Zero-Shot Transfer Learning with Synthesized Data for Multi-Domain Dialogue State Tracking

Giovanni Campagna  Agata Foryciarz  Mehrad Moradshahi  Monica S. Lam
Computer Science Department
Stanford University
Stanford, CA, USA
{gcampagn,agataf,mehrad,lam}@cs.stanford.edu

Abstract

Zero-shot transfer learning for multi-domain dialogue state tracking, if successful, can allow us to handle new domains without incurring the high cost of data acquisition. This paper proposes augmenting zero-shot training with synthesized dialogues for new domains, using templates derived from the annotated validation data. We show that data augmentation through synthesized data can improve the accuracy of the TRADE model and significantly boost the zero-shot learning accuracy of the BERT-based SUMBT model. The improvement obtained ranges from 4% to 32% on different domains in MultiWOZ 2.1 dataset. Our proposed technique achieves a zero-shot accuracy which is 63% to 93% of the accuracy obtained by a model trained on the full data set.

1 Introduction

Automated conversational agents can improve customer service and reduce the operating costs for just about every domain. However, training a goal-directed dialogue agent for a domain requires acquiring annotated dialogues to cover all possible conversation flows. Commonly, this is done using the Wizard-of-Oz technique (Kelley, 1984), where two crowdsource workers converse with each other, while also annotating the state of each turn. This technique has been employed to construct several datasets (Hemphill et al., 1990; Wen et al., 2016; Yu et al., 2019). Recently, it has been used to build the MultiWOZ dataset (Budzianowski et al., 2018), a large corpus of dialogues across 7 domains.

Unfortunately, not only is the initial acquisition expensive, annotating dialogues correctly has proven to be challenging due to human errors, delays in annotation, inconsistent conventions, and normalization issues (Eric et al., 2019; Zhou and Small, 2019). The MultiWoz data set still has significant inconsistencies (Zhou and Small, 2019) despite being constructed through multiple rounds of annotations (Budzianowski et al., 2018; Eric et al., 2019). In addition, annotating real-life sentences and conversations has privacy implications.

We propose a templated synthesis methodology that leverages existing training data to reduce the need for annotated data for new domains. Central to this methodology is a domain-independent model of dialogues, which is derived experimentally from existing annotated data. Each turn of a conversation consists of a pair of agent-user dialogue acts, as well as a context - consist-

1We will release our code and dataset after publication.
ing of an abstract state of the dialogue and the slot-value pairs of each domain. Dialogue templates are a grammar of natural language, with semantic functions providing the formal meaning in terms of context transitions. Domain-independent templates handle the abstract dialogue acts and domain-dependent templates handle the ontology of the domain.

Our synthesis algorithm samples the possible combinations of transitions, generating the corresponding annotated utterances, to cover the space of conversations. A full example of a dialogue that can be synthesized by our model is shown in Fig. 1. In addition, to transfer knowledge between domains with similar slots, we adapt training samples from related domains by substituting them with the vocabulary of the new domain.

To transfer knowledge to a new domain in a zero-shot setting, we train with the synthesized data for the new domain together with existing data for other domains. We can improve the accuracy of the abstract dialogue model as well as the state-tracking neural network, by iteratively refining the templates based on the error analysis on the validation data and by introducing additional annotated data in the new domain. Note that the abstract dialogue model is also applicable to the implementation of the agent.

Contributions of this paper include:

- We propose a templatized-synthesis methodology to bootstrap conversational agents in new domains.
- Our approach improves over the previous state-of-the-art result on zero-shot transfer for MultiWOZ 2.1 task by between 4 and 32 percentage points, depending on the domain.
- We compare TRADE (Wu et al., 2019), an RNN-based model, and SUMBT (Lee et al., 2019), a model based on BERT (Devlin et al., 2018). Our technique improves both models, showing that our approach is independent of the specific model used.
- Our experimental results show that synthesized data complements BERT pretraining. The BERT-based SUMBT model can, in a purely zero-shot fashion, achieve between 63% and 93% of the accuracy obtained by a model trained on the full dataset. We propose combining pretrained models with synthesized data as a general technique to bootstrap new dialogue state trackers.

2 Related Work

Dialogue Datasets and Synthesis. Synthesized data (in training and evaluation) was proposed by Weston et al. (2015) as a toy task to evaluate the ability of models to reason, and was also used in visual question answering (Johnson et al., 2017a; Hudson and Manning, 2019).

Wang et al. (2015) proposed synthesizing data, then crowdsourcing paraphrases to train semantic parsers. Various semantic parsing datasets have been generated with this technique (Su et al., 2017; Zhong et al., 2017) and the technique has also been adapted to the multiturn setting (Cheng et al., 2018; Shah et al., 2018). While it tends to be well-annotated, paraphrase data is expensive to acquire, and these datasets are very small. More recently, Campagna et al. (2019) proposed training with both a large amount of synthesized data and a small amount of paraphrase data for semantic parsing of single sentences. They show that training with synthesized data from well-tuned templates, along with paraphrase data, can perform well on real-world evaluations.

Dialogue State Tracking. Previous work on DST use different approaches ranging from using handcrafted features to elicit utterance information (Henderson et al., 2014; Wang and Lemon, 2013) to language encoders (Lee et al., 2019) and attention mechanisms (Wu et al., 2019). Mrkšić et al. (2016) use Convolutional Neural Networks to learn utterance representations. However, their models do not scale as they do not share parameters across different slots. Zhong et al. (2018) and Nouri and Hosseini-Asl (2018) propose a new global module that shares information to facilitate knowledge transfer. However, they rely on a predefined ontology. Xu and Hu (2018) use a pointer network with a Seq2Seq architecture to handle unseen slot values. Lee et al. (2019) use a pre-trained BERT model (Devlin et al., 2018) to encode slots and utterances and uses multi-head attention (Vaswani et al., 2017) to find relevant information in dialogue context for predicting slot values. Wu et al. (2019) introduce an encoder-decoder architecture with a copy mechanism and shares all model parameters between all domains. Zhou and Small (2019) formulate multi-domain DST as a question answering task and use reading comprehension techniques to generate the answers by either span or value prediction.
From State | Agent | User | To State |
---|---|---|---|
START | Greet | Greeting | START |
| | Ask by name | Info request | |
| | Ask with constraints | Search request | |
Greet | Greet | Ask by name | Info request |
| | Ask with constraints | Search request | |
Search request | Ask to refine search | Provide constraints | Search request |
| Ask question | Answer question | Search request | |
| Propose constraint | Accept constraint | Search request | |
| | Add constraints | Search request | |
Propose entity | Accept | Complete request | |
| | Add constraints | Search request | |
| | Reject | Search request | |
| | Ask slot question | Slot question | |
| | Ask info question | Info question | |
Empty search, offer change | | Change constraints | Search request |
| | Insist | Insist | |
Info request | Provide info, offer reservation | Accept | |
| | Provide reservation info | Accept | |
| | Ask info question | Accept | |
Info question | Answer, offer reservation | Accept | |
| | Provide reservation info | Accept | |
| | Thanks | Accept | Close conversation |
Slot question | Answer, offer reservation | Accept | |
| | Add constraint | Accept | Search request |
| | | | |
Insist | Repeat empty search | Apologize | Close conversation |
| | Change constraints | Search request | |
Complete request | Offer reservation | Accept | |
| | Thanks | Accept | Close conversation |
| Accept | Ask missing slots | Answer | Complete transaction |
Complete transaction | Execute | Ask transaction info | |
| | | Transaction info question | |
| | | Thanks | Close conversation |
Transaction info question | Answer | Thanks | Close conversation |
Close conversation | Anything else | Thanks | END |

Table 1: Our dialogue model for transaction and slot-filling dialogues. Each row represents one state transition.

Johnson et al. (2017b) propose single encoder-decoder models for zero-shot machine translation by encoding language and input sentence jointly, and Zhao and Eskenazi (2018) propose cross-domain zero-shot language generation using a cross-domain embedding space.

3 Template-based Dialogue Synthesis

We model dialogues using a transition system, where each transition is a turn in the dialogue. Transitions occur between contexts, which capture all the information necessary to continue the dialogue. Each context contains the current dialogue state (among a finite set), the current domain, and all the slots that have been provided or requested by the user up to that point, with either their value, a special DONTCARE marker, or a special ? marker to indicate the user is requesting that slot.

Each transition specifies a pair of agent and user utterance, which make up the dialogue turn, as well as the end context of the transition. Thus, any sequence of contexts connected by a valid transition (a path in the transition system) forms a valid dialogue. The utterances of the dialogue can be obtained from the transitions, and the annotation is obtained from each context along the path.

3.1 Dialogue Model

The transitions between the different dialogue states are summarized in Table 1. Each of the 12 dialogue states in the left-most column can transition into a dialogue state on the right-most column using the corresponding pair of dialogue acts for the agent and the user. Despite it being a small set, we have found that these transitions capture sufficient variability in the real-world MultiWOZ task.

3.2 Templates

Given a specific transition between contexts, we synthesize the agent and user utterance pair using templates, which combine a grammar of natural language with a semantic function that provides the formal meaning in terms of contexts.

Each pair of dialogue acts (Table 1) is associated with a set of domain-independent contexts.
The work of building domain-independent templates are combined to form longer phrases. To an end state, as well as how domain-specific of dialogue acts transitioning from an initial state positions from state “Search request” back to itself in transitions invoked to produce the new context for each turn, it computes a random sample of all possible transitions uniform to maximize variety and coverage. Our iterative algorithm maintains a fixed-size working set of incomplete dialogues and their current contexts, starting with just the empty dialogue with the START state. At each turn, it computes a random sample of all possible transitions out of the contexts in the working set. A fixed number of transitions are then chosen, their templates expanded and semantic functions invoked to produce the new context for each dialogue. Extended dialogues become the working set for the next iteration; unextended ones are added to the set of generated results. The algorithm proceeds for a maximum number of turns or until the working set is empty.

### 3.3 Templatized-Dialogue Synthesis

Even with a simplified dialogue model like ours, there is an exponential number of possible dialogues. We use a randomized search algorithm to sample all possible transitions uniformly to maximize variety and coverage. Our iterative algorithm maintains a fixed-size working set of incomplete dialogues and their current contexts, starting with just the empty dialogue with the START state. At each turn, it computes a random sample of all possible transitions out of the contexts in the working set. A fixed number of transitions are then chosen, their templates expanded and semantic functions invoked to produce the new context for each dialogue. Extended dialogues become the working set for the next iteration; unextended ones are added to the set of generated results. The algorithm proceeds for a maximum number of turns or until the working set is empty.

### 3.4 Training Data Adaptations

In addition to synthesizing training data from templates, our *domain adaptation* method takes advantage of similarities between slots of different domains. For example, both restaurants and hotels have locations, so we can adapt a sentence like “find me a restaurant in the city center” to “find

\[
\text{SEARCH\_REQ} := \text{“How about” NAME “?” It is a ” NP “.”}
\]

\[
\langle \text{sep}\rangle \text{Do you have” ANYNP “?”}:
\]

\[
\lambda(cxt, \text{name, np, anynp}) \rightarrow \{
\]

\[
\text{if } \text{ctx.slots} \cap \text{anynp} \neq \emptyset
\]

\[
\text{return } \bot
\]

\[
\text{ctx.slots} = \text{ctx.slots} \cup \text{anynp}
\]

\[
\text{return } \text{ctx}
\]

\[
\}
\]

\[
\text{NP} := \text{ADJ\_SLOT \text{NP}}
\]

\[
\text{NP} := \text{NP PREP \text{SLOT}}
\]

\[
\text{ANYNP} := | \text{“anything”} | \text{“something”} | \text{ADJ\_SLOT}
\]

\[
\text{NP} := \text{“restaurant”}
\]

\[
\text{ADJ\_SLOT} := \text{SLOT FOOD}
\]

\[
\text{PREP\_SLOT} := \text{“in the” SLOT AREA “of town”}
\]

\[
\text{NAME} := \text{“Curry Garden”} | \text{“Pizza Hut”} | \ldots
\]

\[
\text{SLOT FOOD} := \text{“Italian”} | \text{“Indian”} | \text{“Chinese”} | \ldots
\]

\[
\text{SLOT AREA} := \text{“north”} | \text{“south”} | \text{“east”} | \ldots
\]

\[
\lambda(cxt, \text{name, np, anynp}) \rightarrow \{
\]

\[
\text{if } \text{ctx.slots} \cap \text{anynp} \neq \emptyset
\]

\[
\text{return } \bot
\]

\[
\text{ctx.slots} = \text{ctx.slots} \cup \text{anynp}
\]

\[
\text{return } \text{ctx}
\]

\[
\}
\]

\[
\text{NP} := \text{ADJ\_SLOT NP}
\]

\[
\text{NP} := \text{NP PREP SLOTS}
\]

\[
\text{ANYNP} := | \text{“anything”} | \text{“something”} | \text{ADJ\_SLOT}
\]

\[
\text{NP} := \text{“restaurant”}
\]

\[
\text{ADJ\_SLOT} := \text{SLOT FOOD}
\]

\[
\text{PREP\_SLOT} := \text{“in the” SLOT AREA “of town”}
\]

\[
\text{NAME} := \text{“Curry Garden”} | \text{“Pizza Hut”} | \ldots
\]

\[
\text{SLOT FOOD} := \text{“Italian”} | \text{“Indian”} | \text{“Chinese”} | \ldots
\]

\[
\text{SLOT AREA} := \text{“north”} | \text{“south”} | \text{“east”} | \ldots
\]

Figure 2: Templates to generate the example interaction. The semantic function of the first template receives as input the current context, and produces the target context. This template is applicable only if the requested information is not already in the context. It produces a context with the same state and added slots. The other semantic functions are elided. The first four templates are domain-independent; the rest are domain-specific, with the last three automatically generated from the domain ontology.

The semantic function of these templates computes the target context (dialogue state, domain, and slots) given the current context. The semantic function ensures that the annotation of the synthesized turn is correct by construction. The ontology of domains is represented by domain-specific templates, whose semantic functions return the slots and their values.

Due to space constraints, we will just use an example to illustrate the template system. The following agent and user utterances separated by a delimiting token <\text{sep}> is an example of dialogue acts “(Propose Entity, Add Constraint)” that transitions from state “Search request” back to itself in the domain of restaurants, with the additional slot “food”. (Table 1)

\[
\text{Context: SEARCH\_REQUEST restaurant( . . . )}
\]

\[
\text{Agent: How about Curry Garden? It is an Indian restaurant in the south of town. <\text{sep}>}
\]

\[
\text{User: Do you have anything Italian?}
\]

\[
\text{Context: SEARCH\_REQUEST restaurant( . . . , food = “italian”)}
\]

The full set of templates needed to generate the example is shown in Fig. 2.

Domain-independent templates specify the pair of dialogue acts transitioning from an initial state to an end state, as well as how domain-specific templates are combined to form longer phrases. The work of building domain-independent templates is shared across all domains.

Each domain has a subject and a set of slot names and values. There are four kinds of domain-specific templates. *Domain Subject Templates* describe different noun phrases for identifying the domain. *Slot Name Templates* describe ways to refer to a slot name without a value, such as “cuisine”, “number of people” or “arrival time”. *Slot Value Templates* describe phrases that refer to a slot and its value; they can be a noun phrase (“restaurants with Italian food”), passive verb phrase (“restaurants called Alimentum”), active verb phrase (“restaurants that serve Italian food”), adjective-phrase (“Italian restaurants”), preposition clauses (“reservations for 3 people”). *Information Uterance Templates* describe full sentences providing information, such as “I need free parking”, or “I want to arrive in London at 17:00”. These are domain-specific because they use a domain-specific construction (“free parking”) or a domain-specific verb (“arrive”).
me a hotel in the city center”. We substitute a matching domain with the new, and its slot values to those from the target ontology.

We also generate multi-domain dialogues from existing ones. We use heuristics to identify the point where the domain switches and we concatenate single-domain portions to form a multi-domain dialogue.

4 Experimental Setting

4.1 The MultiWOZ Dataset

The MultiWOZ dataset (Budzianowski et al., 2018) is a multi-domain fully-labeled collection of human-human written conversations and has 35 slots in total from 7 domains. Each dialogue consists of a goal, multiple user and agent utterances, and a belief state. The dataset is created through crowdsourcing and has 3,406 single-domain and 7,032 multi-domain dialogues.

Of 7 domains, only 5 have correct annotations and any data in the validation or test sets. Following Wu et al. (2019) we only focus on these 5 domains in this paper. The characteristics of the domains are shown in Table 2.

4.2 Machine Learning Models

We evaluate our data synthesis technique on two state-of-the-art models for the MultiWOZ dialogue state tracking task, SUMBT (Lee et al., 2019) and TRADE (Wu et al., 2019). Here we give a brief overview of each model; further details are provided in the respective papers.

SUMBT Slot-Utterance Matching Belief Tracker (SUMBT) uses an attention mechanism over user-agent utterances at each turn to extract the slot-value information. It deploys a distance-based non-parametric classifier to generate the probability distribution of a slot-value and minimizes the log-likelihood of these values for all slot-types and dialogue turns. Specifically, their model includes four main parts: BERT (Devlin et al., 2018) language model which encodes slot names, slot values, and utterance pairs, a multi-head attention module that computes the attention context vector between slot and utterance representations, a Recurrent Neural Network (RNN) (Sutskever et al., 2014) belief tracking module, and a discriminative classifier which computes the target probabilities which are then used to calculate the loss value. The use of similarity to find relevant slot values makes the model depend on the ontology. Thus the model is not able to track unknown slot values.

TRADE TRAnsferable Dialogue statE generator (TRADE) uses a soft copy mechanism to either copy slot-values from utterance pairs or generate them using an RNN decoder. This model can produce slot-values not encountered during training. The model is comprised of three main parts: an RNN utterance encoder which generates a context vector based on the previous turns of the dialogue; a slot-gate predictor indicating which (domain, slot) pairs need to be tracked, and a state generator that produces the final word distribution at each decoder time-step.

4.3 Software and Hyperparameters

We used the Genie tool (Campagna et al., 2019) to synthesize our datasets. We incorporated our dialogue model and template library in a new version of the tool, which will be released upon publication. For each experiment, we tuned the Genie hyperparameters separately on the validation set. For the models, we use the code that was released by the respective authors, with their recommend hyperparameters.

5 Experiments

5.1 Data Synthesis

Overall, we created a total of 390 domain-independent templates for the 15 agent acts and 17 user acts (Table 1). These templates were optimized using the validation data on the “Restaurant” domain.

We also created a library of domain-specific templates for each domain in MultiWOZ. The number of templates is shown in Table 2. To simulate a zero-shot environment in which training data is not available, we derived the templates from only the validation data of that domain. We did not look at in-domain training data to design the templates, nor did we look at any test data until the results reported here were obtained.

Note that the validation and test sets are the same sets as the MultiWOZ 2.1 release.

5.2 Evaluation On All Domains

Our first experiment evaluates how our synthesized data affects the accuracy of TRADE and SUMBT on the full MultiWOZ dataset. As in previous work (Wu et al., 2019), we evaluate the Joint
Table 2: Characteristics of the MultiWOZ ontology, the MultiWOZ dataset, the template library, and the synthesized datasets for the zero-shot experiment on the 5 MultiWOZ domains. “user slots” refers to the slots the user can provide and the model must track, while “agent slots” refer to slots that the user requests from the agent (such as phone number or address). The number of turns and tokens only refers to in-domain turns. The number of dialogues cannot be summed across domains due to multi-domain dialogues.

<table>
<thead>
<tr>
<th></th>
<th>Attraction</th>
<th>Hotel</th>
<th>Restaurant</th>
<th>Taxi</th>
<th>Train</th>
</tr>
</thead>
<tbody>
<tr>
<td># user slots</td>
<td>3</td>
<td>10</td>
<td>7</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td># agent slots</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td># slot values</td>
<td>167</td>
<td>143</td>
<td>374</td>
<td>766</td>
<td>350</td>
</tr>
<tr>
<td># real dialogues</td>
<td>3,469</td>
<td>4,196</td>
<td>4,836</td>
<td>1,919</td>
<td>3,903</td>
</tr>
<tr>
<td># turns</td>
<td>10,549</td>
<td>18,330</td>
<td>18,801</td>
<td>5,962</td>
<td>16,081</td>
</tr>
<tr>
<td># tokens</td>
<td>312,569</td>
<td>572,955</td>
<td>547,605</td>
<td>179,874</td>
<td>451,521</td>
</tr>
<tr>
<td># domain subject templates</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td># slot name templates</td>
<td>15</td>
<td>17</td>
<td>21</td>
<td>18</td>
<td>16</td>
</tr>
<tr>
<td># slot templates</td>
<td>7</td>
<td>30</td>
<td>30</td>
<td>37</td>
<td>42</td>
</tr>
<tr>
<td># utterance templates</td>
<td>1</td>
<td>14</td>
<td>13</td>
<td>13</td>
<td>27</td>
</tr>
<tr>
<td># synthesized dialogues</td>
<td>6,636</td>
<td>13,300</td>
<td>9,901</td>
<td>6,771</td>
<td>14,092</td>
</tr>
<tr>
<td># synthesized turns</td>
<td>30,274</td>
<td>62,950</td>
<td>46,062</td>
<td>35,745</td>
<td>60,236</td>
</tr>
<tr>
<td># synthesized tokens</td>
<td>548,822</td>
<td>1,311,789</td>
<td>965,219</td>
<td>864,204</td>
<td>1,405,201</td>
</tr>
</tbody>
</table>

Table 3: Accuracy on the full MultiWOZ dataset (test set), with and without synthesized data.

<table>
<thead>
<tr>
<th>Model</th>
<th>Synth.</th>
<th>Joint</th>
<th>Slot Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRADE</td>
<td>no</td>
<td>44.2</td>
<td>96.5</td>
</tr>
<tr>
<td></td>
<td>yes</td>
<td>43.0</td>
<td>96.4</td>
</tr>
<tr>
<td>SUMBT</td>
<td>no</td>
<td>48.6</td>
<td>96.8</td>
</tr>
<tr>
<td></td>
<td>yes</td>
<td>46.5</td>
<td>96.6</td>
</tr>
</tbody>
</table>

Accuracy and the Slot Accuracy. Joint Accuracy measures the number of turns in which all slots are predicted correctly at once, whereas Slot Accuracy measures the accuracy of predicting each slot individually, then averages across slots. Slot Accuracy is significantly higher than Joint Accuracy because, at any turn, most slots do not appear, hence predicting an empty slot yields high accuracy for each slot. Previous results were reported on the MultiWOZ 2.0 dataset, so we reproduce all models on MultiWOZ 2.1.

Results are shown in Table 3. We observe that our augmentation, which is derived from the MultiWOZ dataset, adds no value to this set. We obtain almost identical slot accuracy, and our joint accuracy is within the usual margin of error. It also shows that it does not significantly worsen the result either.

5.3 Zero-Shot Transfer Learning

Before we evaluate zero-shot transfer learning to new domains, we first measure the accuracy obtained for each domain when trained on the full dataset. For each domain, we consider only the subset of dialogues that refers to that particular domain and only consider the slots for that domain when calculating the accuracy. In other words, suppose we have a dialogue involving an attraction and a restaurant, a prediction that gets the attraction correct but not the restaurant will count as joint-accurate for the attraction domain. That is why the joint accuracy of individual domains is uniformly higher than the joint accuracy of all the domains. We observe that the joint accuracy for TRADE varies from domain to domain, from 50.5% for “Hotel” to 74.0% for “Train”. The domain accuracy with the SUMBT model is better than that of TRADE by about 4% for all domains, except for “Taxi” where it drops by about 4.6%.

In the zero-shot experiment reported by Wu et al. (2019), the model for each domain is trained with the full dataset, except that all the slots involving the domain of interest are removed from the context of the dialog. All the slots are present in the validation and test data, however. The method they use, which we reproduced here, has poor joint accuracy, except for 60.2% for “Taxi”. The rest of the domains have a joint accuracy ranging from 13.4% for “Restaurant” to 21.0% for “Train”. Upon closer examination, we found that simply predicting “empty” for all slots would yield higher accuracy. No zero-shot results were reported for SUMBT.

We think that a more meaningful and straightforward zero-shot learning experiment is to withhold all dialogues that refer to the domain of in-
Table 4: Accuracy on the zero-shot MultiWOZ experiment (test set), with and without data augmentation. TRADE refers to Wu et al. (2019), SUMBT to Lee et al. (2019). “Original zero-shot” results are from the unmodified TRADE model, trained on MultiWOZ 2.1. “Zero-shot baseline” results are trained with the approach described in Section 5.3, and “Our zero-shot” is our data synthesis and augmentation technique. “Our zero-shot / full” is the ratio of our zero-shot vs. the same model trained on the full dataset.

<table>
<thead>
<tr>
<th>Model</th>
<th>Training</th>
<th>Attraction Joint</th>
<th>Hotel Joint</th>
<th>Restaurant Joint</th>
<th>Taxi Joint</th>
<th>Train Joint</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Slot</td>
<td>Slot</td>
<td>Slot</td>
<td>Slot</td>
<td>Slot</td>
</tr>
<tr>
<td>TRADE</td>
<td>Full dataset</td>
<td>67.3</td>
<td>87.6</td>
<td>50.5</td>
<td>91.4</td>
<td>61.8</td>
</tr>
<tr>
<td></td>
<td>Original zero-shot</td>
<td>20.3</td>
<td>55.5</td>
<td>13.7</td>
<td>65.6</td>
<td>13.4</td>
</tr>
<tr>
<td></td>
<td>Zero-shot baseline</td>
<td>22.8</td>
<td>50.0</td>
<td>19.5</td>
<td>62.6</td>
<td>16.4</td>
</tr>
<tr>
<td></td>
<td>Our zero-shot</td>
<td>34.9</td>
<td>62.2</td>
<td>28.3</td>
<td>74.5</td>
<td>35.9</td>
</tr>
<tr>
<td></td>
<td>Our zero-shot / full</td>
<td>51.9</td>
<td>71.3</td>
<td>56.0</td>
<td>81.5</td>
<td>58.1</td>
</tr>
<tr>
<td>SUMBT</td>
<td>Full dataset</td>
<td>70.8</td>
<td>89.0</td>
<td>54.0</td>
<td>92.3</td>
<td>66.1</td>
</tr>
<tr>
<td></td>
<td>Zero-shot baseline</td>
<td>22.6</td>
<td>51.6</td>
<td>19.7</td>
<td>63.2</td>
<td>16.4</td>
</tr>
<tr>
<td></td>
<td>Our zero-shot</td>
<td>54.4</td>
<td>80.4</td>
<td>34.4</td>
<td>82.4</td>
<td>44.7</td>
</tr>
<tr>
<td></td>
<td>Our zero-shot / full</td>
<td>77.7</td>
<td>90.5</td>
<td>63.7</td>
<td>89.3</td>
<td>67.6</td>
</tr>
</tbody>
</table>

To analyze the errors, we break down the result according to the turn number and number of slots in the dialogues in the test set, as shown in Fig. 3.
We perform this analysis using the TRADE model on the “Restaurant” domain, which is the largest domain in MultiWOZ. We observe that the zero-shot baseline model achieves 100% accuracy for turns with no slots, and 0% accuracy otherwise. The baseline results in the turn-number plot thus indicate the percentage of dialogues with all empty slots for each turn number. It is possible for 5-turn dialogues to have all empty slots because a multi-domain dialogue may not have filled slots of one of the domains.

By and large, the accuracy degrades for both the “full” model and the “our zero-shot” models, with the latter losing more accuracy than the former when there are 3 or 4 slots.

The accuracy drops almost linearly with increasing turn numbers for the full model. This is expected because the prediction of a turn is considered correct only if the cumulative context of the entire dialogue is correct. Our zero-shot results look similar, except the drop off increases with larger turn numbers. Modeling the first few turns in the dialogue is easier, as the user is exclusively providing information, whereas in later turns there are more interactions possible, some of which are not captured well by our dialogue model.

The results suggest that better modeling of the dialogue state on later turns may be useful. Better dialogue state modeling is worthwhile as it informs the implementation of automated agents. An interactive agent should try to correct errors as soon as possible through confirmations with the user and not have errors accumulate.

### 5.5 Few-Shot Transfer Learning

Following (Wu et al., 2019), we also evaluate the effect of mixing a small percentage of real training data in our augmented training sets. Our experiment uses a naive few-shot training strategy, in which we directly add the subset of the data to the training set.

Fig. 4 plots the joint accuracy achieved on the new domain with the addition of different percentages of real training data. The results for 0% are the same as the zero-shot experiment. The advantage of the synthesized training data decreases as the percent of real data increases, because real data is more varied, informative, and more representative of the distribution in the test set. The impact of synthesized data is more pronounced for SUMBT than TRADE for “Attraction” and “Hotel” even with 10% real data. This suggests that SUMBT needs more data to train, due to the larger model, but can also make better use of synthesized data.

### 6 Conclusion

We propose a method to synthesize dialogues for a new domain using templates extracted from a small validation set. Our technique can replace tens of thousands of crowdsourced, hand-annotated training dialogues with a few hundred validation dialogues, and less than 100 templates per domain. The templates can be built in a few person-hours per domain, not counting training time. This method is general and can be extended to any dialogue state tracking setting or dataset.

We show improvements in joint accuracy in zero-shot and few-shot for both the TRADE and BERT-based SUMBT models, but the improvement in SUMBT is notably higher, up to 32% over the baseline. SUMBT with our zero-shot technique achieves new state-of-the-art results on all 5 domains. This suggests that the BERT pretraining complements the use of synthesized data to learn the domain, and can be a general technique to bootstrap new dialogue systems.

We have packaged our work and library of templates in a tool, which allows developers to quickly bootstrap slot-filling dialogues in their domain by supplying the domain ontology and few domain-specific templates. The tool will be released open-source upon publication.
References


