Diverse Retrieval-Augmented In-Context Learning for Dialogue State Tracking

Brendan King and **Jeffrey Flanigan**
University of California, Santa Cruz
{bking2,jmflanig}@ucsc.edu

Abstract

There has been significant interest in zero and few-shot learning for dialogue state tracking (DST) due to the high cost of collecting and annotating task-oriented dialogues. Recent work has demonstrated that in-context learning requires very little data and zero parameter updates, and even outperforms trained methods in the few-shot setting (Hu et al., 2022). We propose RefPyDST, which advances the state of the art with three advancements to in-context learning for DST. First, we formulate DST as a Python programming task, explicitly modeling language coreference as variable reference in Python. Second, since in-context learning depends highly on the context examples, we propose a method to retrieve a diverse set of relevant examples to improve performance. Finally, we introduce a novel re-weighting method during decoding that takes into account probabilities of competing surface forms, and produces a more accurate dialogue state prediction. We evaluate our approach using MultiWOZ and achieve state-of-the-art multi-domain joint-goal accuracy in zero and few-shot settings.[1]

1 Introduction

Dialogue state tracking (DST) is an important language understanding task required for supporting task-oriented conversational agents. For each turn in a dialogue, the goal of DST is to extract the intentions and arguments a user communicates into a meaning representation aligned with the capabilities of the system. Often, this can be represented as a set of slot-value pairs, using slots defined in a system schema. For example, if a user asks a hotel booking agent for "a four-star hotel with somewhere to park", the agent could extract the state $\{(\text{hotel-stars}, 4), (\text{hotel-parking}, \text{yes})\}$.

Annotating these turn-level dialogue states is challenging and time-intensive (Budzianowski et al., 2018). Further, as system capabilities evolve over time, the schema and DST requirements change. As such, flexible and data-efficient DST methods are highly valuable.

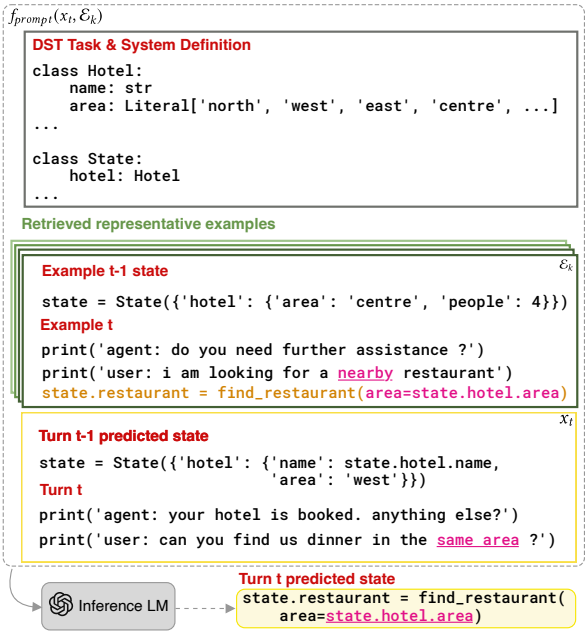

Figure 1: Our retrieval-augmented in-context learning approach to DST. We construct a prompt which re-frames DST as a Python programming task conditioned on a system definition and set of retrieved examples $\mathcal{E}_k$ (green). For each dialogue turn t, the goal is to take the current state (`state`) and turn utterances (`print(...)`) as 'input' and produce a program which *updates* the state with missing values, i.e. (restaurant-area, west). We represent linguistic coreference explicitly as variable reference (pink)

For these reasons, recent work has explored zero and few-shot methods for DST. Few-shot methods often fine-tune a pre-trained language model (LM) on DST or a re-framing of the task (e.g. Su et al., 2021; Shin et al., 2022; Lin et al., 2021a). While these systems are often data efficient, they are inflexible to changing system definitions, requiring re-training as new services are added. To address this, zero-shot methods for domain transfer have been proposed (e.g. Wu et al., 2019; Hosseini-Asl

[1]Our code: https://github.com/jlab-nlp/RefPyDST

Findings of the Association for Computational Linguistics: ACL 2023, pages 5570–5585
July 9-14, 2023 ©2023 Association for Computational Linguistics

et al., 2020; Gupta et al., 2022), but their performance in new domains can significantly depend on conceptual overlap with training domains (Wu et al., 2019).

The in-context learning framework (ICL) (Brown et al., 2020) is particularly appealing in this setting given that it is highly data-efficient and flexible: instead of fine-tuning, ICL methods prompt a fixed LM with templated examples for a task. This approach requires no re-training when adapting to schema changes. In recent work, Hu et al. (2022) find that prompting a language model with examples for DST in a text-to-SQL format can outperform fine-tuned zero and few-shot methods.

In this work, we propose **RefPyDST**, a retrieval-augmented in-context learning approach to DST for use with language models pre-trained on code, such as OpenAI Codex (Chen et al., 2021), by building on recent ICL methods for DST (Hu et al., 2022). Our approach advances the state of the art with three key contributions.

First, we develop a novel in-context prompt that re-frames DST as text-to-python, explicitly modeling slot value coreferents using variables. We provide an overview of this prompt and example of such coreference in Figure 1. We demonstrate that this approach significantly improves system performance in the zero and few-shot settings, and particularly improves accuracy on predictions requiring coreference resolution.

Second, we introduce a novel method for diverse supervised example retrieval, which yields a set of in-context examples $\mathcal{E}_k$ that are both individually relevant and collectively representative of the output space. Our approach significantly improves performance in few-shot settings, overcoming a failure mode in supervised example retrieval in which examples are each similar to an input x but redundant in the outputs they demonstrate.

Third, we propose a novel scoring method PMI^{β} which compensates for surface-form competition among sampled LM completions in constrained generation settings. Inspired by Holtzman et al. (2021), we re-weigh each completion y by an estimate of its a priori likelihood in the task context. We find this improves system performance in both the zero and few-shot settings.

Together, our contributions address key challenges in DST and in retrieval-augmented ICL generally. Our method produces state-of-the-art results on MultiWOZ 2.1 and 2.4 DST benchmarks across a variety of few-shot settings. Similarly, we obtain a new zero-shot state-of-the-art in the multi-domain setting.

2 Task Definition

A task-oriented dialogue consists of turns or paired utterances between a user and an agent which interfaces the user with a programmable system. At each turn t, the purpose of a DST module is to use the dialogue history up to that turn to predict a dialogue state y_t, which represents the user's goal and progress in using the system. Let A_i be an agent utterance, U_i be a user utterance, and $C_t = [(A_1, U_1), (A_2, U_2), ...(A_t, U_t)]$[2] be the dialogue history up to turn t. The task is to map the history C_t to a state representation y_t. In this work, we predict dialogue states y_t which can be represented as slot-value pairs:

$$y_t = \{(s_1, v_1), (s_2, v_2)...(s_n, v_n)\}$$

where each slot s_i and the types of values it permits are defined in a system schema. For example, an agent supporting hotel reservations might have a slot 'hotel-parking' taking boolean values for constraining search to hotels that include parking.

We can equivalently define this task as predicting *state changes*, as proposed in Hu et al. (2022). Let $x_t = [y_{t-1}, (A_t, U_t)]$ be a dialogue *context* consisting of the previous dialogue state prediction and utterances for the current turn. Using this turn context x_t, we predict a state change:

$$\Delta y_t = \{+(s_i, v_i)... - (s_j, v_j)...\}$$

where y_t is computed by applying the difference Δy_t to y_{t-1}. This approach has two advantages for few-shot in-context learning. First, the turn context x_t requires fewer tokens to represent than the complete history C_t, permitting more in-context examples. Second, the number of distinct state changes Δy_t observed in practice is much smaller than the number of distinct states y_t, simplifying the search for relevant examples and the generation problem.

For these reasons, we formulate our DST problem as mapping from the turn context x_t to a state change Δy_t. For readability, we often use 'turn' to refer to this turn context x_t, distinguishing it from the history C_t or turn number t using notation.

[2]For user-initiated dialogues, A_1 may be omitted

3 Methods

Given a dialogue turn t, our method produces a state change Δy_t by (1) retrieving a set of in-context examples $\mathcal{E}_k$, (2) formatting these into a prompt $f_{prompt}(x_t, \mathcal{E}_k)$, (3) generating and scoring possible program solutions (LM completions) with OpenAI Codex (Chen et al., 2021), (4) executing the program to compute a state change Δy_t. Given the state change, we compute the complete dialogue state y_t by applying the difference to y_{t-1}. We describe our prompting function $f_{prompt}(x_t, \mathcal{E}_k)$, in § 3.1. In § 3.2, we describe our method for retrieving a diverse and representative set of examples $\mathcal{E}_k$. Finally, we describe our method for scoring LM completions with a pointwise mutual information estimate in § 3.3.

3.1 Prompting with Text-to-Python

We design a novel prompt that re-frames DST as a text-to-Python task, allowing us to explicitly represent coreference phenomena and leverage the unique capabilities of language models pre-trained with code. Figure 1 provides an overview. Formally, we define a prompting function $f_{prompt}(x_t, \mathcal{E}_k)$, which takes a test dialogue turn x_t and a set of k in-context examples $\mathcal{E}_k = \{(x_1, \Delta y_1), ...(x_k, \Delta y_k)\}$ and produces a string representing the program synthesis task.

Our prompt (Figure 1) starts with a task definition represented as a set of Python classes corresponding to each DST domain. Each informable slot is an attribute in the appropriate class. Type hints are used to label categorical slots with their values and non-categorical slots with the most appropriate type. The dialogue state is also represented as an object which can be manipulated, having an attribute per-domain.

We represent instances of our programming synthesis task with in-context python examples. Each in-context example $([y_{t-1}, A_t, U_t], \Delta y_t)$ is represented as follows: the previous dialogue state y_{t-1} is represented as a dictionary, mapping slot names to values. Non-categorical values such as names are de-lexicalized by replacing their string value with a variable referencing their existing value in the state. Solutions to the programming task are represented as function calls that manipulate the dialogue state. One of the key benefits of our formulation of the DST task as python is explicit representation of coreference phenomena. For example, the solution corresponding to a user input "find me a restaurant in the same area as my hotel" would be `state.restaurant = find_restaurant(area = state.hotel.area)`, explicitly modeling the resolution of the linguistic coreference.

3.2 Retrieving Diverse Relevant Examples

We propose a method for in-context example selection that produces an example set $\mathcal{E}_k$ that is both relevant to a test turn x_t and diverse, representing the relevant portions of the output space. We first learn an embedding space in which similar state changes have high cosine similarity with one another (§3.2.1), following (Hu et al., 2022). Using this, we propose a novel method for decoding $\mathcal{E}_k$ such that examples are similar to x_t but dissimilar to each other (§3.2.2).

3.2.1 Retriever Training

We fine-tune an embedding model to approximate the true similarity between two turn contexts x_i, x_j with the *cosine similarity* between their encoded representations, following prior works (Hu et al., 2022; Rubin et al., 2021). Let D_{train} be a set of dialogue turns serving as training data for an example retriever and selection pool at inference time. As described in §2, each example $e_i \in D_{train}$ is a context state-change pair $e_i = (x_i, \Delta y_i)$. A single example e_i is shown in the green box in Figure 1.

We encode an example or query turn context $x = [y_{t-1}, (A_t, U_t)]$ by concatenating each element of the turn context and passing the result through an embedding model[3] emb. For two example turn contexts x_i, x_j, the cosine similarity between their embeddings $cos(emb(x_i), emb(x_j))$ approximates their relevance to each other. At inference time, we can embed a test turn x_t and retrieve highly similar examples with nearest neighbors search.

We fine-tune our embedding model with a supervised contrastive loss, such that high cosine similarity of representations correlates with high similarity between dialogue state changes, following the procedure in Hu et al. (2022). For our learning objective, we assume a metric that gives the *true* similarity between two dialogue state changes for a pair of turns sim_{F_1}, which we define below. For each dialogue turn in the training set, we use sim_{F_1} to define positive and (hard) negative examples as the top and bottom 5% of the current nearest 200 examples, respectively. We train each retriever for

[3]We use all-mpnet-base-v2 (Song et al., 2020), available in sentence-transformers (Reimers and Gurevych, 2019)

15 epochs using the hyperparameters detailed in Appendix C.

We define the ground-truth similarity sim_{F_1} between two dialogue state changes as follows. Let $\Delta y^a = \{(s_1^a, v_1^a)...(s_m^a, v_m^a)\}$ and $\Delta y^b = \{(s_1^b, v_1^b)...(s_n^b, v_n^b)\}$ be two dialogue state changes. For any slot value v_i exhibiting coreference to another slot s_j, we replace v_i with s_j. For example, the state change corresponding to a turn "I need a taxi to my hotel" would become $\{(\text{taxi-destination}, \text{hotel-name})\}$, regardless of the particular hotel name value. We then compute true state similarity using the average between the F_1 score comparing updated slots and the F_1 score comparing updated slot-value pairs, as proposed in Hu et al. (2022):

$$sim_{F_1}(\Delta y^a, \Delta y^b) = \frac{1}{2}F_1(\{s_1^a, ...\}, \{s_1^b, ...\}) + \frac{1}{2}F_1(\{(s_1^a, v_1^a), ...\}, \{(s_1^b, v_1^b), ...\})$$

3.2.2 Decoding Diverse Examples

We propose a method for using our learned embedding model emb to produce a diverse set of examples $\mathcal{E}_k$ that maximizes similarity to x_t and minimizes similarity between examples in $\mathcal{E}_k$. Particularly for encoders that are fine-tuned to approximate output similarity, this yields a set of examples that is more representative of the output space than simply selecting the nearest k, which may all have the same label. Formally, we define the ideal set of in-context examples $\mathcal{E}_k^*$ for an input x_t to be the k examples satisfying:

$$\mathcal{E}_k^* = \underset{\mathcal{E}_k \subset \mathcal{D}_{train}}{argmax} \sum_{x_i \in \mathcal{E}_k} cos(emb(x_t), emb(x_i)) - \alpha \sum_{x_i, x_j \in \mathcal{E}_k} cos(emb(x_i), emb(x_j))$$

where the hyperparameter α is a dissimilarity factor and $\alpha = 0$ corresponds to typical nearest-k example selection. We greedily approximate $\mathcal{E}_k^*$ by iteratively selecting the example which maximizes the equation at each step. For more efficient decoding of $\mathcal{E}_k$ with large selection pools, we limit the considered examples to the nearest N such that $|D_{train}| >> N >> k$. For example in one run in the 5% MultiWOZ few-shot setting, $|D_{train}| = 2754$, $N = 100$, and $k = 10$.

3.3 Decoding with Point-wise Mutual Information

We introduce a new rescoring function, PMI^β, to mitigate surface form competition when generating from language models, that we use for making predictions in our setting. PMI^β is an extension of PMI_{DC}, which was proposed in Holtzman et al. (2021) for mitigating surface form competition in the classification setting. We first describe surface form competition and PMI_{DC} (§3.3.1), and then describe PMI^β, an adaptation of this method to the constrained generative setting with in-context examples (§3.3.2).

3.3.1 Surface-form Competition

Conditioned on a prompt, a language model assigns a likelihood to all completing strings, from which we can sample. While string likelihoods can be used as a proxy for output class or structure likelihoods, these are not the same. For example, in our DST formulation, many strings can correspond to the same state change Δy_t, or may not correspond to a valid state change at all. As such, Holtzman et al. (2021) argue string likelihoods can be unreliable for scoring the best among a fixed set of choices which may each contain numerous surface forms in V^*. To compensate for this, they propose scoring with Domain Conditional Point-wise Mutual Information ($PMI_{DC} = \frac{P(y|x,domain)}{P(y|domain)}$). This re-weighs choices by a priori likelihood of their string form in the task context $P(y|domain)$.

3.3.2 Scoring with PMI^β

To mitigate surface-form competition, we propose PMI^β: a prompt conditional pointwise mutual information scoring method that adapts PMI_{DC} to our constrained generative setting with in-context examples. Doing so requires overcoming two key challenges. First, our choices to score amongst are not practically enumerable. Second, the task context we condition on is partly defined by our choice of in-context examples $\mathcal{E}_k$. We overcome these by first generating a small set of plausible completions $\mathcal{C}$ and their likelihoods according to a language model. Then, we re-weigh these likelihoods according to an estimate of their a priori likelihood conditioned on only the task context and selected examples $\mathcal{E}_k$:

$$PMI^\beta(x; y|\mathcal{E}_k) = \frac{P(y|f_{prompt}(x_t, \mathcal{E}_k)))}{P(y|f'_{prompt}(\mathcal{E}_k))^\beta} \quad (1)$$

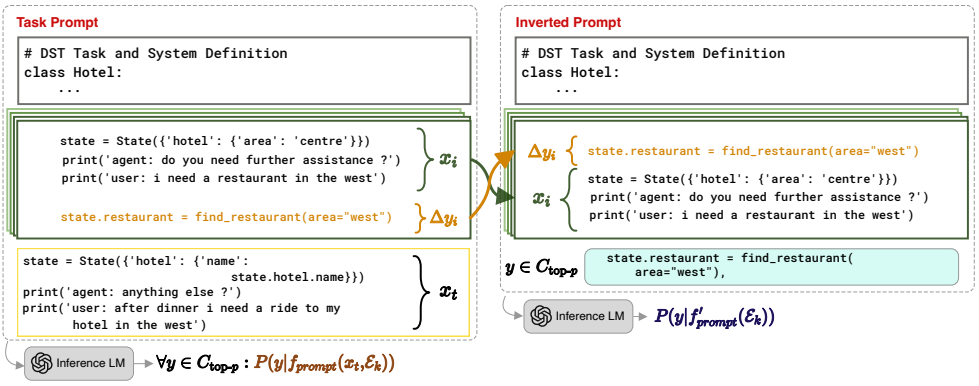

Figure 2: An overview of our method (§3.3) for scoring completions y from Codex with PMI^{β}, which re-weighs using an estimate of the a priori likelihood of y in the context of the task. On the left, is our primary text-to-Python prompt $f_{prompt}(x_t, \mathcal{E}_k)$ (§3.1). We use nucleus sampling to generate a set of reasonable candidates $C_{\text{top-}p}$ and their probabilities. On the right is an inverted prompt with state changes preceding their inputs, allowing us to produce an in-context estimate of the probability of y not conditioned on x

where f'_{prompt} is a prompt designed for estimating $P(y|\mathcal{E}_k)$ without conditioning on x_t, described below, and β is a hyperparameter for adjusting the impact of re-weighing by a priori likelihood.[4]

To generate the candidate completions $\mathcal{C}$, we sample a set of plausible candidates using nucleus sampling (Holtzman et al., 2020).

While one could simply use the language model to compute $P(y)$ directly, such unconditional estimates tend to vary wildly. Following Holtzman et al. (2021), we instead estimate the probability of the completion in context, but further account for the use of in-context examples. To do this, we construct an additional prompt which contains the same problem definition, but reverses the order outputs and inputs. Using this, we can estimate the probability of a completion y in the context of our task and examples without x_t, illustrated in Figure 2. Finally, we select the completion $\hat{y}$ which maximizes Eq. 1, and parse it to a dialogue state change Δy_t:

$$\hat{y} = \underset{y \in \mathcal{C}}{argmax}\, PMI^{\beta}(x; y|\mathcal{E}_k)$$

We choose a minimum a priori likelihood of between 10^{-7} and $10^{-}5$, as estimates for $P(y|f'_{prompt}(\mathcal{E}_k))$ can be very low, particularly when rare slot values implied by x_t are not present in any example. When constructing our candidate set $\mathcal{C}$, we choose the five most likely sampled completions under the original prompt. FFinally, we canonicalize each completion y when computing $P(y|f'_{prompt}(\mathcal{E}_k))$ by first parsing it to a dialogue state change, and then re-writing it as a string in the form as if it were an example in $\mathcal{E}_k$. In effect, this normalizes mis-spellings and enforces the expected order of keyword arguments in the update string, further controlling for high variance in our estimates.

[4]While only $\beta = 1$ corresponds neatly to a point-wise mutual information estimate $pmi(x_t; y)$, we find $0 < \beta < 1$ to be more effective in practice. Prior work in terminology extraction has also proposed scaling PMI estimates, though in a different context (Daille, 1994)

4 Experiments

We describe our zero and few-shot experimental setups, evaluation, and baselines. Hyperparameter and implementation details can be found in Appendix C.

4.1 Experimental Settings

We conduct zero and few-shot DST experiments on the MultiWOZ dataset (Budzianowski et al., 2018), containing over ten thousand multi-domain task-oriented dialogues crowd-sourced in a wizard-of-oz setup. There are five domains in the validation/test sets and a total of thirty informable slots. We evaluate on the newest MultiWOZ 2.4 (Ye et al., 2022a). For comparison with prior work, we also report on MultiWOZ 2.1 (Eric et al., 2020).

We evaluate performance with standard joint-goal accuracy (JGA) for all of our experiments. For a turn x_t, a dialogue state prediction $\hat{y}_t$ is considered correct only if all slot names and values exactly match the ground-truth state y_t.

For the few-shot setting, following (Wu et al., 2020), we sample 1%, 5%, or 10% of the dialogues from the training set to serve as a training

	MultiWOZ 2.1				MultiWOZ 2.4			
Model	1%	5%	10%	100%	1%	5%	10%	100%
TRADE (Wu et al., 2019)	12.6	31.2	36.2	46.0	-	-	-	55.1
DiSTRICT (Venkateswaran et al., 2022)	13.4	41.3	49.7	56.1	-	-	-	-
DS2 (Shin et al., 2022)	33.8	44.2	45.4	52.3	36.8	49.9	51.1	57.9
IC-DST Codex (Hu et al., 2022)	43.1	47.1	48.7	50.7	48.4	55.4	56.9	62.4
RefPyDST (ours)	**47.3**	**49.6**	**50.8**	52.0	**55.2**	**62.3**	**62.5**	65.2

Table 1: Multi-domain JGA evaluated on MultiWOZ 2.1 & 2.4 using samples from 1%, 5%, 10%, and 100% of the training set. Average of three runs is reported. Our method achieves state-of-the-art (**bolded**) for both dataset versions in the 1%, 5%, and 10% few-shot settings. Our method also out-performs all few-shot baselines which report results in the 100% setting on MultiWOZ 2.4. Line distinguishes fine-tuned from in-context learning methods.

set D_{train} for each experiment. We fine-tune our retriever using D_{train} and select in-context examples from it. We conduct three independent runs for each sample size and report the average JGA across runs. We also perform a single run in the full setting, using 100% of the training data.

For the zero-shot setting, there are no labeled examples to select from, but a single formatting example is used for all inference turns, as in (Wang et al., 2022; Hu et al., 2022). We consider two evaluation settings. The first is the typical assessment on all test set dialogues, as in few-shot and complete training regimes, which we will refer to as the standard MultiWOZ benchmark. These results allow comparison to few-shot and full-data results, as well as other methods which use zero supervised dialogues in training. We also report results on the MultiWOZ 'leave-one-out' benchmark for zero-shot transfer methods (Wu et al., 2019), reporting JGA considering only slots in each individual domain, as well as the average of these five single-domain results.

We compare to a number of prior state-of-the-art zero-shot and few-shot DST methods as baselines. These include DST specific architectures (Wu et al., 2019), various fine-tuning methods (Gupta et al., 2022; Shin and Van Durme, 2022; Venkateswaran et al., 2022), and a strong ICL baseline (Hu et al., 2022).

5 Results

Few-shot DST on MultiWOZ We present few-shot and full-shot dialogue state tracking results on MultiWOZ 2.1 & 2.4 in Table 1. We find that our method achieves state-of-the-art in the 1%, 5%, and 10% few-shot settings for both MultiWOZ 2.1 & 2.4, outperforming all fine-tuned methods as well as other in-context learning methods. While all methods considered improve with additional data, our method is remarkably data efficient: RefPyDST achieves 95% of its full-shot performance using only 5% of the training data, on average. In comparison, using 5% of the training data with IC-DST Codex only achieves 89% of its full-shot performance.

Zero-shot DST on MultiWOZ We present zero-shot multi-domain results on MultiWOZ 2.4 in Table 3. We find our method outperforms all zero-shot methods, achieving a 12.4% increase in multi-domain JGA over IC-DST Codex, our strongest performing baseline. Comparisons are limited to methods that use zero training data, as opposed to transfer methods that train on some MultiWOZ domains and evaluate on others.

For comparison with domain transfer methods, we present zero-shot results on the leave-one-out benchmark for MultiWOZ 2.1 & 2.4 in Table 2. Following prior work, we evaluate only dialogues and slots in the held-out domain.[5] Evaluating average performance in this setting, we find our method outperforms all methods except for the current state-of-the-art transfer method, SDT-seq. Their method outperforms ours by 1.5% on each held-out domain on average. However, transfer methods such as SDT-seq require significant out-of-domain DST training data, while ours requires none. Despite this training data disadvantage, our approach outperforms all other zero-shot transfer methods.

6 Analysis & Ablations

In this section, we further analyze the performance characteristics of our method.

[5]Prior work on the leave-one-out setting evaluates using the following method: (1) filter to dialogues which *contain* the held out domain (this can include dialogues in multiple domains) and (2) only check slots in that domain when computing JGA. (Wu et al., 2019)

	attraction	hotel	restaurant	taxi	train	**Avg.**
MultiWOZ 2.1						
TRADE (Wu et al., 2019) †	20.1	14.2	12.6	59.2	22.4	25.7
TransferQA (Lin et al., 2021a) †	31.3	22.7	26.3	61.9	36.7	35.8
DiSTRICT (Venkateswaran et al., 2022) †	33.4	22.4	24.0	66.6	47.7	38.8
D3ST (Zhao et al., 2022) †	56.4	21.8	38.2	78.4	38.7	46.7
SDT-seq (Gupta et al., 2022) †	**74.4**	33.9	**72.0**	**86.4**	62.9	**65.9**
IC-DST (Hu et al., 2022)	60.0	46.7	57.3	71.4	49.4	57.0
RefPyDST (ours)	70.9	**51.2**	65.6	67.1	**69.2**	64.7
MultiWOZ 2.4						
IC-DST Codex (Hu et al., 2022)	62.1	53.2	54.9	**71.9**	51.4	58.7
RefPyDST (ours)	**74.5**	**56.6**	**68.2**	68.5	**76.1**	**68.8**

Table 2: Zero-shot joint-goal accuracy (JGA) for each domain in MultiWOZ 2.1 & 2.4 in the leave-one-out set up. We report results on each held-out domain and the average held-out domain performance (Avg.) Domain transfer methods (marked with †) learn from dialogues in the other four domains and are tested on the held-out domain. Unlike domain transfer methods, IC-DST and our method do not use any DST data. Following prior work, we evaluate only dialogues and slots in the held-out domain. For full evaluation of all dialogues in the zero-shot setup, see Table 3.

MultiWOZ 2.4	
IC-DST Codex (Hu et al., 2022)	35.3
RefPyDST (ours)	**47.9**

Table 3: Zero-shot (zero DST training data) multi-domain JGA evaluated on MultiWOZ 2.4. Our method achieves state-of-the-art for this setting. Comparisons with zero-shot transfer methods, which train on subsets of the MultiWOZ dataset, can be found in Table 2.

Ablations In order to assess how each part of our method contributes to performance, we conduct a leave-one-out ablation, as well as reporting the performance of using only our prompting method. Each ablation is conducted using a 20% sample of the development data in the MultiWOZ 2.4 dataset (200 dialogues), sampled independently of the set used to tune hyperparameters. We present results in Table 4 for the zero and 5% few-shot setting. In the few-shot setting, we find leaving out our diverse retrieval to be most impactful.

Does using Python improve coreference resolution? Since our Python prompting method explicitly models coreference through variable reference, we analyzed how our system performed on state predictions requiring coreference resolution. Using coreference annotations released with the 2.3 version of the MultiWOZ dataset (Han et al., 2021), we evaluate accuracy on slot values which require coreference to resolve. Our results are presented in Table 5. Overall, our full model improves upon the baseline for coreference. Removing Python greatly reduces our model's performance, demonstrating the benefit of modeling coreference as Python variable reference.

Does our retrieval method improve demonstrated label diversity? We investigate to what degree our diverse decoding procedure increases diversity in the distribution of demonstrated labels for a given input. To approximate a label, we define $S(e_i)$ as the distinct combination of *slot names* in the output for an in-context example $e_i = (x_i, \Delta y_i)$, ignoring assigned values.

First, we simply count the average number of distinct combinations of slot names in $\mathcal{E}_k$, shown in upper half of Table 6. For each x_t, we retrieve a set of in-context examples $\mathcal{E}_k$. We count the number of distinct slot combinations across each $e_i \in \mathcal{E}_k$, and report the development set average. A value of 1 indicates the retriever is fully redundant: all k examples demonstrate the same combination of slots, while a value of k indicates every example in $\mathcal{E}_k$ is unique.

Second, we consider the entropy of slot combinations present in $\mathcal{E}_k$, shown in the lower half of Table 6. For each x_t, we again compute $S(e_i)$ for each retrieved example in $\mathcal{E}_k$. We then compute the specific conditional entropy $H(S|X = x_t)$, estimating the probability of each slot combination $p(S|x_t)$ using its frequency in $\mathcal{E}_k$. We report the development set average or conditional entropy $H(S|X)$. $H(S|X = x_t) = 0$ indicates a fully redundant retriever that retrieves the same set of slots

Few-Shot (5%)	
IC-DST (baseline)	52.4
RefPyDST (prompt only)	53.7
RefPyDST – Python	54.8
RefPyDST – diverse	54.6
RefPyDST – PMI^{β}	56.1
RefPyDST (full)	**57.9**

Zero-Shot	
IC-DST (baseline)	43.0
RefPyDST – Python	40.7
RefPyDST – PMI^{β}	46.0
RefPyDST (full)	**46.7**

Table 4: MultiWOZ joint-goal accuracy in the few-shot (5%) and zero-shot settings, leaving out individual components of our method. We evaluate on a 20% sample of the development set (200 dialogues). For few-shot, we average over three runs, each with independently sampled D_{train}. For ablating the removal of our Python prompt, we use the Text-to-SQL format from (Hu et al., 2022) as a baseline. The alternatives to our diverse retrieval approach and PMI^{β} scoring are top-k retrieval and greedy decoding, respectively

Model	0%	5%
IC-DST (baseline)	67.7	78.9*
RefPyDST (prompt only)	77.1*	77.9*
RefPyDST – Python	62.9	73.0
RefPyDST (full)	76.8*	81.8

Table 5: Accuracy on slot value predictions which require coreference resolution for the zero-shot (0%) and few-shot (5%). For a given setting (column), * indicates the difference is not statistically significant. All other differences in a column are significant to $p < 0.02$

for all examples, and a uniform distribution of slot combinations yields $H(S|X = x_t) = log_2(k)$.[6]

We find our retrieval methods increase the diversity of in-context examples across all settings. For a given training set size, we see that diverse decoding increases the number of distinct 'labels', measured by $S(e_i)$, as well as the entropy $H(S|X)$. Still, selected examples are not random, as we can see when comparing $H(S|X)$ to a random retriever which uniformly samples from D_{train}.[7] Finally, we see that as the size of the training set

[6]While this is true of a uniform distribution over demonstrated slot combinations, we find uniformly sampling from D_{train} yields an entropy of ~ 2.6, as the distribution of labels in the training data is not uniform.

[7]In Appendix D, we also compare few-shot task performance for our retrieval method against random retrieval

Number of Distinct S in $\mathcal{E}_k$

	1%	5%	10%	100%
random	7.1	7.2	7.2	7.3
top-k	3.4	2.2	1.8	1.5
diverse ($\alpha = .2$)	<u>5.3</u>	<u>4.1</u>	3.3	2.2
diverse ($\alpha = .3$)	5.7	4.5	<u>3.5</u>	2.3
diverse ($\alpha = .5$)	7.5	5.7	4.8	<u>2.8</u>

Entropy $H(S|X)$

	1%	5%	10%	100%
random	2.6	2.6	2.6	2.6
top-k	1.2	0.63	0.47	0.30
diverse ($\alpha = .2$)	<u>1.8</u>	<u>1.5</u>	1.1	0.64
diverse ($\alpha = .3$)	1.9	1.6	<u>1.2</u>	0.68
diverse ($\alpha = .5$)	2.7	2.0	1.7	<u>0.93</u>

Table 6: We analyze the *outputs* demonstrated in $\mathcal{E}_k$ for different in-context example retrieval methods. Above, we show the average number of distinct slot combinations demonstrated in $\mathcal{E}_k$. Below, we show the conditional entropy $H(S|X)$ of the distribution of slot combinations in $\mathcal{E}_k$. We underline the values corresponding to methods used in our final models

increases, the diversity in exemplified labels for a given choice of α *decreases*. Increasing training data leads to a higher density of each slot combination, requiring more aggressive discounting to achieve the same diversity in $\mathcal{E}_k$. As such, we increase α with training set size, using $\alpha = 0.2$ for 1% and 5% settings and $\alpha = 0.3$ & $\alpha = 0.5$ for 10% and 100% settings, respectively.

7 Related Work

Dialogue State Tracking There has been a recent increase in work on the zero and few-shot DST systems. Many approaches fine-tune a pre-trained language model by re-framing DST as some form of text-to-text or auto-regressive language modeling task (Wu et al., 2020; Peng et al., 2021; Hosseini-Asl et al., 2020; Su et al., 2021; Shin et al., 2022; Lin et al., 2021b; Gupta et al., 2022; Li et al., 2021; Xie et al., 2022). Many of these methods often exhibit zero-shot transfer capabilities (Wu et al., 2019; Gupta et al., 2022; Li et al., 2021; Hosseini-Asl et al., 2020). However, these approaches still require re-training when a domain is added or changed, and zero-shot transfer performance is dependent on the relatedness of the new domain to existing ones.

Some recent works instead model DST as an in-context learning problem (Hu et al., 2022; Xie et al.,

2022; Madotto et al., 2021), bypassing the need for re-training when system definitions change. In particular, we build on the work of Hu et al. (2022), which models DST by predicting dialogue state *changes* at each turn, relying on only a state summary and agent/user turn utterances for inference. Their work models DST as a text-to-SQL problem, whereas we model it as a Python programming problem with novel methods for selecting in-context examples and scoring language model completions.

In-Context Learning Some recent works explore the properties of effective in-context examples. In classification settings, Gao et al. (2021) find random examples can significantly limit performance, and propose using a pre-trained embedding model to find examples semantically close to x, retrieving one per class. Other works investigate the role of examples in ICL performance in detail, finding that ICL methods perform best when example inputs and test inputs are as close in distribution as possible, and when the distribution of exemplified labels closely matches the target distribution (Min et al., 2022; Liu et al., 2022).

Paralleling this, a number of works across NLP tasks propose methods for retrieving relevant in-context examples. Pasupat et al. (2021) use an unsupervised embedding model to embed a test input x and all available examples, retrieving the k with highest embedding cosine similarity. Other works use a similar dense retriever but in an embedding space learned with supervision. Rubin et al. (2021) fine-tune an example retriever with contrastive learning in which positive examples maximize $p_{LM}(y|x, e_i)$. Hu et al. (2022) propose a contrastive learning objective specific to DST, fine-tuning an embedding model to embed turns with similar state changes in proximity to each other. Rather than use a separate retrieval module, Shin and Van Durme (2022) use the LM itself to select examples which are most likely when conditioned on x. Given a test input x, each of these works scores the relevance of an individual example e_i to a test input x and then selects the k most relevant ones to include in a prompt. In most cases, this yields a set of examples $\mathcal{E}_k$ which are meaningfully similar to x. However, considering examples individually does not necessarily lead to adequate exemplification of the output space. In supervised settings that learn a relevance metric which approximates output similarity, this can lead to degenerate examples sets $\mathcal{E}_k$ which all exemplify the same output. In contrast to this, we propose a novel method for using this score to construct $\mathcal{E}_k$ with examples that are relevant to x while being distinct from each other.

In concurrent work to our own, Ye et al. (2022b) propose a similar algorithm for decoding diverse examples of explanations from a retriever for use in reasoning problems, a strategy called maximum-marginal-relevance (MMR) selection. Their work uses unsupervised measures of similarity between explanations, where ours uses a supervised retriever which approximates similarity of outputs. Thus, diversity in our example sets correlates to diversity in exemplified outputs. In another concurrent work to our own (Levy et al., 2022) propose to use method for diverse examples selection in a semantic parsing task, using the outputs of selected examples to incrementally cover more structures in $\mathcal{E}_k$.

For tasks which can be re-framed as program synthesis, a number of works have also developed ICL methods for use with LMs pre-trained on code such as Codex and Codegen (Chen et al., 2021; Nijkamp et al., 2022). Shin and Van Durme (2022) use ICL with Codex to generate Lisp-like programs in a dialogue semantic parsing task. Rajkumar et al. (2022) evaluate such models capabilities in Text-to-SQL problems, and Hu et al. (2022) use a Text-to-SQL framing to use Codex for DST. Instead of SQL queries, we generate Python programs, allowing for intuitive modeling of phenomena like coreference.

Finally, recent works have considered adjusting how completion strings are scored with an LM. Brown et al. (2020) normalize log-likelihoods by length before scoring completions. Zhao et al. (2021) re-weigh LM probabilities by learning an affine transformation that yields uniform scores given 'content-free inputs'. Holtzman et al. (2021) propose PMI_{DC}, a method for re-scoring completions using pointwise mutual information (pmi), which we adapt to our constrained generative setting.

8 Conclusion

We propose RefPyDST, an in-context learning method for DST. Our contributions address key challenges in DST and in retrieval-augmented ICL, producing state-of-the-art results on MultiWOZ DST benchmarks for few-shot and zero-shot setups. Future work could apply methods developed here to other in-context learning problems.

9 Limitations

While in-context learning methods for DST are promising in their data efficiency and flexibility to new domains, they typically require very large models to perform effectively. At 175 billion parameters, OpenAI Codex (Chen et al., 2021) is much larger than some of the fine-tuned approaches to DST, though with better performance and ability to adapt to new domains without re-training. Despite our advances, there are still significant errors when applying ICL for DST. As such, ICL may not necessarily be relied on in safety-critical settings.

Acknowledgements

We thank Geetanjali Rakshit, Nilay Patel, Changmao Li, Chris Toukmaji, Rongwen Zhao, and other JLab members for insightful feedback on preliminary drafts of this work, and thank the anonymous reviewers and area chairs for their detailed and helpful feedback. The authors were supported in part by the NSF National AI Institute for Student-AI Teaming (iSAT) under grant DRL 2019805. The opinions expressed are those of the authors and do not represent views of the NSF. We are thankful for the computing resources provided by the Pacific Research Platform's Nautilus cluster, supported by the National Science Foundation under Award Numbers CNS-1730158, ACI-1540112, ACI1541349, OAC-1826967, the University of California Office of the President, and the University of California San Diego's California Institute for Telecommunications and Information Technology/Qualcomm Institute.

References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. *arXiv:2005.14165 [cs]*. ArXiv: 2005.14165.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Ultes Stefan, Ramadan Osman, 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 (EMNLP)*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating Large Language Models Trained on Code. ArXiv:2107.03374 [cs].

Béatrice Daille. 1994. Approche mixte pour l'extraction de terminologie : statistique lexicale et filtres linguistiques.

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.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making Pre-trained Language Models Better Few-shot Learners. *arXiv:2012.15723 [cs]*. ArXiv: 2012.15723.

Raghav Gupta, Harrison Lee, Jeffrey Zhao, Abhinav Rastogi, Yuan Cao, and Yonghui Wu. 2022. Show, Don't Tell: Demonstrations Outperform Descriptions for Schema-Guided Task-Oriented Dialogue. *arXiv:2204.04327 [cs]*. ArXiv: 2204.04327.

R. Hadsell, S. Chopra, and Y. LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742.

Ting Han, Ximing Liu, Ryuichi Takanobu, Yixin Lian, Chongxuan Huang, Dazhen Wan, Wei Peng, and Minlie Huang. 2021. MultiWOZ 2.3: A multi-domain task-oriented dialogue dataset enhanced with annotation corrections and co-reference annotation. *arXiv:2010.05594 [cs]*. ArXiv: 2010.05594 version: 3.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The Curious Case of Neural Text Degeneration. ArXiv:1904.09751 [cs].

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

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A Simple Language Model for Task-Oriented Dialogue. In *Advances in Neural Information Processing Systems*, volume 33, pages 20179–20191. Curran Associates, Inc.

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. Number: arXiv:2203.08568 arXiv:2203.08568 [cs].

Itay Levy, Ben Bogin, and Jonathan Berant. 2022. Diverse Demonstrations Improve In-context Compositional Generalization. ArXiv:2212.06800 [cs].

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. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1063–1074, Online. Association for Computational Linguistics.

Zhaojiang Lin, Bing Liu, Andrea Madotto, Seungwhan Moon, Zhenpeng Zhou, Paul Crook, Zhiguang Wang, Zhou Yu, Eunjoon Cho, Rajen Subba, and Pascale Fung. 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, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zhaojiang Lin, Bing Liu, Seungwhan Moon, Paul Crook, Zhenpeng Zhou, Zhiguang Wang, Zhou Yu, Andrea Madotto, Eunjoon Cho, and Rajen Subba. 2021b. Leveraging Slot Descriptions for Zero-Shot Cross-Domain Dialogue StateTracking. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5640–5648, Online. Association for Computational Linguistics.

Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill 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, Dublin, Ireland and Online. Association for Computational Linguistics.

Andrea Madotto, Zhaojiang Lin, Genta Indra Winata, and Pascale Fung. 2021. Few-Shot Bot: Prompt-Based Learning for Dialogue Systems. *arXiv:2110.08118 [cs]*. ArXiv: 2110.08118.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the Role of Demonstrations: What Makes In-Context Learning Work? *arXiv:2202.12837 [cs]*. ArXiv: 2202.12837 version: 1.

Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. 2022. CodeGen: An Open Large Language Model for Code with Multi-Turn Program Synthesis. ArXiv:2203.13474 [cs].

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

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. *arXiv:2005.05298 [cs]*. ArXiv: 2005.05298.

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. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2021. Learning To Retrieve Prompts for In-Context Learning. *arXiv:2112.08633 [cs]*. ArXiv: 2112.08633.

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 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.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. MPNet: Masked and Permuted Pre-training for Language Understanding. ArXiv:2004.09297 [cs].

Yixuan Su, Lei Shu, Elman Mansimov, Arshit Gupta, Deng Cai, Yi-An Lai, and Yi Zhang. 2021. Multi-Task Pre-Training for Plug-and-Play Task-Oriented Dialogue System. *arXiv:2109.14739 [cs]*. ArXiv: 2109.14739.

Praveen Venkateswaran, Evelyn Duesterwald, and Vatche Isahagian. 2022. DiSTRICT: Dialogue State Tracking with Retriever Driven In-Context Tuning. ArXiv:2212.02851 [cs].

Gengyu Wang, Cheng Qian, Lin Pan, Haode Qi, Ladislav Kunc, and Saloni Potdar. 2022. Benchmarking language-agnostic intent classification for virtual assistant platforms. In *Proceedings of the Workshop on Multilingual Information Access (MIA)*, pages 69–76, Seattle, USA. Association for Computational Linguistics.

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. ArXiv:1905.08743 [cs].

Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, Michihiro Yasunaga, Chien-Sheng Wu, Ming Zhong, Pengcheng Yin, Sida I. Wang, Victor Zhong, Bailin Wang, Chengzu Li, Connor Boyle, Ansong Ni, Ziyu Yao, Dragomir Radev, Caiming Xiong, Lingpeng Kong, Rui Zhang, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 2022. UnifiedSKG: Unifying and Multi-Tasking Structured Knowledge Grounding with Text-to-Text Language Models. Number: arXiv:2201.05966 arXiv:2201.05966 [cs].

Fanghua Ye, Jarana Manotumruksa, and Emine Yilmaz. 2022a. 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, Edinburgh, UK. Association for Computational Linguistics.

Xi Ye, Srinivasan Iyer, Asli Celikyilmaz, Ves Stoyanov, Greg Durrett, and Ramakanth Pasunuru. 2022b. Complementary Explanations for Effective In-Context Learning. ArXiv:2211.13892 [cs].

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. Number: arXiv:2201.08904 arXiv:2201.08904 [cs].

Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate Before Use: Improving Few-Shot Performance of Language Models. *arXiv:2102.09690 [cs]*. ArXiv: 2102.09690.

A Dialogue State Normalization

Real world task oriented dialogue systems can interface users with thousands or more entities, such as restaurants or hotels in MultiWOZ. Since reasoning directly over all such entities is intractable, dialogue understanding modules often first predict a surface form (e.g. a restaurant name mentioned by a user) which another module links to a canonical form (e.g. that restaurants name in a database). While dialogue state trackers evaluated on MultiWOZ do not need to interact with a database, handling of typos and unexpected surface forms is important for a realistic assessment of system performance, since predictions for a slot are evaluated on exact string match. As such, most research systems including the baselines in this paper use rule-based functions to fix typos and unexpected surface forms. We propose a robust rule-based method for effective linking of surface forms to canonical forms described below.

Mapping to canonical forms We begin by first reading in canonical forms for every informable slot in the MultiWOZ system. For categorical slots, these are defined in a schema file, as released with MultiWOZ 2.1 (Eric et al., 2020). For non-categorical slots, we read in values from the database defined with the original MultiWOZ data collection (Budzianowski et al., 2018). Neither source of information contains dialogue data, only information defining the task. The taxi and train service have informable slots for departure and destination locations. In addition to the locations listed for these slots in a database (i.e. scheduled train journeys), we accept the name of any entity which has an address as a canonical form for these slots. For time slots we consider any time represented in "hh:mm" form as canonical. Overall, this gives us a mapping from a slot name s_i to a set of canonical forms $\mathcal{C}_)$ for all slot names.

Given a slot name s_i and a slot value surface form v_j, we select the correct canonical form c_j as follows: (1) we first generate a set of aliases for v_j. These are acceptable re-phrasings of v_j, such as

adding the leading article "the", a domain specifying suffix such as "hotel" or "museum", or switching numbers to/from digit form (e.g. "one" ↔ "1"). We then consider a surface form v_j as mapped to a canonical form c_j if any of the aliases $a_j \in A_j$ is a fuzzy match for the canonical form c_j, using the `fuzz.ratio` scorer in the `fuzzywuzzy` [8] package. We require a score of 90 or higher, and verify in the development data that no surface form maps to more than one canonical form.

Choosing the most likely surface form While in a real world dialogue system we would only need to link to canonical forms, **gold dialogue state states in MultiWOZ are themselves annotated with surface forms**, not always matching the name of the entity in the database and occasionally disagreeing on an entity name. So as to not alter the evaluation process and make sure we can fairly compare to prior work, we use the training data available in each experimental setting to choose the most likely surface form for a given canonical form c_j. To do this, we simply count the occurrences of each surface form in the gold labels of the training set for that experiment, and select the most frequently occurring one for c_j. However for low data regimes, we often do not observe all canonical forms. Following numerous prior works, we make use of the ontology file released with the dataset (Eric et al., 2020; Ye et al., 2022a), which lists all observed surface forms for a slot name, and treat each of these as if we had seen them 10 times. This serves as a smoothing factor for selecting the most likely surface form. For the zero-shot experiments, we use only the counts derived from the ontology file, as we have no training data to observe.

Overall, we find this approach to normalization to be robust when compared to other works, which rely on hard-coded fixes for commonly observed typos. Further, our normalization can be initialized with any similarly formatted system definition and data set, allowing for use in other domains.

To verify that our approach to normalization is not the key factor distinguishing our performance from previous methods, we apply it to a faithful re-implementation of our IC-DST Codex baseline (Hu et al., 2022) in our ablation in Table 4.

[8] `https://pypi.org/project/fuzzywuzzy/`

B Prompt Examples

Please see our GitHub repository for prompt examples: https://github.com/jlab-nlp/RefPyDST.

C Implementation Details

C.1 Hyperparameters

All hyperparameter tuning is performed using a 10% split of the development set (100 dialogues) and manual tuning. We find that a smaller choice for p (0.7) in nucleus sampling helps performance in the zero-shot setting. Similarly, we find that in order to select a diverse set of examples, we need to scale α. We use $\alpha = 0.2$ for the 1% & 5% settings, $\alpha = 0.3$ for 10%, and $\alpha = 0.5$ for the full setting. For the full setting, we also increase the the number of considered examples from the nearest 100 to nearest 200. Across all settings, we compute PMI^{β} with $\beta = 0.4$. We use a robust approach to normalizing predicted values (i.e. to resolve mis-spellings, etc.) described in Appendix A. We apply this normalization to our strongest baseline (IC-DST Codex) in our ablations (§ 6). When computing $P(y|f'_{prompt}(\mathcal{E}_k))$, we clip low token log probabilities at 5e-7 in the few-shot setting and 5e-4 in the zero-shot setting, as the lack of examples leads to poorer calibration in the zero-shot setting. We also clip full-sequence log probabilities at 1e-7 in the few-shot setting and 1e-5 in the zero-shot setting.

C.2 Retriever fine-tuning details

For both our methods and the re-implementation of IC-DST Codex (Hu et al., 2022) used in our ablations (§ 6), we fine-tune the retriever using the `sentence-transformers` package (Reimers and Gurevych, 2019), following the procedure of (Hu et al., 2022). We begin with pre-trained `all-mpnet-base-v2` embedding model, which we use as a retriever with nearest neighbors search[9]. Each of our retrievers is trained for 15 epochs using the `OnlineContrastiveLoss`, which computes the contrastive loss proposed by Hadsell et al. (2006) using only hard positives and hard negatives. For each dialogue turn in the training set, we use sim_{F_1} to define positive and (hard) negative examples as the top and bottom 5% of the nearest 200 examples, respectively.

[9] We use the scipy implementation: `https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.KDTree.html`

Few-Shot (5%)	
RefPyDST (random-k)	43.5
RefPyDST (top-k)	54.6
RefPyDST (full)	**57.9**

Table 7: MultiWOZ joint-goal accuracy in the 5% few-shot setting, ablating different retrieval methods. The full model includes both our trained retriever and diverse example decoding methods (§3.2). Top-k uses the trained retriever but decodes the top-k nearest examples instead of using our diverse decoding procedure. Random retrieval samples k examples from D_{train} uniformly at random

C.3 Arguments to Codex

For all methods, we make requests to OpenAI Codex with arguments `engine = 'code-davinci-002', max_tokens = 120`, and stop sequences of either `['-', '\n', ';', '#']` (IC-DST Codex baseline replication) or `["\n\n", "", "print("]` (ours). For methods which utilize nucleus sampling (Holtzman et al., 2020) with the `top_p` parameter. In the few-shot setting, we sample with `best_of=10`, keeping only `n=5` most likely results. In the zero-shot setting, we increase `best_of` to 32.

D Random Retrieval Ablation

In Table 7, we compare our retrieval methods to random retrieval, on the 20% split of the development set used in our previous ablations. For random retrieval, we sample k examples from D_{train} uniformly at random to construct $\mathcal{E}_k$. We find this significantly under-performs our learned retrieval methods, whether selecting the top-k examples or using our diverse decoding approach.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Yes, in the limitations section of the paper.

☑ A2. Did you discuss any potential risks of your work?
Yes, in the limitations section of the paper.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
1 (Introduction)

☑ A4. Have you used AI writing assistants when working on this paper?
Yes, I would occasionally ask an AI writing assistant to re-write a paragraph of my own ideas, to see if I liked any of the re-phrasings it proposed. Only the ideas and material first provided by myself are included in th epaper

B ☑ Did you use or create scientific artifacts?

Used, not created.

☑ B1. Did you cite the creators of artifacts you used?
5 (Experiments)

☒ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
We use a public open source dataset which was collected for research use and language model provided via API as product which is well known to the NLP community. Their license terms support research

☒ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
We use the MultiWOZ dataset and OpenAI Codex model as intended and in a way that would be easily understood from the context by a reader

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. The dataset does not contain information about persons, and is public/open source

☐ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Not applicable. Left blank.

☐ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Not applicable. Left blank.

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

C ☑ Did you run computational experiments?

4 (Experiments)

☒ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Yes, in 9 limitations, and in the appendix we describe our training procedure as following existing work which provided a clear computational budget.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Appendix C

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
5 (Results)

☐ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Not applicable. Left blank.

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
Not applicable. Left blank.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Not applicable. Left blank.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Not applicable. Left blank.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Not applicable. Left blank.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
Not applicable. Left blank.