leverage a tabular domain description in the LM prompt, we reformulate DST into a text-to-SQL problem. We also propose a novel approach to retrieve annotated dialogues as exemplars. Empirical results on MultiWOZ show that our method IC-DST substantially outperforms previous fine-tuned state-of-the-art models in few-shot settings. In addition, we test IC-DST in zero-shot settings, in which the model only takes a fixed task instruction as input, finding that it outperforms previous zero-shot methods by a large margin.[1]

1 Introduction

Dialogue state tracking (DST) is an important module in many task-oriented dialogue systems. The goal of this module is to extract users' intentions at each turn in the dialogue as represented in the slot values of a predefined schema. 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., 2020; Li et al., 2021; Su
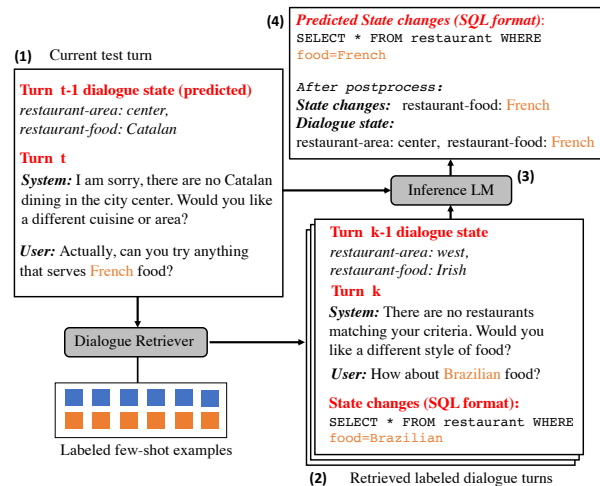

Figure 1: Illustration of DST task and IC-DST approach. The task is to track the slot values associated with a user request up to the current turn (dialogue state). In few-shot settings, given a test turn (1), IC-DST first retrieves a few most similar turns from the labeled dialogues as examples (2). 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).

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 is that it requires no fine-tuning (i.e., no parameter updates), which makes

---

[1]Our code : https://github.com/Yushi-Hu/IC-DST

2627

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 pretrained with code: Codex (Chen et al., 2021), 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. We also substantially improve the zero-shot state of the art by 10–30% absolute accuracy on each domain. A further contribution is an extensive analysis demonstrating impact from each innovation.

In summary, our work 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 zero/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.

2 DST System Design

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,[2] 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 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 1. Possible state changes include slot addition,

---

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

User: I would like to find a particular restaurant in Cambridge. The name of the restaurant is restaurant 2 two. Could you give me the location ?

System: Restaurant 2 two is nice French restaurant located at 22 Chesterton road. Would like me to book you a table?

User: That would be great. I need it for 8 on Friday.

System: Do you have a time preference ?

User: Yes at 11:15 if that's not available I can do 10:15

Previous states + single turn

Context: restaurant-name: 2 two, restaurant-book day: Friday, Restaurant-book people: 8

System: Do you have a time preference ?

User: yes at 11:15 if that's not available I can do 10:15

Current turn dialogue state: restaurant-name: 2 two, restaurant-book day: Friday, restaurant-book people: 8, restaurant-book time: 11:15
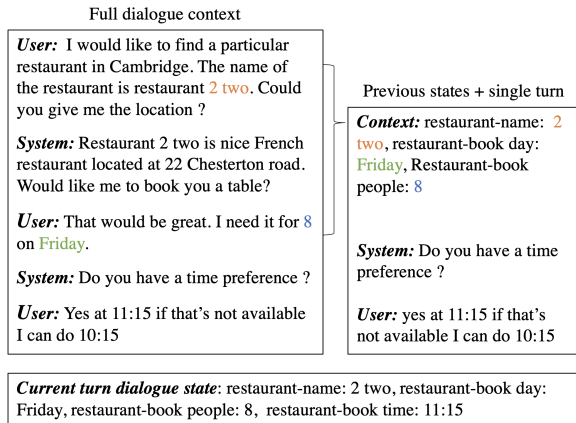
Figure 2: Illustration of the dialogue context representation: the full dialogue context C_{t-1} before the current turn is replaced by the associated dialogue state y_{t-1}.

slot deletion, and slot value change. For example, in the current test turn of Figure 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 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.

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

Task schema

```
CREATE TABLE attraction(
  name text, area text, type text)
       ... ...
CREATE TABLE restaurant(
  name text, food text,......,area text)
```

In-context examples

Context: hotel-stars: 4, hotel-area: west
System: I was able to book one night.
User: I am also looking for a place to go in the same area.
SQL: SELECT * FROM attraction WHERE area = west

Test dialogue turn

Context: restaurant-book time: 15:30, restaurant-area: centre
System: Booking was successful. Anything else?
User: Could you also find me some places to go in the same area as the restaurant?
SQL:

Expected output

Inference LM →
```
SELECT * FROM attraction WHERE
area = centre
```
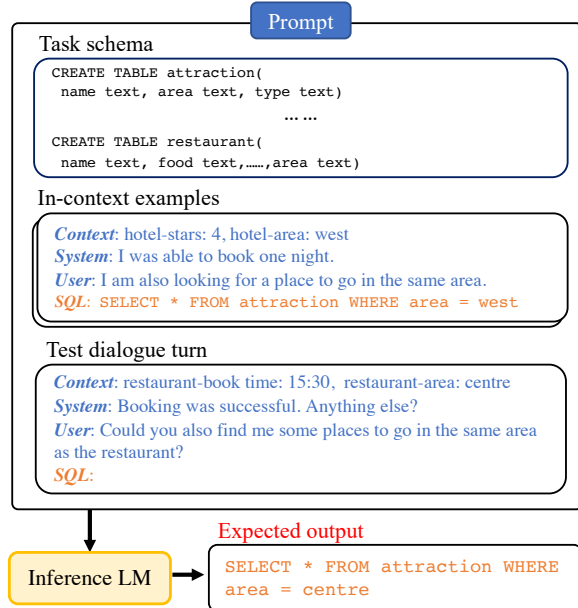
Figure 3: The prompt contains the task schema, in-context examples, and the current test dialogue turn.

to represent 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 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 $t-1$.

2.2 In-Context Learning

In-context learning is an alternative to finetuning that keeps pretrained language model parameters fixed (Brown et al., 2020). 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.

An overview of our IC-DST system is shown in Figure 1 for the few-shot setting. The details of our prompt are shown in Figure 3. The task description is the schema associated with the task ontology, and a retriever is used to select labeled example turns

```
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: The SQL table for the taxi domain schema. We follow the table prompt from Rajkumar et al. (2022).

from the training data. In the zero-shot setting, there is no retriever.

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. 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. In the zero-shot scenario, following previous work on zero-shot learning (Wang et al., 2022), the in-context example is a formatting example turn. We call this setting "zero shot" because the prompt is fixed by the system developer and does not use any labeled data. Prompt examples for the few-shot and zero-shot settings are given in Appendices A.1 and A.2, respectively.

2.3 Dialogue Retriever

In few-shot settings, successful prediction of in-context learning relies on the quality of the context examples. Usually this is accomplished by semantic retrieval using the testing input as the query. Previous studies have explored methods for building sentence-level retrievers (Poesia et al., 2022; Rubin et al., 2022). Our work goes beyond sentences to retrieving dialogue histories (contexts).

We want to retrieve example dialogue contexts that are relevant for the predicted state change of a test sample. Formally, suppose $X = \{(e_i, c_i)\}$

is a dataset of dialogue context e_i and corresponding state change c_i pairs. Each labeled turn in a dialogue is a candidate. Given a test dialogue context x, our goal is to retrieve k examples $\{(e_1, c_1), (e_2, c_2), \cdots, (e_k, c_k)\}$ such that $c_1, c_2, \cdots, c_k$ should be similar to the state changes associated with x.

Unsupervised retriever One approach is to use a pretrained embedding model with a cosine similarity score as the retriever. This approach does not need any DST data. Let M be an embedding model. For each test example x, we retrieve the k training examples such that $e_i, e_2, \cdots, e_k$ are the nearest neighbors of x^*, given by a similarity score,

$$score(x, e) = \cos(M(x), M(e)).$$

We experiment with RoBERTa (Liu et al., 2019) and SBERT (Reimers and Gurevych, 2019) as embedding models. We also try BM25 (Robertson and Zaragoza, 2009), a retriever based on surface text similarity. We find SBERT leads to the best result; therefore, all IC-DST results reported here use SBERT as the retriever.

Retriever finetuning We also finetune SBERT on the few-shot examples to get a retriever that is better matched to the objective of predicting state changes. We first define the similarity between state changes. Suppose there are two sets of state changes, $c_a = \{(s_1^a, v_1^a), \cdots, (s_m^a, v_m^a)\}$, $c_b = \{(s_1^b, v_1^b), \cdots, (s_n^b, v_n^b)\}$. Let $F(set_1, set_2)$ be the average of two F_1 scores calculated by using set_1 vs. set_2 as the target. Here we use the standard definition $F_1 = \frac{2PR}{P+R}$, in which P is precision, and R is recall. We define the slot similarity as

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

and the slot-value pair similarity as

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

Then the similarity between c_a and c_b is

$$s(c_a, c_b) = \frac{1}{2}(F_{slot} + F_{slot\text{-}value}).$$

The positive and negative examples for training sample $x_i = (e_i, c_i)$ are identified by computing $s(c_i, c_j)$ for each sample $x_j = (e_j, c_j)$, sorting, and taking the k highest and lowest scoring samples, respectively. We finetune the embedding model M with a contrastive loss so that the similarity between a positive example pair is high and the similarity between a negative example pair is low.

2630

3 Experiments

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., 2020) and MultiWOZ 2.4 (Ye et al., 2021a). 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.[3]

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.

SGP-DST (Lee et al., 2021) In SGP-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 us, this model still requires finetuning on DST labels.

3.3 Experimental settings

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

Zero-shot setting There are no labeled examples to retrieve, but a single formatting example turn is included, following previous zero-shot learning work (Wang et al., 2022).

3.4 Experimental Details

Language models GPT3 (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., 2021), is pretrained using open-source code from Github.[4] This enables interesting applications such as code completion. In our initial studies, as in (Shin and Van Durme, 2022), Codex substantially outperforms GPT3; therefore, we use Codex for the following experiments. In this paper, we use **Codex-Davinci**.[5] 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., 2020a) and open-source code.

Evaluation The standard joint goal accuracy (JGA) is used as the evaluation metric. It treats a prediction as correct only if for every domain all slots exactly match the ground-truth values. To be consistent with prior work (Wu et al., 2019), we report all-domain JGA on few-shot settings and per-domain JGA on zero-shot settings. We also report the F_1 on slot-value pairs for analysis.

IC-DST details The retriever is initialized with SBERT all-mpnet-v2 (220M). We use AdamW optimizer (Loshchilov and Hutter, 2018) and set 2×10^{-5} as learning rate, 1000 as warmup steps. During retriever finetuning, for each training sample, we first compute similarity based on target labels. The positive and negative examples are the top and bottom 5% samples among the 10% nearest neighbors of the sample. This sampling strategy gives us hard negative examples. The maximum

---

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

[4]https://openai.com/blog/openai-codex/

[5]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.

| | MultiWOZ 2.1 | | | | MultiWOZ 2.4 | | | |
Model	1%	5%	10%	100%	1%	5%	10%	100%
Baselines (finetuned)								
TRADE (Wu et al., 2020)	12.58	31.17	36.18	46.00	-	-	-	55.05
SGPDST (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 1: Multi-domain 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.

length for retriever input is 512. With the dialogue state context representation, no turn contexts in MultiWOZ exceed this length limit. For few-shot experiments, we use 10 context exemplars in Codex experiments and 5 for GPT-Neo due to different length limits. We set temperature to 0 to enable greedy argmax sampling during generation.

4 Results

Few-shot DST on MultiWOZ Table 1 shows the result on few-shot settings and full-shot settings of our IC-DST compared with several recent baselines on MultiWOZ 2.1 and 2.4. As discussed in Section 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 Unified-SKG (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.

Zero-shot DST on MultiWOZ Table 2 shows the zero-shot DST results on MultiWOZ 2.1 and 2.4. Our IC-DST outperforms previous results by a large margin. In addition, IC-DST has the ad-

vantage over these approaches that no training is required. The multi-domain JGA of our IC-DST is 35.3% on MultiWOZ 2.4, which can be compared to 48.35% for the system using few-shot learning with 1% training (roughly 80 labeled dialogues). SimpleTOD++ (Hosseini-Asl et al., 2020) and T5DST (Lin et al., 2021b) are trained on four domains on MultiWOZ and tested on the unseen domain. In addition, T5DST uses human-written slot descriptions to boost zero-shot performance. TransferQA (Lin et al., 2021a) does not need any training on DST data. However, each slot has to be reformatted into a question, and the model is trained on a large amount of QA data. Our results show the flexibility of IC-DST. For each new domain or slot added, by updating the SQL tables and adding a demonstration example, the model attains good performance on the new ontology without any training.

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.

Representation of dialogue context Table 3 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

	attraction	hotel	restaurant	taxi	train
MultiWOZ 2.1					
SimpleTOD++ (Hosseini-Asl et al., 2020)	28.01	17.69	15.57	59.22	27.75
T5DST + description (Lin et al., 2021b)	33.09	21.21	21.65	64.62	35.42
TransferQA (Lin et al., 2021a)	31.25	22.72	26.28	61.87	36.72
IC-DST Codex	**59.97**	**46.69**	**57.28**	**71.35**	**49.37**
MultiWOZ 2.4					
IC-DST Codex	**62.09**	**53.18**	**54.87**	**71.87**	**51.42**

Table 2: Zero-shot per-domain JGA on MultiWOZ 2.1 and 2.4. For reference, the multi-domain JGA for IC-DST on MultiWOZ 2.4 is 35.3%.

Dialogue context representation	Example label (retrieval objective)	MW 2.4 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 3: Comparison of DST 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.
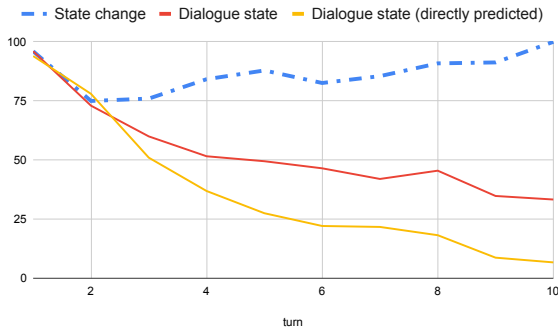

Figure 5: JGA of each turn. The blue line is the JGA of state changes predicted by the system on Table 3 row 5 (our IC-DST setting). The red line is the JGA of dialogue states produced by accumulating predicted state changes. The yellow line is the JGA of dialogue states predicted by the system on Table 3 row 2.

both of the retrieval objectives, representing the dialogue context by the previous dialogue state and the current turn gives the best performance. There are multiple possible explanations. First, the previous state is more relevant than the full context for predicting the next-turn dialogue state, especially after a topic shift. Second, full dialogue contexts are too long. To fit in the length limit of Codex, we sometimes need to truncate examples when using full context, which lowers the system performance, as evidenced by the fact that a single turn outperforms the full context but not the state-based representation.

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 3 shows that using state changes gives substantial improvement. A likely reason is that state changes contain fewer slots compared with dialogue states, making for an easier prediction task for the LM. Also, there are many more example turns with the same state changes, compared with examples with the same dialogue states, making it easier for the retriever to fetch good examples in few-shot scenario.

We further compare these two kinds of example labels by investigating the JGA on each turn, illustrated in Figure 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 3 row 5) as compared to predicting the full state (yellow line, Table 3 row 2).

		Copy		IC-DST Codex	
		JGA	F_1	JGA	F_1
Unsuper.	Random	0.7	4.9	29.8	81.1
	BM25	4.9	31.8	49.0	91.0
	SBERT	5.9	28.7	49.2	91.7
Super.	**Ours**	**16.3**	**61.4**	**58.7**	**93.3**
Oracle	F_1-oracle	24.6	76.8	82.9	98.1

Table 4: Comparison of different retrievers using 5% of the training data and Codex. The "copy" baseline copies the label of retrieved nearest example as prediction. The oracle retriever assumes prior knowledge of the gold test instance dialogue state.

Performance dependence on the retriever Table 4 compares Codex's performance when using different retrievers. F_1 is computed over the pre-

dicted and gold full dialogue states. The "copy" baseline makes a prediction by copying the label of the nearest example retrieved by the corresponding retriever in the selection pool. This baseline gives a direct measure of the quality of the retrieved examples. First, the result shows that Codex's performance is highly dependent on the quality of the retrievers and their retrieved examples. We can also see that F_1 of "copy" baseline correlates with the end task performance (JGA), which supports the use of state-change F_1 as a learning objective for our similarity measure. Our finetuned SBERT retriever obtains substantial improvements over the unsupervised one. The oracle retriever fetches the examples that have the highest state change F_1, assuming prior knowledge of test instance gold dialogue states. It shows the upper bound of the retriever performance in this few-shot setting. Note that the state-of-the-art full-shot performance on MultiWOZ 2.4 is 73.62% (Ye et al., 2021b). The oracle experiment shows that the few-shot ICL may match or outperform the state-of-the-art full-shot model with further retriever improvements.

Table 4 also shows that the LM is *not* simply copying the example labels. For both evaluation metrics and all the retrievers, the "copy" baseline performance is much lower than IC-DST, indicating that the inference language model is generalizing over the examples, rather than copying the closest example labels.

	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 5: 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 5 shows the performance of our ICL framework given different input-output formats. We follow Simple-TOD (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.3. 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 5, 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 6. 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. 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. The third type of error is *failure to generalize from examples*, which happens when the model fails to learn from examples.

6   Related Work

Dialogue state tracking   Many DST systems have been proposed (Wu et al., 2019; Zhang et al., 2020; Peng et al., 2021; Lin et al., 2020; Lee et al., 2021; Zhao et al., 2022; Kim et al., 2020; Heck et al., 2020). Hosseini-Asl et al. (2020), Ham et al. (2020) and Peng et al. (2021) use naive formats to represent the dialogue states. Cheng et al. (2020), Moradshahi et al. (2021), and Platanios et al. (2021) use hierarchical representations for dialogue states as in semantic parsing tasks. To reduce the need for labeled data in DST, many approaches are proposed for few-shot DST (Wu et al., 2020; Li et al., 2021; Gao et al., 2020b; Lin et al., 2021b; Campagna et al., 2020; Su et al., 2022). The state-of-the-art few-shot DST model is (Shin et al., 2022), in which the authors reformulate DST as a summarization task. We propose to represent the dialogue states as SQL and reformulate DST into a text-to-SQL task.

Most zero-shot methods have involved domain-transfer approaches, including multi-task training with related task-oriented domains (Hosseini-Asl et al., 2020; Lin et al., 2021b) and question-answering datasets (Lin et al., 2021a). Unfortunately, performance of these systems is quite low.

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 6: 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.

In-context learning ICL of large PLMs like GPT3 (Brown et al., 2020) has shown increasingly impressive few-shot performance on many NLP tasks. ICL has obtained success in semantic parsing (Pasupat et al., 2021; Rubin et al., 2022; Shin and Van Durme, 2022), intent classification (Yu et al., 2021) and other primarily sentence-level tasks (Min et al., 2022). To imitate ICL, Pasupat et al. (2021) finetuned a T5 model with inputs augmented by retrieved in-context examples. Shin and Van Durme (2022) and Rajkumar et al. (2022) find that GPT3 or Codex can be generalized to produce different target programs with a few in-context exemplars. Xie et al. (2022) and Madotto et al. (2021) were the first to apply ICL for DST, but their systems underperform other methods. Future work may consider improving dialogue retrieving methods and task prompting formulation.

Retrieval Most current work on ICL focuses on sentences or documents, while our task involves retrieving dialogues. There are two general kinds of semantic retrievers. The first is similarity-based retrieval. Poesia et al. (2022) and Das et al. (2021) define a similarity metric between semantic parsing results and use this similarity as the training objective for the retriever. Another approach is LM-score based retrieval. Rubin et al. (2022) and Shin et al. (2021) measure the quality of an example by the probability of a large language model decoding the correct answer. The k highest and lowest quality samples are used as positive and negative samples for the retriever training. The most relevant retrieval studies on dialogue focus on tasks like knowledge identification (Wu et al., 2021) and response selection (Yuan et al., 2019; Han et al., 2021). Their tasks and settings are different from ours.

7 Conclusion

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 both few-shot and zero-shot settings. 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.

8 Acknowledgments

9 Limitations

The performance of our framework is highly dependent on the inference language model, which may limit the framework's usage. For example, our framework may not work as well on speech recognition transcripts or other languages because of the lack of such data during language model pre-training. Future work may explore the robustness and generalization ability of in-context learning, for which our IC-DST can serve as a test bed. Also, there is a tradeoff between avoiding the cost of fine tuning with a large language model vs. the cost of inference. Fortunately, thanks to the recent efforts in open-source large models like OPT (Zhang et al., 2022) and BLOOM (BigScience), and model compression techniques like LLM.int8() (Dettmers et al., 2022), the cost of running large language models has been drastically reduced, and we are optimistic that this trend will continue in the future. Further, it is possible to leverage zero/few-shot methods as a teacher model to generate "pseudo-labels" for training a system that has a lower inference cost, as in Ye et al. (2022). Future work may investigate more on low-cost approaches to applying in-context learning and large language models.

References

BigScience. Bigscience language open-science open-access multilingual (bloom) language model. *International, May 2021-May 2022*.

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

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

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 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, Brussels, Belgium. Association for Computational Linguistics.

Paweł Budzianowski and Ivan Vulić. 2019. Hello, it's gpt-2 - how can i help you? towards the use of pre-trained language models for task-oriented dialogue systems. *Proceedings of the 3rd Workshop on Neural Generation and Translation*.

Giovanni Campagna, Agata Foryciarz, Mehrad Moradshahi, and Monica Lam. 2020. Zero-shot transfer learning with synthesized data for multi-domain dialogue state tracking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 122–132, Online. Association for Computational Linguistics.

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. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Jianpeng Cheng, Devang Agrawal, Héctor Martínez Alonso, Shruti Bhargava, Joris Driesen, Federico Flego, Dain Kaplan, Dimitri Kartsaklis, Lin Li, Dhivya Piraviperumal, et al. 2020. Conversational semantic parsing for dialog state tracking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8107–8117.

Rajarshi Das, Manzil Zaheer, Dung Thai, Ameya Godbole, Ethan Perez, Jay Yoon Lee, Lizhen Tan, Lazaros Polymenakos, and Andrew McCallum. 2021. Case-based reasoning for natural language queries over knowledge bases. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9594–9611, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*.

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

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

Shuyang Gao, Sanchit Agarwal, Di Jin, Tagyoung Chung, and Dilek Hakkani-Tur. 2020b. From machine reading comprehension to dialogue state tracking: Bridging the gap. *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI.*

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.

Janghoon Han, Taesuk Hong, Byoungjae Kim, Youngjoong Ko, and Jungyun Seo. 2021. Fine-grained post-training for improving retrieval-based dialogue systems. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1549–1558.

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

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

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.

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.

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, Melbourne, Australia. 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. *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume.*

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.

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.*

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations.*

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.*

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

Mehrad Moradshahi, Victoria Tsai, Giovanni Campagna, and Monica S Lam. 2021. Contextual semantic parsing for multilingual task-oriented dialogues. *arXiv preprint arXiv:2111.02574.*

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, 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.

Emmanouil Antonios Platanios, Adam Pauls, Subhro Roy, Yuchen Zhang, Alexander Kyte, Alan Guo, Sam Thomson, Jayant Krishnamurthy, Jason Wolfe, Jacob Andreas, et al. 2021. Value-agnostic conversational semantic parsing. 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 3666–3681.

Gabriel Poesia, Alex Polozov, Vu Le, Ashish Tiwari, Gustavo Soares, Christopher Meek, and Sumit Gulwani. 2022. Synchromesh: Reliable code generation from pre-trained language models. In *International Conference on Learning Representations*.

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

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Stephen Robertson and Hugo Zaragoza. 2009. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc.

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, Seattle, United States. Association for Computational Linguistics.

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, Dublin, Ireland. Association for Computational Linguistics.

Richard Shin, Christopher Lin, Sam Thomson, Charles Chen, Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, Dan Klein, Jason Eisner, and Benjamin Van Durme. 2021. Constrained language models yield few-shot semantic parsers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7699–7715, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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, Seattle, United States. Association for Computational Linguistics.

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, Dublin, Ireland. Association for Computational Linguistics.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. 2022. Benchmarking generalization via in-context instructions on 1,600+ language tasks. *arXiv preprint arXiv:2204.07705*.

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

Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. 2019. Transferable multi-domain state generator for task-oriented dialogue systems. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics.

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, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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. *arXiv preprint arXiv:2201.05966*.

Fanghua Ye, Jarana Manotumruksa, and Emine Yilmaz. 2021a. Multiwoz 2.4: A multi-domain task-oriented dialogue dataset with essential annotation corrections to improve state tracking evaluation. *arXiv preprint arXiv:2104.00773*.

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

Jiacheng Ye, Jiahui Gao, Qintong Li, Hang Xu, Jiangtao Feng, Zhiyong Wu, Tao Yu, and Lingpeng Kong. 2022. Zerogen: Efficient zero-shot learning via dataset generation. *arXiv preprint arXiv:2202.07922*.

Dian Yu, Luheng He, Yuan Zhang, Xinya Du, Panupong Pasupat, and Qi Li. 2021. Few-shot intent classification and slot filling with retrieved examples. In

Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 734–749.

Chunyuan Yuan, Wei Zhou, Mingming Li, Shangwen Lv, Fuqing Zhu, Jizhong Han, and Songlin Hu. 2019. Multi-hop selector network for multi-turn response selection in retrieval-based chatbots. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 111–120.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Yichi Zhang, Zhijian Ou, and Zhou Yu. 2020. Task-oriented dialog systems that consider multiple appropriate responses under the same context. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9604–9611.

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

A Prompt Examples

A.1 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
```

A.2 Prompt in Zero-Shot Settings

Below is an example of the prompt in zero-shot settings. Some slots are renamed to be easier for model to understand. There is one crafted turn for task demonstration, which is marked as "Example

#1". The "Example #2" is the test instance that needs to be completed.

```
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_number_of_days 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_number_of_days 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,
  depart_time text,
  arrive_by_time text
)
/*
3 example rows:
SELECT * FROM train LIMIT 3;
destination departure day book_people depart_time
arrive_by_time
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 CHECK (type IN (architecture, boat, church,
  cinema, college, concert hall, entertainment, hotspot,
  multiple sports, museum, nightclub, park, special, swimming
   pool, theatre))
)
/*
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_type 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_type 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,
  depart_time text,
```

```
  arrive_by_time text
)
/*
3 example rows:
SELECT * FROM taxi LIMIT 3;
destination departure depart_time arrive_by_time
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]
[system]
Q: [user] i am looking for a guest house to stay in the west.
 i do not need internet .
SQL: SELECT * FROM hotel WHERE type = guest house AND area =
 west AND internet = no;

Example #2
[context]
[system]
Q: [user] i would like a taxi from saint john s college to
pizza hut fen ditton .
SQL: SELECT * FROM
-------------------Prompt Ends here!--------------------
LM completion: taxi WHERE departure = saint john s college
AND destination = pizza hut fen ditton
```

A.3 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, swimming pool, theatre

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
```

B Details on Example Errors

This section provides a more detailed description of errors described in Table 6.

The first type of error is caused by a *noisy training example*. In this example, there is a missing slot 'attraction-type' in the example label.

The second type of error is *unable to retrieve good examples*. 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 *fail to generalize 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'.

C 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.

D Dataset Details

MultiWOZ (Budzianowski et al., 2018) is an English multi-domain task-oriented dialogue dataset. It contains 7 different domains. 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. (2021a). This script mainly fixes typos and standardizes the formatting. All data are downloadable from Ye et al. (2021a).

E Detailed Results

Table 7 shows each run's JGA on MultiWOZ 2.1 and 2.4.

	MultiWOZ 2.1			MultiWOZ 2.4		
	1%	5%	10%	1%	5%	10%
GPT-Neo						
Run 1	18.44	26.24	32.10	19.29	28.79	35.22
Run 2	14.32	27.63	31.62	14.32	30.78	33.98
Run 3	17.35	26.83	31.23	18.46	29.30	33.93
Mean	16.70	26.90	31.65	17.36	29.63	34.38
(Stdev)	(2.13)	(0.70)	(0.44)	(2.66)	(1.03)	(0.73)
CodeGen						
Run 1	22.11	29.33	33.70	23.25	33.03	37.83
Run 2	17.78	30.05	34.08	19.22	33.36	37.43
Run 3	22.27	29.47	33.66	23.13	33.08	37.08
Mean	20.72	29.62	33.81	21.87	33.16	37.45
(Stdev)	(2.55)	(0.38)	(0.23)	(2.29)	(0.18)	(0.38)
Codex						
Run 1	44.25	47.52	48.34	49.92	56.27	56.62
Run 2	41.99	47.18	49.04	47.31	54.33	56.65
Run 3	43.16	46.55	48.64	47.83	55.69	57.37
Mean	43.13	47.08	48.67	48.35	55.43	56.88
(Stdev)	(1.13)	(0.49)	(0.35)	(1.38)	(1.00)	(0.42)

Table 7: Detailed test JGA in few-shot settings of MultiWOZ 2.1 and MultiWOZ 2.4.