A Few-Shot Semantic Parser for Wizard-of-Oz Dialogues with the Precise ThingTalk Representation

Giovanni Campagna Sina J. Semnani Ryan Kearns Lucas Jun Koba Sato
Silei Xu Monica S. Lam
Computer Science Department
Stanford University
Stanford, CA, USA
{gcampagn,sinaj,kearns,satojk,silei,lam}@cs.stanford.edu

Abstract

Previous attempts to build effective semantic parsers for Wizard-of-Oz (WOZ) conversations suffer from the difficulty in acquiring a high-quality, manually annotated training set. Approaches based only on dialogue synthesis are insufficient, as dialogues generated from state-machine based models are poor approximations of real-life conversations. Furthermore, previously proposed dialogue state representations are ambiguous and lack the precision necessary for building an effective agent.

This paper proposes a new dialogue representation and a sample-efficient methodology that can predict precise dialogue states in WOZ conversations. We extended the ThingTalk representation to capture all information an agent needs to respond properly. Our training strategy is sample-efficient: we combine (1) few-shot data sparsely sampling the full dialogue space and (2) synthesized data covering a subset space of dialogues generated by a succinct state-based dialogue model. The completeness of the extended ThingTalk language is demonstrated with a fully operational agent, which is also used in training data synthesis.

We demonstrate the effectiveness of our methodology on MultiWOZ 3.0, a reannotation of the MultiWOZ 2.1 dataset in ThingTalk. ThingTalk can represent 98% of the test turns, while the simulator can emulate 85% of the validation set. We train a contextual semantic parser using our strategy, and obtain 79% turn-by-turn exact match accuracy on the reannotated test set.[1]

1 Introduction

Virtual assistants and task-oriented dialogue agents are transforming how consumers interact with computers. This has led to active research on dialogue state tracking networks (Ren et al., 2019; Zhou and Small, 2019; Zhang et al., 2020; Chen et al., 2020; Heck et al., 2020), and even full neural networks that track dialogue states, implement dialogue policies, and generate agent utterances (Williams and Zweig, 2016; Eric et al., 2017; Zhang et al., 2020; Peng et al., 2020; Hosseini-Asl et al., 2020).

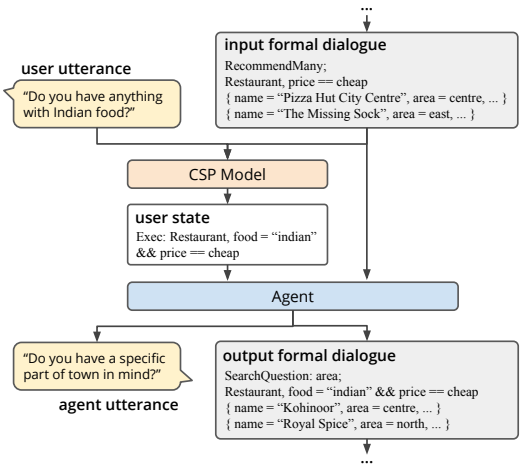

Figure 1: The inference-time flow of a dialogue agent with a contextual semantic parser based on the ThingTalk representation.

Dialogue state tracking on Wizard-of-Oz task-oriented conversations, where humans are asked to simulate both the agent and the user, has proven to be challenging. For example, despite multiple rounds of manual annotation, the MultiWOZ multi-domain task-oriented dataset still contains significant errors which hamper the development of accurate semantic parsers (Zang et al., 2020; Han et al., 2020; Ye et al., 2021a). An approach to bypass manual annotations is to generate dialogues using a simulator and then manually paraphrase them (Shah et al., 2018). Unfortunately, as we shall show in this paper, such dialogue simulators do not exercise many of the possible dialogue flows seen in Wizard-of-Oz conversations. This gap is likely to widen with real-life conversations.

Given the many attempts to create accurate semantic parsers for the MultiWOZ data set, this

[1] Our data and code can be downloaded from https://oval.cs.stanford.edu/releases/

Findings of the Association for Computational Linguistics: ACL 2022, pages 4021 - 4034

paper takes a fresh look at the problem of understanding Wizard-of-Oz conversations. We observe two fundamental flaws with the current approach. Previously proposed state representations such as slot-value pairs and the recently proposed hierarchical forms (Cheng et al., 2020) do not capture critical details in the user utterances, such as logical "or" and negation. Even if the semantic parser is 100% accurate, the agent will not be able to satisfy the user's request. Second, it is easy to make errors. The existing slot representation is ambiguous, so it is not possible to be consistently correct. This leads to poor quality of annotation.

This paper shows that it is possible to create a precise and accurate semantic parser for Wizard-of-Oz conversations in a sample-efficient manner. We introduce the MultiWOZ 3.0 dataset, a reannotation of the full test set and partial validation set of MultiWOZ 2.1 (Eric et al., 2019), using a new, more precise formal representation. The contributions of this paper include:

1. A precise, complete, executable ThingTalk representation for dialogues. In previous work, we proposed the ThingTalk programming language to represent just a single utterance (Campagna et al., 2019). Here we extend it to a full formal representation of a dialogue, including multiple turns of user input, results from the user request (such as a database lookup or API invocation), and the agent's response. We show that the extended ThingTalk for dialogues is precise enough to capture 98% of the turns in MultiWOZ 3.0. In the rest of the paper, we will refer to the extended ThingTalk language as ThingTalk, unless noted otherwise.

We also demonstrate that ThingTalk is a complete representation for dialogues. The agent directly executes the ThingTalk representation to retrieve the results from the databases and APIs, without referring to any of the user utterances. In fact, the same agent code can be used both during simulation and in a real agent deployment.

2. We show that we can obtain a high-quality synthetic training data set with a simulator that adopts the ThingTalk representation. The precision of ThingTalk makes it possible to generate many distinctively different dialogue paths that mirror those in the WOZ conversation. Our experiment shows that our simulator can generate 85% of the user turns.

3. We show that by leveraging synthesized dialogues represented in ThingTalk, we can train an effective semantic parser for WOZ conversations. This is significant since it is difficult to annotate dialogues accurately. ThingTalk does not make it easier to annotate, but it is unambiguous. We annotate manually only a *few-shot* training set, and rely on synthesis for the rest. The few-shot training data is 2% of the typical amount of annotated data.

The few-shot training samples in ThingTalk help the semantic parser generalize from the simulated dialogues to WOZ conversations. Whereas the simulator can only generate a subset of the states representable by ThingTalk, ThingTalk can precisely represent nearly all WOZ data.

Our novel contextual semantic parser, described in Section 5, obtains a turn-by-turn accuracy of 79% on MultiWOZ 3.0. Note that this model generalizes to utterances that fall out of the realm of simulation.

2 Related Work

State Representation for DST *Dialogue State Tracking* is the task of predicting a formal representation of a conversation. The standard representation used in DST contains the values of all slots mentioned in the dialogue (El Asri et al., 2017; Budzianowski et al., 2018). This is inadequate in practice. First of all, the definition is ambiguous, as it could mean "all slots mentioned by the user" or "all slots mentioned by either the user or the agent". This has lead to inconsistency in the annotation. Second, the representation does not track the comparison or logical operators in the request, so it cannot model complex queries.

Recently, Cheng et al. (2020) proposed adopting a formal representation for both the user and agent state, using the TreeDST representation. TreeDST was built to support only dialogues synthesized and paraphrased from a compatible state machine, while ThingTalk supports the full generality of Wizard-of-Oz conversations.

Data Acquisition for DST In recent years, a number of very large DST datasets have been released (Budzianowski et al., 2018; Byrne et al., 2019; Rastogi et al., 2020). The preferred technique to acquire such datasets is through Wizard-of-Oz (Kelley, 1984), a technique in which two humans are instructed to converse with each other, with one person taking the role of the agent. WOZ datasets are expensive, and the annotation quality is poor. A different approach synthesizes a large

corpus of dialogues using a state machine, then employs crowdworkers to paraphrase them. Paraphrasing has been applied to semantic parsing (Wang et al., 2015) and dialogues (Shah et al., 2018; Rastogi et al., 2020; Cheng et al., 2020). Paraphrased datasets have less variety than WOZ, and crowdsourced paraphrases are also expensive. Our approach has a significant cost advantage, while matching the variety of WOZ dialogues.

Campagna et al. (2020) found that using data synthesized from a small finite state machine, it is possible to increase the accuracy of DST in the transfer learning setting. Later, Yu et al. (2021) proposed using synthesized data to pre-train a DST model, using a different objective function. They showed modest improvements in MultiWOZ 2.1, using the full training set. We instead propose using the same fine-tuning objective for both synthesized and few-shot annotated data.

3 The ThingTalk Dialogue Language

The ThingTalk Dialogue Language is designed to formally capture all relevant information in task-oriented dialogues to interpret what the user says next. This includes the user utterances, the result of the user requests, as well as the agent's replies.

To see why the results and the agent's reply are needed, consider the example in Fig. 1. The user has previously asked for a cheap restaurant, and now asks "Do you have anything with Indian food?". In the example, the agent noted that there are many cheap restaurants available, so it is likely that the user wants both "Indian" and "cheap". This is reflected in the query that the command maps to. Conversely, had the agent responded that there are no cheap restaurants, it is likely that the user no longer cares about finding a cheap and only wants Indian. The user query thus would be just:

Exec : Restaurant, *food* = "indian"

This illustrates that the meaning of the user utterance depends on the result and the agent's response, so we must include them in the formal dialogue. The previous slot-based representation captures only what is mentioned by the user; it is not precise enough to handle this example.

Formally, ThingTalk represents (1) the user state $u \in U$ with the semantics of a single user turn, (2) the agent state $a \in A$ with the semantics of the single agent turn, and (3) the formal dialogue $d \in \mathcal{D}$ to capture all information necessary to interpret the user utterance. In this section, we provide the detailed definition of each component. The formal syntax is included in Appendix A.

(a) Sorting and ranking in ThingTalk
Agent: There are 14 trains that arrive by 12:45. What time would you like to leave?
User: What's the latest train i can take that will still get me there by 12:45?
u_1 = Exec : `sort`(*arrive_by* `desc of` *Train*, *arrive_by* $\leq$ 12:45 `&&` ...)[1]

(b) Projection and logical operators in ThingTalk
User: I think i would like to visit both churchill and magdalene colleges. May I have their phone numbers?
u_1 = Exec : [*phone*] `of` Attraction, *name* = "churchill" || *name* = "magdalene"

Figure 2: ThingTalk representations of user utterance examples in the MultiWOZ 3.0 validation set. u_1 denotes the user state.

Agent: [...] Would you like me to make you a reservation?
User: Yes, please make a reservation.
u_1 = Exec : Restaurant.MakeReservation(*name* = "...")
Agent: What day and time?
a_1 = SlotFill: *book_day*, *book_time*
Restaurant.MakeReservation(*name* = "...")

(a) User answers the question
User: At 17:30 on Friday.
u_2 = Exec : Restaurant.MakeReservation(*name* = "...", *book_time* = 17:30, *book_day* = friday);

(b) Or, user switches to a new domain instead
User: Nevermind. Not at this time. Can you help me find the postcode for the Holiday Inn Cambridge?
u_2 = Exec : Hotel, *name* = "holiday inn cambridge";

Figure 3: Examples of a user continuing or abandoning a transaction, adapted from the MultiWOZ 3.0 validation set. The user state u_2 denotes this fact by propagating or discarding the action. a_1 is the agent state.

User State. The formal semantics of a user turn is represented by a *user state* $u \in U$, which consists of an abstract dialogue act and, for dialogue acts that provide or request information, a sequence of *statements*: either database queries, or actions with side effects (such as making a reservation). Queries specify the domain of interest and can use the standard relational operators: selection, projection, aggregation, sorting. Actions specify the domain, the action name, and the parameters necessary for the action. User state examples in Figures 1 and 2 with abstract act "Exec" are all queries, while the example in Fig. 3 uses the action "Restaurant.MakeReservation".

The user state includes new statements that are implied by the current utterance and statements that the user has previously mentioned and is still interested in pursuing (Fig. 3). Note that a single user utterance may map to multiple ThingTalk statements, possibly in different domains.

Feature	Slots	TreeDST	Express	TT
User				
Executable Semantics	×	×	✓	✓
Canonicalizable	×	×	×	✓
Greetings	×	×	?	✓
Learn More, Ask Recomm.	×	×	?	✓
Multi-domain Turns	×	✓	✓	✓
Request Features:				
Slot Constraints	✓	✓	✓	✓
Comparisons	×	✓	✓	✓
Logical And	✓	✓	✓	✓
Logical Or, Not	×	×	✓	✓
Projection	×	✓	✓	✓
Ranking	×	×	?	✓
Agent				
Dialogue Acts	×	✓	✓	✓
Requested Slots	✓	✓	✓	✓
Proposed Slots	×	✓	✓	✓

Table 1: Comparison of representation power for different lexical features of different formal dialogue languages. TreeDST refers to Cheng et al. (2020), Express refers to Andreas et al. (2020). TT indicates ThingTalk.

Agent State. Analogously, each agent turn has a formal *agent state* $a \in A$ representation, which is computed by the agent policy. The agent state includes an abstract dialogue act, as well as an optional agent statement, which either requests some slots from the user, proposes a new statement to the user, or asks the user to confirm an action.

Formal Dialogue Representation. A formal dialogue $d \in \mathcal{D}$ captures all the information in the conversation needed to interpret the user utterance. Specifically, it contains the current agent state, the accumulated results of executing the user statements in previous turns, and the user statements that the user has asked to execute but that are missing some required parameters. The results for queries are the items retrieved from the database; the results for actions are returned by the API call.

Comparison with previous representations In Table 1 we compare ThingTalk with three existing state representation: the slots and values representation used in MultiWOZ, the TreeDST representation (Cheng et al., 2020), and the Express representation (Andreas et al., 2020; Tellman, 2021). Note that neither Express nor TreeDST are open-source or available to use, whereas ThingTalk is fully open-source and comes with tools that developers can use. Limited documentation exists for Express, so we use "?" for features we do not know are supported or not.

ThingTalk represents user queries and commands as *executable* database queries and API calls. An executable representation is easier to annotate manually. Other approaches require annotators to be familiar with the semantics of each domain, whereas in our approach annotators just need to learn the database query syntax to annotate for different domains. Additionally, the implementation of the agent only needs to execute ThingTalk statements; no custom per-domain logic is necessary.

Furthermore, ThingTalk is *canonicalizable*: the annotation of the semantics of a turn is syntactically unique, regardless of how the turn is phrased, and the unique form can be computed automatically. This is important both to enforce conventions on manually annotated data, as well as to be able to paraphrase: if the annotation depends on the syntactic form of the utterance, the annotation must be changed after paraphrasing. Express, while executable, is not canonicalizable because it represents coreferences explicitly and expresses updates to the dialogue state as edits. Both features lead to syntactically different representations for the same semantics, for example if the coreference is by name, by constraints, or by pronoun.

ThingTalk can represent the full generality of WoZ conversations. For example, ThingTalk can represent turns that have no request, at the beginning and end of the conversation. Neither slots nor TreeDST have a representation for those turns. This oversight highlights the need to design the representation based on real conversations.

One feature present in the previous representation that we drop from ThingTalk is the precise slots mentioned by the agent. For example, in response to a user asking for a restaurant, the agent may mention the restaurant "name" and "address." Such slots do not affect the interpretation of the user utterance. Removing them from the agent state coalesces many more utterances into the same state, and allows to approximate more complex human agent utterances, increasing the state coverage and boosting the accuracy of the semantic parser.

4 Simulator-Agent Architecture

To synthesize data for training, we propose a *simulator-agent* architecture. The state-based simulator takes the role of the human user. The same agent that would be used at deployment time is used during synthesis. The agent is built based on the semantics of ThingTalk, not just the simulator. It can respond correctly to any dialogue $d \in \mathcal{D}$ representable in ThingTalk. On the other hand, the

simulator samples a subset space $\mathcal{D}_{\text{Sim}} \subset \mathcal{D}$. We refer to dialogues in $\mathcal{D}_{\text{Sim}}$ as *in-simulation*; other dialogues are *out-of-simulation*.

Formally, the architecture has three components:

$\text{Agent}(d, u) : \mathcal{D} \times U \to \mathcal{D}$: an agent that accepts a formal dialogue $d \in \mathcal{D}$, and the user state $u \in U$ representing the last user utterance, to produce a new dialogue $d' \in \mathcal{D}$. The agent guarantees that if $d \in \mathcal{D}_{\text{Sim}}$ then $d' \in \mathcal{D}_{\text{Sim}}$.

$\text{Sim}(d) : \mathcal{D}_{\text{Sim}} \to X \times U$: a simulator that accepts an in-simulation dialogue $d \in \mathcal{D}_{\text{Sim}}$, and non-deterministically creates a new user utterance $x \in X$ and its user state $u \in U$.

$\text{CSP}(d, x) : \mathcal{D} \times X \to U$, a contextual semantic parsing model that accepts a dialogue $d \in \mathcal{D}$, which may not be in $\mathcal{D}_{\text{Sim}}$, and a user utterance $x \in X$ to predict the user state in U.

In this section, we describe how the components are used to synthesize training data and build a functional dialogue agent.

4.1 Training Data Synthesis

We synthesize training data for CSP as follows:

$\text{Syn}(d) : \mathcal{D}_{\text{Sim}} \to \mathcal{D}_{\text{Sim}} \times X \times U$: the synthesizer accepts a dialogue $d \in \mathcal{D}_{\text{Sim}}$ and returns a training sample produced by using Sim to generate a possible user utterance and a resulting in-simulation dialogue to be predicted, then applying the Agent to continue:

$$\begin{aligned} &\text{Syn}(d) = (d', x, u), \text{ where} \\ &(x, u) = \text{Sim}(d), d' = \text{Agent}(d, u) \end{aligned}$$

Starting with a null dialogue, we iteratively use Syn to synthesize training samples. During synthesis, the agent is called in a mock execution environment with no side effects, and it uses a non-deterministic policy that generates many possible agent behaviors. It is helpful to include many agent behaviors because it helps model the human WOZ agent.

Following Campagna et al. (2020), both the simulator and the agent policy are implemented using a domain-independent state machine which includes many natural language templates for user and agent utterances. Using the templates and a few natural language phrases for each slot, we can generate dialogues for any new domain with minimal effort.

User: Please book a table for 5 at 14:30 on wednesday at Royal Spice. I also need to find a place to stay.
$u_1 =$ Exec : Restaurant.MakeReservation(
name = "royal spice", *book_people* = 5,
book_time = 14:30, *book_day* = wednesday);
Hotel;
Agent: I was able to book your table successfully. Your reference number is kqmxil0z. Now, what type of accommodations are you looking for today?

Figure 4: Example of an out-of-simulation dialogue from the MultiWOZ 3.0 test set, where the same turn mentions two domains. The simulator never generates such a turn but the agent can reply to it.

4.2 Deployment

After training, the same agent can be used at deploy time to reply to the real user.

$\text{Deploy}(d, x) : \mathcal{D} \times X \to \mathcal{D}$: given the current dialogue, a deployable system uses CSP to map the next user utterance to a formal dialogue, which is then used by Agent to continue the dialogue. Let d_0 be the empty dialogue and user input $x_1, x_2, \ldots$

$$\begin{aligned} d_i &= \text{Deploy}(d_{i-1}, x_i) \\ &= \text{Agent}(d_{i-1}, \text{CSP}(d_{i-1}, x_i)) \end{aligned}$$

4.3 Out-of-simulation Dialogues

While the simulator can cover only the most common dialogue paths, ThingTalk is designed to be general, covering many more possible dialogues. To improve generality, the CSP is trained not only with simulated dialogues but also few-shot data annotated with the full expressiveness of ThingTalk. Correspondingly, the agent is written to handle the full representation of ThingTalk. This design makes our parser and agent more robust than those that only train with simulated dialogues. Fig. 4 shows an out-of-simulation dialogue from the MultiWOZ test set. In the example, the agent must reply to two domains at once.

We show below some of the out-of-simulation dialogue patterns handled by our agent.

- *Domain switch*: the user switches to a new domain in the middle of a discussion about another; the simulator switches domains only after completing the action.
- *Multidomain*: the user refers to two domains in the same utterance; the simulator only refers to one domain at a time.
- *Eager action parameters*: the user specifies parameters for an action before completing the query, ignoring a prompt from the agent to refine the query.

- *Abandoning transactions*: the user abandons a transaction after it has been initiated; the simulator never interrupts a transaction.

These examples illustrate the many plausible ways in which the user can change the course of a dialogue. Trying to simulate all these possibilities is infeasible, nor is it desirable, as it will worsen the distribution of the training data by overemphasizing uncommon patterns. At the same time, handling these cases is important; thus, we train with few-shot annotated data and rely on the model's inherent generalization capability.

5 Contextual Semantic Parsing Model

5.1 Model Architecture

Our CSP neural model is fine-tuned from the pre-trained BART model (Lewis et al., 2020). BART is a Transformer encoder-decoder neural network (Vaswani et al., 2017) pre-trained with the task of reconstructing noised inputs. Our model for the user encodes a concatenation of the formal dialogue and the user utterance, and is trained to generate the user state as its output.

To reduce the length of the input, the formal dialogue is truncated before feeding to the model: only the last executed query and action in each domain are kept, and the rest is discarded. Previous statements are no longer relevant; information that is still relevant is carried over in the last statement. Additionally, we encode at most one result per query. We observe that the user uses either a coreference to refer to the only/first choice, or uses the entity name. The model is trained to copy entity names from the user utterance.

We use BART-Large, with about 400M parameters. We train it with token-level cross-entropy loss and teacher forcing. Hyperparameters and preprocessing details are included in Appendix B.

5.2 Training Data

Data Synthesis. We use Syn to synthesize an initial set of training dialogues, covering all possible combinations of slots at each turn, and many possible paths in $\mathcal{D}_{\text{Sim}}$.

Automatic Paraphrasing. We apply *automatic paraphrasing with filtering* (Xu et al., 2020) to increase the variety of natural language in each turn. We use a pre-trained BART model fine-tuned on the ParaBank2 general-purpose paraphrasing dataset (Hu et al., 2019). Each user utterance is paraphrased individually. We apply *filtering* to ensure that the user state does not change for each utterance: each paraphrased utterance, with its associated formal dialogue, is passed to a model trained on synthesized data; the utterance is discarded if the model predicts a different user state than the annotation before paraphrasing.

Few-Shot Fine-Tuning. To expose the model to the variety in real-world data, we fine-tune the model with a small number of manually annotated dialogues.

Self-Training. Acquiring large fully-annotated WOZ datasets is challenging, because annotations are often erroneous. Acquiring *unannotated* WOZ datasets, on the other hand, is easier. To use such data when available, we propose using *self-training* (McClosky et al., 2006; Einolghozati et al., 2019; Zoph et al., 2020). We apply the model fine-tuned on few-shot data to unannotated input, create a training set using the predicted result as annotations, and use that to further fine-tune the model.

The annotation of WOZ dialogues requires predictions of the agent state as well, unlike the simulated dialogues where the agent state is generated automatically. We apply the same methodology as for the user states to the agent state, so as to annotate the full dialogues for training.

6 Evaluation

Our evaluation attempts to answer these research questions:

1. How well does our ThingTalk representation model Wizard-of-Oz conversations?
2. What accuracy can a model achieve in the task of predicting ThingTalk, given our training data acquisition strategy?
3. How well do our dialogue simulator and our dialogue agent approximate real dialogues?

6.1 Experimental Setting

We conduct our experiments using the MultiWOZ dataset (Budzianowski et al., 2018; Eric et al., 2019). This dataset includes English task-oriented dialogues across five domains: Attraction, Hotel, Restaurant, Taxi, and Train.

We reannotated parts of MultiWOZ 2.1 with ThingTalk annotations, and we name this version MultiWOZ 3.0. The authors of this paper reannotated the full test set and, due to a lack of time, 36% of the validation set, discarding the rest. Our

result is thus a lower-bound on the possible accuracy: with more of the validation set annotated, we expect higher test accuracy.

The slot values in our new test set differ from the original annotations in 83% of the turns. This is not surprising because others have already found problems in MultiWOZ 2.1 (Zhou and Small, 2019; Zang et al., 2020; Han et al., 2020), and because ThingTalk and the existing annotations adopt different conventions for when a slot should be included. We found mistakes in the annotations, inconsistent normalization of names, and inconsistent annotation of slots offered by the agent. We dropped 1% of test turns due to unrecoverable human errors, such as the user acting as the agent.

We use four datasets for training:

- *Synthesized* dataset, generated using our state-machine-based simulator and agent, consisting of around 1M dialogues across all five domains. The state machine has 20 abstract transitions for the agent, and 43 for the user.
- *Paraphrase* dataset, obtained by automatically paraphrasing the synthesized data.
- *Few-Shot* dataset, a split of 168 dialogues from the original validation set. This amounts to 2% of the original training set. Another 265 dialogues in the original validation set are used as the 3.0 validation set.
- *Self-Trained* dataset, obtained by self-training on the MultiWOZ training set.

Dataset statistics are detailed in Appendix C.

We use the Genie Toolkit (Campagna et al., 2019) for data synthesis and Hugging Face's Transformers library (Wolf et al., 2020) for the model.

6.2 Precision of ThingTalk

ThingTalk is designed to precisely cover the semantics of Wizard-of-Oz dialogues. We first observe that ThingTalk captures the semantics of the sentences well: it can represent the validation set in its entirety, and 99.8% of the user utterances and 97.6% of the agent utterances in the test set are representable. Overall, that comprises 97.7% of the test turns. ThingTalk cannot represent, for example, out-of-domain questions, questions that cannot be answered using the given database, and agent utterances such as asking the users to wait.

User utterances in the test set that cannot be represented are simply counted as errors, while agent utterances that cannot be represented as marked with a single "invalid" dialogue act, which is given as input to the neural model. The model can choose to ignore the invalid dialogue act and attempt to predict the correct user state regardless.

6.3 Accuracy on the MultiWOZ 3.0 Test Set

Our first experiment evaluates how well our CSP model can understand the user utterances in the MultiWOZ 3.0 dataset on four metrics.

Exact match accuracy requires the predicted user state to identically match the annotation.

Slot accuracy requires the slots provided by the user in the predicted user state to match the annotation, ignoring comparison operators, requested slots, and the dialogue act.

Turn-by-turn accuracy assumes that the gold dialogue up to the current turn is available as input.

Dialogue accuracy requires predicting the correct state for *all* the previous and current turns of a given dialogue. This is a challenging but meaningful metric because in practice, once the model fails, the conversation diverges from the WOZ dialogue.

We train our CSP model on the combination of Synthesized and Paraphrased sets, fine-tune it on the Few-Shot training set, and fine-tune it again on the Self-Trained set. Our model achieves a 79.2% turn-by-turn accuracy and 44.1% dialogue accuracy in exact match (Table 2).

To understand the role of synthesized data, we removed all synthesized data, and train with only the manually annotated few-shot data. The synthesized data improves the turn-by-turn exact match accuracy by 5.5% and the dialogue exact match accuracy by 8.4%. This shows that the low-cost automatically generated training data is effective.

We performed an ablation study on the validation set to evaluate the components of our training strategy (Table 2). We first observe that the validation accuracy is higher than the test accuracy, because we used the validation set to refine our synthesis. Training with only synthesized data already delivers a respectable 61.8% turn-by-turn accuracy; with the augmentation of auto-paraphrasing data, turn-by-turn accuracy improves 0.1%, and dialogue accuracy improves 0.4%.

The few-shot training alone delivers a high accuracy of 75.6%. When the model trained on synthesized and paraphrased data is fine-tuned with few-shot data, the accuracy is 81.0%, showing that these two approaches complement each other. Self-training further improves the turn-by-turn accuracy by 0.4%, with 1% better dialogue accuracy.

Training Strategy		Turn-by-Turn		Dialogue	
		EM	Slot	EM	Slot
Test	Full training	**79.2%**	**87.5%**	**44.1%**	**61.0%**
	Few-shot only	73.7%	81.6%	35.7%	46.3%
Dev	Full training	**81.4%**	**88.7%**	**51.9%**	**67.2%**
	− self-training	81.0%	88.0%	50.9%	65.3%
	Synth. only	61.8%	73.1%	29.1%	38.0%
	Synth. + para.	61.9%	73.3%	29.5%	37.4%
	Few-shot only	75.6%	81.7%	41.8%	51.6%

Table 2: Turn-by-turn and dialogue accuracy, both exact match (EM) and slot, of the CSP model, on the MultiWOZ 3.0 test and validation sets.

Category	% Turns	Accuracy
Trained	15.5%	93.1%
In-simulation	69.7%	82.4%
Out-of-simulation	14.7%	62.7%
Unknown agent state	6.3%	66.0%
Domain switch	4.0%	84.0%
Eager action parameters	0.9%	73.3%
Multidomain	0.8%	16.7%
Abandon transaction	0.5%	25.0%

Table 3: Turn-by-turn exact match accuracy of validation set, categorized by whether each user utterance is synthesizable by our simulator. For the unsynthesizable category, we further divide in common classes of user behavior not captured by the simulator.

6.4 Generalization of the Dialogue Model

Our strategy is to handle the complexity of Wizard-of-Oz dialogues with a combination of simulated dialogues and few-shot training samples to teach generalization beyond simulated dialogues. We analyze the validation set to understand the difference between the simulated dialogues and the Wizard-of-Oz dialogues, and its effect on accuracy.

The results are shown in Table 3. The validation set is divided into:

1. Trained: 15.5% of the validation set turns share the same formal dialogue and user state with some sample in training (ignoring the slot values). Accuracy obtained: 93.1%.
2. In-simulation: 69.7% of the validation set turns can be represented by the simulator: the formal context is contained in $\mathcal{D}_{\text{Sim}}$, and the user state can be generated by the simulator. Accuracy obtained: 82.4%.
3. Out-of-simulation: 14.7% of the validation turns require the model to generalize beyond $\mathcal{D}_{\text{Sim}}$, either through few-shot or its own generalization capabilities. Accuracy obtained: 62.7%.

Our synthesizer covers the Wizard-of-Oz conversations well. Even though our simulator and agent are built using a state machine with only 54 user transitions and 24 agent transitions, 85.2% of the validation set is in-simulation.

Model	Training Data	Accuracy
TRADE	MultiWOZ 2.1	37.3%
TRADE	0-shot 2.1	12.1%
SUMBT	MultiWOZ 2.1	39.3%
SUMBT	0-shot 2.1	18.3%
STAR	MultiWOZ 2.1	**49.9%**
CSP-NoAgent	MultiWOZ 2.1	45.6%
CSP-NoAgent	0-shot 2.1	13.3%
CSP-NoAgent	+ auto-parap.	12.2%
CSP	MultiWOZ 3.0	37.3%
CSP	Synthesized	23.6%
CSP	+ auto-parap.	25.2%

Table 4: Dialogue slot accuracy on the MultiWOZ 2.1 test set. CSP-NoAgent has no formal agent state; it encodes the previous slots, and the current agent and user utterances. 0-shot 2.1 is the synthesized data by Campagna et al. (2020). CSP was trained on MultiWOZ 3.0 but tested on 2.1.

Research that trains and validates on simulated data is missing a non-trivial population of Wizard-of-Oz dialogues. We found that 14.7% of the validation turns are representable in ThingTalk but are out-of-simulation.

Our training strategy generalizes beyond the simulated dialogues. For the out-of-simulation turns, our model achieves an accuracy of 62.7%. The model can generalize well on validation turns where the agent state is unseen in training, achieving 66% accuracy. This result speaks to the strength of using a formal representation of the agent, which avoids interpreting untrained agent utterances.

The model also reacts well to strong signals in the user utterance. The model achieves 84.1% accuracy when the user switches domains unexpectedly, and 73.3% accuracy when the user starts issuing slots for the action before completing the query.

Finally, when the user issues a command over two domains at once, the model achieves 16.7% accuracy. When the user abandons a booking transaction mid-way, the model achieves 25% accuracy. These kinds of out-of-simulation states are also rare in the few shot training set. The model can generalize, but is biased towards the common cases seen in the training data.

6.5 Dialogue History vs. Formal Context

We wish to evaluate the difference between using dialogue history, as in DST models, and using a formal context. We do so by measuring the dialogue

accuracy, which has the same definition for DST and CSP.

Because we do not have the resources to reannotate the training data with ThingTalk, we will use the MultiWOZ 2.1 training set for this experiment. For a DST parser, we use TRADE (Wu et al., 2019), SUMBT (Lee et al., 2019), and STAR (Ye et al., 2021b), three high-performing models for MultiWOZ 2.1. For CSP, we train a model we call CSP-NoAgent, which uses the same neural architecture as our CSP. Because MultiWOZ 2.1 has no formal agent state annotations, CSP-NoAgent uses the original slot-value annotation from the immediately preceding turn as the formal input context. This context, the current agent utterance, and the current user utterance are used to predict all the slots from the dialogue. This is the best approximation to ThingTalk possible given the available data; the results provide a lower bound on CSP with fully annotated training data.

The results are shown in Table 4. We see that CSP-NoAgent outperforms TRADE by 8.3% and SUMBT by 6.3% in dialogue accuracy, and is within 4% of STAR, a highly optimized model. Note that CSP-NoAgent needs *no new annotations*, and the slot representation captures only a small subset of the information in the utterances. This shows the advantage of replacing the dialogue history with a formal context. It also shows that the use of formal contexts can be applied in other representations.

For comparison, we also test our CSP on MultiWOZ 2.1, using self-predicted formal agent states. Our model, trained on MultiWOZ 3.0, reaches 37.3% dialogue accuracy in the MultiWOZ 2.1 test set. This is due to the reannotation of MultiWOZ 3.0, and because the model is trained and tested on data with different annotation conventions. Compared to the dialogue slot accuracy on MultiWOZ 3.0, we observe a gap of about 11%, which serves as a lower bound on the benefit of having experts annotate the test data. Note that our approach does not require manual annotation of a large training set, and therefore expert annotation of test data was feasible.

6.6 Comparison with Previous 0-Shot Model

Our last experiment compares our work with the zero-shot model proposed by Campagna et al. (2020). Their paper only included results with transfer learning on new domains. Here, we evaluate TRADE, SUMBT, and CSP-NoAgent trained with their synthesized data in a zero-shot fashion. The results shown in Table 4 indicate that the previous approach is inadequate, achieving only 12.1% dialogue accuracy with TRADE and 18.3% with SUMBT. CSP-NoAgent achieves 13.3% dialogue accuracy. Our approach, instead, achieves 23.6% dialogue accuracy. Adding automatic paraphrasing increases the turn-by-turn accuracy by about 3% for both models.

This result shows that our approach is much more effective in synthesizing data. In particular, it is important to represent the agent state formally when training with synthesized data, as it eliminates the need to synthesize and parse agent utterances.

7 Conclusion

This paper presents a sample-efficient methodology, based on the extended ThingTalk representation, to predict precise dialogue states in Wizard-of-Oz conversations. We achieve a turn-by-turn exact-match accuracy of 79.2% on the MultiWOZ 3.0 dataset, while using 50x less manually annotated training data than the original MultiWOZ dataset.

The proposed ThingTalk dialogue representation is precise, complete, and executable. It is *precise* enough to cover 98% of the dialogue turns in MultiWOZ. The precision enables automatic synthesis of dialogues covering 85% of the MultiWOZ data set. ThingTalk is *complete* and *executable*, as evidenced by a fully working agent that can simply execute ThingTalk queries without referring to the user input. Furthermore, the agent can handle dialogue flows beyond those that can be simulated.

The accuracy is achieved with a contextual semantic parser (CSP) where the dialogue context is represented in ThingTalk rather than the natural language dialogue history. It is trained first with auto-paraphrased synthetic data, fine-tuned with the few-shot annotated data, then self-trained.

In summary, this paper shows that with ThingTalk, we can predict WOZ dialogues accurately with training data mostly generated from a state machine. Our methodology thus combines the best of the WOZ and M2M approaches, as it can handle the more realistic WOZ dialogues, while having a low data acquisition cost like M2M.

8 Ethical Considerations

We envision that our training strategy will broaden the availability of task-oriented agents for tasks and populations not currently covered by existing large-scale datasets, due to its low annotation requirement. We have open-sourced tool set designed around our representation for bootstrapping affordable contextual semantic parsers for new domains.

Our agent was tuned and evaluated on the MultiWOZ benchmark. MultiWOZ is a crowdsourced Wizard-of-Oz dataset; WOZ datasets are known not to fully represent real-world conversations (Ganhotra et al., 2020). Further research is needed before a dialogue agent based on our methodology can be deployed in the real world. Additionally, the current version of the agent was tuned for English; future work should investigate techniques to automatically localize a contextual semantic parser, analogous to prior research done for single-turn semantic parsers (Moradshahi et al., 2020).

Our training strategy replaces manual annotation of data with automatically obtained data, which requires some additional amount of computation time. In practice, such additional compute is small: data synthesis runs in 5 hours on a single machine with no GPUs; the paraphrase dataset can be obtained in about 5 hours on a machine with 4 Nvidia T4 GPUs; training completes within 8 hours on a machine with one Nvidia V100; self-training requires 2 hours on a single Nvidia T4 GPU, and fine-tuning is another 1.5 hours on one Nvidia V100. Overall, the whole process is done with about 22 hours of compute time, well below the cost of human annotation of equivalent amounts of data. We note that the large amount of synthetic data poses no challenge to convergence in practice, so training models with a large amount of synthesized data has little effect on the compute cost.

The manually annotated portion of our dataset was obtained from the previously released MultiWOZ 2.1 dataset, a crowdsourced dataset. No crowdsourcing was employed in this paper; the data was annotated by the authors.

Acknowledgments

This work is supported by the National Science Foundation under Grant No. 1900638, and the Alfred P. Sloan Foundation under Grant No. G-2020-13938.

References

Jacob Andreas, John Bufe, David Burkett, Charles Chen, Josh Clausman, Jean Crawford, Kate Crim, Jordan DeLoach, Leah Dorner, Jason Eisner, Hao Fang, Alan Guo, David Hall, Kristin Hayes, Kellie Hill, Diana Ho, Wendy Iwaszuk, Smriti Jha, Dan Klein, Jayant Krishnamurthy, Theo Lanman, Percy Liang, Christopher H. Lin, Ilya Lintsbakh, Andy McGovern, Aleksandr Nisnevich, Adam Pauls, Dmitrij Petters, Brent Read, Dan Roth, Subhro Roy, Jesse Rusak, Beth Short, Div Slomin, Ben Snyder, Stephon Striplin, Yu Su, Zachary Tellman, Sam Thomson, Andrei Vorobev, Izabela Witoszko, Jason Wolfe, Abby Wray, Yuchen Zhang, and Alexander Zotov. 2020. Task-oriented dialogue as dataflow synthesis. *Transactions of the Association for Computational Linguistics*, 8:556–571.

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)*.

Bill Byrne, Karthik Krishnamoorthi, Chinnadhurai Sankar, Arvind Neelakantan, Ben Goodrich, Daniel Duckworth, Semih Yavuz, Amit Dubey, Kyu-Young Kim, and Andy Cedilnik. 2019. Taskmaster-1: Toward a realistic and diverse dialog dataset. 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 4506–4517.

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.

Giovanni Campagna, Silei Xu, Mehrad Moradshahi, Richard Socher, and Monica S. Lam. 2019. Genie: A generator of natural language semantic parsers for virtual assistant commands. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI 2019, pages 394–410, New York, NY, USA. ACM.

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

Jianpeng Cheng, Devang Agrawal, Héctor Martínez Alonso, Shruti Bhargava, Joris Driesen, Federico Flego, Dain Kaplan, Dimitri Kartsaklis, Lin Li, Dhivya Piraviperumal, Jason D. Williams, Hong Yu, Diarmuid Ó Séaghdha, and Anders Johannsen. 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, Online. Association for Computational Linguistics.

Arash Einolghozati, Sonal Gupta, Mrinal Mohit, and Rushin Shah. 2019. Improving robustness of task oriented dialog systems. *arXiv preprint arXiv:1911.05153*.

Layla El Asri, Hannes Schulz, Shikhar Sharma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra, and Kaheer Suleman. 2017. Frames: a corpus for adding memory to goal-oriented dialogue systems. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 207–219, Saarbrücken, Germany. Association for Computational Linguistics.

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

Mihail Eric, Lakshmi Krishnan, Francois Charette, and Christopher D. Manning. 2017. Key-value retrieval networks for task-oriented dialogue. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 37–49, Saarbrücken, Germany. Association for Computational Linguistics.

Jatin Ganhotra, Robert Moore, Sachindra Joshi, and Kahini Wadhawan. 2020. Effects of naturalistic variation in goal-oriented dialog. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4013–4020, Online. Association for Computational Linguistics.

Ting Han, Ximing Liu, Ryuichi Takanobu, Yixin Lian, Chongxuan Huang, Wei Peng, and Minlie Huang. 2020. Multiwoz 2.3: A multi-domain task-oriented dataset enhanced with annotation corrections and co-reference annotation. *arXiv preprint arXiv:2010.05594*.

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, 1st virtual meeting. 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.

J. Edward Hu, Abhinav Singh, Nils Holzenberger, Matt Post, and Benjamin Van Durme. 2019. Large-scale, diverse, paraphrastic bitexts via sampling and clustering. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 44–54, Hong Kong, China. Association for Computational Linguistics.

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

Hwaran Lee, Jinsik Lee, and Tae-Yoon Kim. 2019. Sumbt: Slot-utterance matching for universal and scalable belief tracking. *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

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

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 152–159, New York City, USA. Association for Computational Linguistics.

Mehrad Moradshahi, Giovanni Campagna, Sina Semnani, Silei Xu, and Monica Lam. 2020. Localizing open-ontology QA semantic parsers in a day using machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5970–5983, Online. Association for Computational Linguistics.

Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao. 2020. Soloist: Few-shot task-oriented dialog with a single pre-trained auto-regressive model. *arXiv preprint arXiv:2005.05298*.

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

Liliang Ren, Jianmo Ni, and Julian McAuley. 2019. Scalable and accurate dialogue state tracking via hierarchical sequence generation. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

Pararth Shah, Dilek Hakkani-Tür, Gokhan Tür, Abhinav Rastogi, Ankur Bapna, Neha Nayak, and Larry Heck. 2018. Building a conversational agent

overnight with dialogue self-play. *arXiv preprint arXiv:1801.04871*.

Zachary Tellman. 2021. Designing a framework for conversational interfaces. https://www.microsoft.com/en-us/research/group/msai/articles/designing-a-framework-for-conversational-interfaces/.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Yushi Wang, Jonathan Berant, and Percy Liang. 2015. Building a semantic parser overnight. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1332–1342. Association for Computational Linguistics.

Jason D Williams and Geoffrey Zweig. 2016. End-to-end lstm-based dialog control optimized with supervised and reinforcement learning. *arXiv preprint arXiv:1606.01269*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, 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*, pages 808–819.

Silei Xu, Sina Semnani, Giovanni Campagna, and Monica Lam. 2020. AutoQA: From databases to Q&A semantic parsers with only synthetic training data. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 422–434, Online. Association for Computational Linguistics.

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*, WWW '21, page 1598–1608, New York, NY, USA. Association for Computing Machinery.

Tao Yu, Rui Zhang, Oleksandr Polozov, Christopher Meek, and Ahmed Hassan Awadallah. 2021. SCoRE: Pre-training for context representation in conversational semantic parsing. In *International Conference on Learning Representations*.

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

Yichi Zhang, Zhijian Ou, and Zhou Yu. 2020. Task-oriented dialog systems that consider multiple appropriate responses under the same context. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9604–9611.

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

Barret Zoph, Golnaz Ghiasi, Tsung-Yi Lin, Yin Cui, Hanxiao Liu, Ekin Dogus Cubuk, and Quoc Le. 2020. Rethinking pre-training and self-training. *Advances in neural information processing systems*, 33:3833–3845.

A ThingTalk Definition

A.1 Syntax

Formal Dialogue d	: $a\ \ r^*\ \ s^*$
User State u	: $ua\ \ s^*$
Agent State a	: $aa\ \ as^?$
User Act ua	: Greet | Exec | Cancel | Insist | AskRecommend | LearnMore | ActionQuestion | End | Invalid
Agent Act aa	: Init | Greet | RecommendOne | RecommendMany | Propose | SearchQuestion | SlotFill | LearnMoreWhat | EmptySearch | Confirm | ActionSuccess | ActionError | AnythingElse | Invalid
User Statement s	: q | ac
Result r	: $s\ \ \left[\{[sn = v]^+\}\right] *$
Agent Statement as	: Request sn^+ | [Propose | Confirm] $[q$ | $ac]$
Query q	: <ThingTalk query>
Action ac	: $dn\ ([sn = v]^*)$
Domain Name dn	: <identifier>
Slot Name sn	: <identifier>
Value v	: <constant>

A.2 Agent Definition

The agent is a function $\text{Agent}(d, u) = d'$ that computes the new formal representation of the entire dialogue. The representation is constructed incrementally, starting from the initial dialogue d_0 which is empty.

Let $d = (a, r, s) \in \mathcal{D}$ and $u = (ua, s_\text{u}) \in U$ be the two inputs to the agent. The agent computes the new agent state a as follows:

$$
\begin{aligned}
(r_\text{u}, is_\text{u}) &= \text{Execute}(s_\text{u}) \\
a' &= \text{Policy}(ua, r||r_\text{u}, is_\text{u}) \\
d' &= (a', r||r_\text{u}, is_\text{u})
\end{aligned}
$$

where $||$ denotes concatenation. The Execute function calls the ThingTalk runtime to execute the statements in the user state, s_u. It returns (1) the results r_u by executing all statements in s_u whose required parameters are available, (2) the rest of the (incomplete) statements, is_u. The Policy function determines the agent state a' from the user state ua. all the results r_u appended to previous results r, and is_u. The agent returns the new dialogue d' with the new agent state, all the results and the new incomplete statements. The incomplete statements s in d are discarded. If the user has not changed topics, information in s is incorporated in s_u.

B Training

B.1 Preprocessing

We apply the same preprocessing used by TRADE (Wu et al., 2019) to the input utterances. We also use a rule-based preprocessor to identify time expressions, and replace them with placeholder tokens. All slot values in the result and agent states that have string or time type are replaced with a placeholder when input to the model.

We normalize all slot values in the user state to match the utterance, regardless of typos. When comparing the slot values for equality, we normalize entity names via a database lookup.

B.2 Hyperparameters

Our model uses a BART large model which has 400 million trainable parameters. We use the Adam optimizer, with the Transformer learning rate schedule (800 iterations of warm-up, 0.04 multiplier).

We train our model for 50,000 gradient updates on the synthesized data and choose the model with the highest validation exact-match accuracy. We then fine-tune that model on the few-shot training set for 15,000 gradient updates, and again choose the model with the highest validation accuracy. We repeat this process for another 15,000 updates on the self-train set. Training is done on a single GPU with 16 GB memory and batch size is chosen based on the length of the examples in each batch: we choose as many examples as we can fit in the GPU memory. Gradient accumulation is used to increase the effective batch size by a factor of 20.

C Dataset Statistics

	# dlgs	# turns	# words
Synthesized	968,007	830,792	11,390,957
Paraphrased	592,970	945,946	12,845,548
Few-Shot	168	1,061	14,669
Self-Training	8,420	56,546	760,927
Validation	265	1,582	21,256
Test	995	7,271	100,814

Table 5: Statistics of our training and evaluation sets: number of dialogues, of turns, and of words. For the synthesized dataset, we do not count turns that appear identically in multiple dialogues.

D Annotated Example Dialogue

Here is an example of a dialogue generated by the simulator, between the user U and the agent A. US denotes the user state, D denotes the formal dialogue.

U: List good restaurants which offer Indian.
US: Exec: Restaurant, food = “Indian”;
A: Do you have a specific part of town in mind?
D: SearchQuestion: area;
Restaurant, food = “Indian”
[{ name = “Cocum”, area = west, ... },
{ name = “Rajmahal”, area = east, ... },
...];
U: Yes I’d like near the east.
US: Exec: Restaurant, food = “Indian” && area = east;
A: I have Rajmahal and Curry Prince.
D: RecommendTwo
Restaurant, food = “Indian”
[...];
Restaurant, food = “Indian” && area = east
[{ name = “Rajmahal”, area = east, ... },
{ name = “Curry Prince”, area = east, ... },
...];
U: Is Curry Prince a restaurant serving Indian?
US: Exec: [food] of Restaurant, food = “Indian” && area = east && name = “Curry Prince”;
A: Curry Prince serves Indian. Would you like me to book it?
D: RecommendOne: Propose Restaurant.MakeReservation(name=“Curry Prince”, book_people=6);
Restaurant, food = “Indian”
[...];
Restaurant, food = “Indian” && area = east
[...];
[food] of Restaurant, food = “Indian” && area = east && name = “Curry Prince”
[{ name = “Curry Prince”, food = “Indian” }];
U: I will take it. Please make a reservation at the restaurant on Monday at 13:00.
US: Exec: Restaurant.MakeReservation(name=“Curry Prince”, book_people=6, book_day=Sunday, book_time=13:00)
A: Alright, Consider your request done! Its confirmation number is ay05tzjq. Anything else for you today?
D: ActionSuccess;
Restaurant, food = “Indian”
[...];
Restaurant, food = “Indian” && area = east
[...];
[food] of Restaurant, food = “Indian” && area = east && name = “Curry Prince”
[...];
Restaurant.MakeReservation(name=“Curry Prince”, book_people=6, book_day=Sunday, book_time=13:00)
[{ reference_number = “ay05tzjq” }];
U: Thanks, this is good for now. Goodbye!
US: End