# Low-Resource Domain Adaptation for Compositional Task-Oriented Semantic Parsing

**Xilun Chen** **Asish Ghoshal**

**Yashar Mehdad** **Luke Zettlemoyer** **Sonal Gupta**

Facebook Inc.
{xilun,aghoshal,mehdad,lsz,sonalgupta}@fb.com

## Abstract

Task-oriented semantic parsing is a critical component of virtual assistants, which is responsible for understanding the user's intents (set reminder, play music, etc.). Recent advances in deep learning have enabled several approaches to successfully parse more complex queries (Gupta et al., 2018; Rongali et al., 2020), but these models require a large amount of annotated training data to parse queries on new *domains* (e.g. reminder, music).

In this paper, we focus on adapting task-oriented semantic parsers to low-resource domains, and propose a novel method that outperforms a supervised neural model at a 10-fold data reduction. In particular, we identify two fundamental factors for low-resource domain adaptation: better *representation learning* and better *training techniques*. Our representation learning uses BART (Lewis et al., 2020) to initialize our model which outperforms encoder-only pre-trained representations used in previous work. Furthermore, we train with optimization-based meta-learning (Finn et al., 2017) to improve generalization to low-resource domains. This approach significantly outperforms all baseline methods in the experiments on a newly collected multi-domain task-oriented semantic parsing dataset (TOPv2[1]).

## 1 Introduction

Virtual Assistants now play an ever increasingly important role in our daily life, and can help users perform a wide spectrum of tasks ranging from setting personal reminders, checking local weather, to controlling smart home devices and online shopping. A critical step in any virtual assistant is to understand the user's intent (e.g. set reminder, get weather info, etc.) given the user utterance. In recent years, a number of successful models have emerged to tackle such task-oriented semantic parsing task, for both simple and more complex queries.

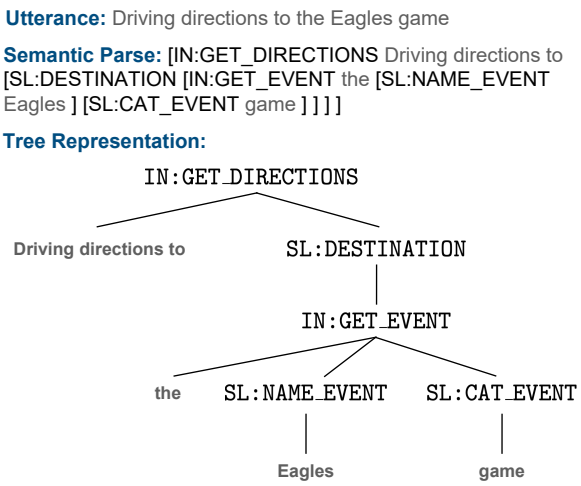


Figure 1: An compositional query from TOP dataset.

Traditionally, task-oriented semantic parsers treat the problem as a joint *intent classification* and *slot filling* task (Liu and Lane, 2016), where the model first predicts the intent of the input utterance from a set of pre-defined intent labels, and then identify all the necessary slots for that intent. For instance, for the query *"How's the weather in San Francisco?"*, the model would predict the GET_WEATHER intent, and tag *San Francisco* as a LOCATION slot. With the elevated expectation of virtual assistants, however, techniques for handling the more complex *compositional* queries have been proposed recently using neural parsers (Gupta et al., 2018) or Seq2Seq models (Rongali et al., 2020). In particular, these approaches can handle complicated queries with multiple intents or nested slots. For example, the following query from the TOP dataset (Gupta et al., 2018) "*Driving directions to the Eagles game*" is composed of a GET_DIRECTIONS intent and a GET_EVENT one nested in a tree structure (Figure 1).

[1]The dataset can be downloaded at https://fb.me/TOPv2Dataset



*Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 5090–5100,
November 16–20, 2020. 

On the other hand, most of these deep neural models require a large amount of annotated training data to achieve a good performance, which is aggravated by the fact that virtual assistants need to support hundreds of tasks each mandating a separate set of labeled training samples. Therefore, in order to support more diverse use cases without an excessive need in human annotated data, it becomes crucial that the semantic parsing model has the capability to generalize to new *tasks* or *domains* (reminder, music, etc.) with a limited number of labeled samples in the target domain. While transfer learning methods have been proposed for traditional sequence tagging models (Jaech et al., 2016; Goyal et al., 2018) to help facilitate learning slot filling models for domains with less annotated data, the efforts have been lacking when it comes to developing compositional semantic parsers in such low-resource domain adaptation setting.

Therefore, we in this paper show it is possible to build compositional task-oriented semantic parsers for low-resource domains (e.g. 25 training samples per intent or slot label), and propose a solution that is competitive against supervised models trained with 10x more data. We identify two key factors for successfully adapting task-oriented semantic parsers to new domains: better **representation learning** and better **training techniques**.

We first show that pre-trained language representations are critical in the low-resource setting for the model to quickly generalize to new intents and slots. Furthermore, most pre-trained language representations used in previous work such as BERT (Devlin et al., 2019) or RoBERTa (Liu et al., 2019) are encoder-only models, and are hence not ideal for a compositional parser with an encoder-decoder (seq2seq) architecture. We therefore propose to use BART (Lewis et al., 2020), a pre-trained seq2seq model that can be used to initialize both the encoder and decoder of our semantic parser, which significantly outperforms other pre-trained representations such as RoBERTa.

More importantly, these large pre-trained models are sometimes known to pose challenges to fine-tuning with very few training samples. In order to better adapt the semantic parser to low-resource domains, we employ *optimization-based meta-learning* (Finn et al., 2017) to improve generalization of the BART model trained on the source domains, making it easier to be fine-tuned on the target domains with very little training data.

Finally, in order to evaluate our approach, we collect a multi-domain compositional task-oriented semantic parsing dataset (`TOPv2`), based on the original `TOP` (Gupta et al., 2018) dataset. In addition to the *navigation* and *event* domains found in `TOP`, our `TOPv2` dataset has 6 new domains: *alarm, messaging, music, reminder, timer*, and *weather*, with more than $137k$ new samples. We conduct extensive experiments on this new dataset, showing that our proposed method significantly outperforms all the baseline models in the low data regime. We further show that our model achieves competitive performance compared to supervised state-of-the-art models while using 10x less data.

## 2 Problem Setup

We first formally define the task of domain adaptation (or *domain scaling*) for task-oriented semantic parsing. As illustrated in Figure 1, the task-oriented parsing task aims to predict the semantic parse given the *user utterance* (or *query*). The *semantic parse* has a tree structure and is represented using a serialized tree representation defined in the `TOP` dataset (Gupta et al., 2018). Following recent state-of-the-art practices (Rongali et al., 2020), we formulate the problem as a sequence-to-sequence (seq2seq) task, where the utterance is treated as the *source sequence* $\mathcal{S}$, while the semantic parse serves as the *target sequence* $\mathcal{T}$.

In our domain scaling setting, the goal is to develop a semantic parser with minimal training examples on a set of new *target domains*.[2] Formally, denote $\mathbb{T} = \{D_1^T, ..., D_N^T\}$ as the set of $N$ target domains. On the other hand, we assume access to training data for a number of *source domains*, which can be used to help build models on the target domains. Denote $\mathbb{S} = \{D_1^S, ..., D_M^S\}$ as the set of $M$ source domains. For each source domain $d \in \mathbb{S}$, there exists a set of annotated training data $D^d = (\mathcal{S}_d, \mathcal{T}_d)$ where $\mathcal{S}_d$ and $\mathcal{T}_d$ are the utterance and semantic parse, respectively.

For a target domain $t \in \mathbb{T}$, however, only a very limited number of training instances exist. As domains differ drastically in terms of complexity (shown in Table 1), we would expect models to require varying amount of training data for each. To normalize for such effects, we introduce a new task-specific measure of training set size: **SPIS**, which

[2]Zero-shot transfer is very challenging since a new domain has unique otherwise unseen intents and slots. Previous work (Lee and Jha, 2019) on zero-shot transfer relied on additional prior knowledge such as slot descriptions.

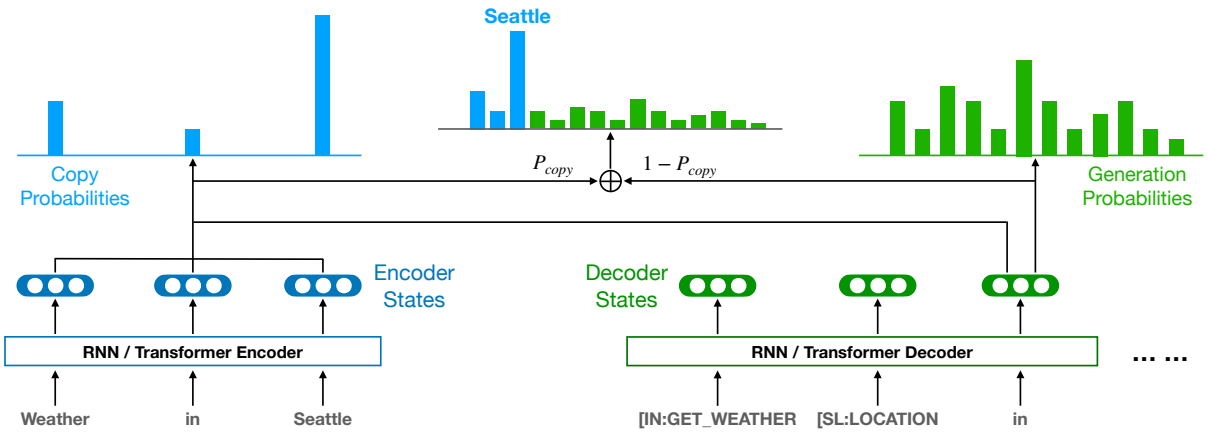


Figure 2: Overview of our sequence-to-sequence architecture with copy pointer (SEQ2SEQ-COPYPTR). Note that the target sequence is slightly different from original TOP dataset, as explained in the pre-processing steps in §5.2.

stands for *samples per intent and slot*, indicating the number of training samples available for each intent and slot label. Traditionally, at least a few hundreds of training samples are needed for each label to successfully train a deep neural semantic parser (see §5.3). In the low-resource setting, in contrast, we focus on a scenario with much less training data at 25 SPIS. That is, for each intent and slot in the target domain, 25 training samples are required[3]. (See §5.4 for discussions on more SPIS settings.) To benchmark the performance on the low-resource domains, we compare with various high-resource supervised baselines trained with much more data up to 500 or 1000 SPIS.

## 3 Base Model

In this section, we present our core model architecture. Our meta-learning technique will be introduced in Section 4. We follow recent state-of-the-art approaches (Rongali et al., 2020; Aghajanyan et al., 2020) and adopt a seq2seq model as our base architecture (SEQ2SEQ-COPYPTR), derived from the Pointer Generator Network (See et al., 2017). The base architecture is shown in Figure 2.

For an input sequence $\mathcal{S} = [w_1, w_2, \ldots, w_n]$, the encoder first encodes it into a series of hidden vectors (encoder states) $[\boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_n]$. The encoder states are then passed to an decoder that autoregressively produces target tokens $\boldsymbol{o}_t$ for each timestamp $t$. Specifically, the decoder first outputs a hidden decoder state $\boldsymbol{d}_t$ based on the decoder states from previous timestamps as well as all encoder states:

$$\boldsymbol{d}_t = Decoder(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n; \boldsymbol{d}_1, \ldots, \boldsymbol{d}_{t-1})$$

In task-oriented semantic parsing, the target sequence consists of two types of tokens: *utterance tokens* that always come from the source sequence, and *ontology tokens* that represent intent and slot labels. Therefore, two probability distributions are formulated and combined in order to produce the output token, namely the *copy probability* and the *generation probability*. The generation probability $\boldsymbol{g}_t$ is produced by the decoder by mapping the decoder state onto the output vocabulary, which only includes ontology tokens but not utterance tokens.

$$\boldsymbol{g}_t = softmax(OutputEmbed(\boldsymbol{d}_t))$$

The copy probability $\boldsymbol{c}_t$, on the other hand, indicates whether to copy one of the source tokens as the decoder output for timestamp $t$, and is predicted by using $\boldsymbol{d}_t$ as the query to perform a multi-head attention (MHA, Vaswani et al., 2017) over the encoder states.

$$\boldsymbol{c}_t, \boldsymbol{\omega}_t = \text{MHA}(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n, Linear(\boldsymbol{d}_t))$$
$$P_{copy} = sigmoid(Linear([\boldsymbol{d}_t; \boldsymbol{\omega}_t]))$$

where $\boldsymbol{c}_t$ are the attention weights indicating the copy probability and $\boldsymbol{\omega}_t$ is the attended vector used to compute a scalar $P_{copy}$ to weigh between copying and generation when constructing the final output token $\boldsymbol{o}_t$:

$$\boldsymbol{o}_t = P_{copy} \cdot \boldsymbol{c}_t + (1 - P_{copy}) \cdot \boldsymbol{g}_t$$

### 3.1 Pre-trained Language Representations

In §3, we introduce a general model framework where the encoder and decoder can be any RNN

[3]Note that one sample may contain multiple intents and slots. Empirically, only around 10 distinct samples are selected for each intent and slot (Table 2).

or Transformer (Vaswani et al., 2017) architecture. In practice, pre-trained language representations such as BERT (Devlin et al., 2019) have greatly improved performance across many NLP tasks. For task-oriented semantic parsing, in particular, Rongali et al. (2020) achieved state-of-the-art performance with RoBERTa (Liu et al., 2019). Departing from previous work, we argue that such encoder-only pre-trained models are not the most suitable choice for a seq2seq semantic parser, as it couples a pre-trained encoder with a randomly initialized decoder, resulting in challenges during training. Instead, we adopt BART (Lewis et al., 2020), a pre-trained seq2seq model, which can be used to initialize both the encoder and decoder in our SEQ2SEQ-COPYPTR model.

### 3.2 Training Stages

Finally, we clarify the terminology adopted for various training stages. Several training strategies exist for domain adaptation. For instance, one can employ *joint training* that trains a single model with all the available data on both source and target domains (with optional upsampling on the target domains). Another approach, which we found superior (Table 2), is the *pre-training + fine-tuning* strategy, where a model is first trained on the source domains and then fine-tuned on the low-resource target domains.

On the other hand, as pre-trained language representations such as RoBERTa or BART are adopted, the latter strategy becomes a 3-stage training process: train RoBERTa/BART; fine-tune on the source domains; fine-tune again on the target domains. To avoid confusion, we standardize the terminology used to refer to each of these three stages. The first stage, which is out of scope for this paper, is the **pre-training** stage where self-supervised language representations are learned. Then, it is fine-tuned on the source domains. We call this stage **source training** to avoid ambiguity with the final stage. In the final stage, which is denoted as **fine-tuning**, the source-trained model is fine-tuned again on the target domains. It is possible to omit the second stage and directly fine-tune pre-trained RoBERTa/BART on the target domains. As shown in Table 2, however, source-training significantly improves the final performance.

## 4 Meta Learning

As mentioned in §3.2, model training for low-resource domain adaptation consists of two stages: source training and fine-tuning (or target training), where the model (initialized with pre-trained representations) is first trained on the source domains and then fine-tuned on each low-resource target domain. As target domain training data is scarce, it might be challenging to effectively fine-tune a large BART model with only 25 samples per intent and slot. One reason is that traditional source training optimizes the model performance solely for the source domains, which may result in a model with strong performance on the source domains but less than ideal for transferring to new target domains via fine-tuning.

Therefore, we propose to replace source training with optimization-based meta-learning (Finn et al., 2017) to improve generalization. Instead of directly optimizing towards source domain accuracy, meta-learning, when trained on the source domains, looks for a good *initialization* $\boldsymbol{\theta}_0$ that can easily be adapted to new tasks (domains) with minimal fine-tuning. In order to learn a model that is easier to be fine-tuned on new tasks with a small amount of training data, meta-learning adopts a different training objective that explicitly optimizes for generalization by *repeatedly simulating low-resource fine-tuning* during training.

Specifically, in each iteration (episode), the MAML (Finn et al., 2017) algorithm samples two batches of training samples from a source domain $d \in \mathbb{S}$: $D_s^d$ and $D_q^d$, conventionally named the *support* and *query* set respectively. In standard source training, one simply computes the loss on $D_s^d$ and takes a gradient step to update the model. In MAML, however, low-resource fine-tuning is simulated at each training episode. Let $\boldsymbol{\theta}$ denote the model parameters being meta-learned, MAML first takes a gradient step on $D_s^d$ that leads to:

$$\boldsymbol{\theta}^d \leftarrow \boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}; D_s^d)$$

where $\eta$ is the *inner* learning rate. $\boldsymbol{\theta}^d$ can be viewed as a minimally fine-tuned model with only one fine-tuning iteration on the source domain $d$. We then use $D_q^d$ to evaluate how well $\boldsymbol{\theta}^d$ generalizes to new unseen data and update of our original model $\boldsymbol{\theta}$ with this generalization loss:

$$\begin{aligned} \boldsymbol{\theta} &\leftarrow \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}^d; D_q^d) \\ &= \boldsymbol{\theta} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}; D_s^d); D_q^d) \end{aligned} \quad (1)$$

| Domain | #Train | #Valid | #Test | #Int | #Slt | Flat% | Depth | Example Utterance |
|---|---|---|---|---|---|---|---|---|
| alarm | 20430 | 2935 | 7123 | 8 | 9 | 84% | 2.16 | *Set alarm for noon tomorrow.* |
| event | 9170 | 1336 | 2654 | 11 | 17 | 80% | 2.37 | *whats happening in san francisco tonight* |
| messaging | 10018 | 1536 | 3048 | 12 | 27 | 84% | 2.23 | *Text yes to Bill and Mindy.* |
| music | 11563 | 1573 | 4184 | 15 | 9 | 100% | 1.98 | *Repeat the last album* |
| navigation | 20998 | 2971 | 6075 | 17 | 33 | 57% | 2.68 | *I need to know if there's a lot of traffic on my way home* |
| reminder | 17840 | 2526 | 5767 | 19 | 32 | 79% | 2.45 | *erase reminder to attend conference this monday* |
| timer | 11524 | 1616 | 4252 | 11 | 5 | 96% | 2.00 | *Set a timer for 2 hours* |
| weather | 23054 | 2667 | 5682 | 7 | 11 | 100% | 1.93 | *how cold is it?* |
| **total** | 125k | 17k | 39k | 80 | 82 | 84% | 2.24 | |

| Domain | **Canonical** semantic parse for the example utterance (see §5.2) |
|---|---|
| alarm | [IN:CREATE_ALARM [SL:DATE_TIME for noon tomorrow ] ] |
| event | [IN:GET_EVENT [SL:DATE_TIME tonight ] [SL:LOCATION san francisco ] ] |
| messaging | [IN:SEND_MESSAGE [SL:CONTENT_EXACT yes ] [SL:RECIPIENT bill ] [SL:RECIPIENT mindy ] ] |
| music | [IN:REPLAY_MUSIC [SL:MUSIC_TYPE album ] ] |
| navigation | [IN:GET_INFO_TRAFFIC [SL:DESTINATION [IN:GET_LOCATION_HOME ] ] ] |
| reminder | [IN:DELETE_REMINDER [SL:DATE_TIME this monday ] [SL:TODO attend conference ] ] |
| timer | [IN:CREATE_TIMER [SL:DATE_TIME for 2 hours ] [SL:METHOD_TIMER timer ] ] |
| weather | [IN:GET_WEATHER [SL:WEATHER_ATTRIBUTE cold ] ] |

Table 1: Statistics of the TOPv2 dataset. #Int: *number of intents*; #Slt: *number of slots*; Flat%: *percentage of flat (depth $\leq 2$) queries*; Depth: *average depth of queries*. (The example in Figure 1 has a depth of 4.)

where $\alpha$ is the *outer* learning rate. Such updates are performed repeatedly on all source domains to simulate the low-resource fine-tuning scenario, which eventually learns a better initialization that only requires a small amount of data for fine-tuning to achieve good performance on the target domains. Also note that one can accumulate gradients from multiple episodes (domains) before updating the model $\boldsymbol{\theta}$, but we choose to update $\boldsymbol{\theta}$ after every episode following Antoniou et al. (2019).

Finally, MAML requires the computation of second derivatives when unrolling Equation (1), which consumes too much memory for large models such as BART. Therefore, we instead adopt a first-order meta-learning algorithm, Reptile (Nichol et al., 2018), which has shown comparative or even superior performance than MAML despite its simplicity (Dou et al., 2019). In Reptile, $k > 1$ batches of training instances $D_1^d, \ldots, D_k^d$ are sampled for a source domain $d \in \mathbb{S}$ in each episode, and the model is updated as follows:

$$\boldsymbol{\theta}^d \leftarrow \text{Adam}^k(\boldsymbol{\theta}; D_{1..k}^d, \eta),$$
$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha(\boldsymbol{\theta}^d - \boldsymbol{\theta}),$$

where $\text{Adam}^k(.; D_{1..k}^d, \eta)$ denotes performing $k$ consecutive updates on $D_1^d, \ldots, D_k^d$ using Adam (Kingma and Ba, 2015) with inner learning rate $\eta$. Note that this surprisingly simple algorithm becomes equivalent to standard source training when $k = 1$. When $k > 1$, however, Reptile behaves differently and performs similar updates compared to MAML as shown by Nichol et al. (2018) using Taylor Series analysis.

## 5 Experiments

In this section, we first introduce TOPv2, a multi-domain task-oriented semantic parsing dataset we are releasing to the community. It is an extension to the TOP dataset with 6 additional domains and $137k$ new samples. We then outline the setup of our low-resource domain scaling experiments in §5.2, and present the experimental results in §5.3.

### 5.1 The TOPv2 Dataset

While multiple datasets exist for task-oriented semantic parsing such as ATIS (Price, 1990) or SNIPS (Coucke et al., 2018), the TOP dataset (Gupta et al., 2018) is unique in that it contains compositional queries with complex and hierarchical structures (Figure 1). On the other hand, the queries from the TOP dataset are limited to only two domains, namely *navigation* and *event*, making it unsuited for domain scaling experiments. To this end, we extend the TOP dataset with 6 additional domains: *alarm*, *messaging*, *music*, *reminder*, *timer*, and *weather*, with a good mixture of simple (flat) and complex (compositional) domains. Table 1 shows some basic statistics of the TOPv2 dataset. We follow the same process of dataset collection as outlined in the TOP paper.

| № | Target Domain | reminder | | | weather | | | |
|---|---|---|---|---|---|---|---|---|
| | | #Train | #Valid | Accuracy | #Train | #Valid | Accuracy | Average |
| *Supervised models with 1000 SPIS* | | | | | | | | |
| 1 | LSTM-CopyPtr | 7552 | 2526 | 71.7 | 4197 | 2667 | 81.0 | 76.4 |
| *Supervised models with 500 SPIS* | | | | | | | | |
| 2 | LSTM-CopyPtr | 4788 | 2526 | 65.9 | 2372 | 2667 | 78.6 | 72.3 |
| 3 | RoBERTa-CopyPtr (Rongali et al., 2020) | 4788 | 2526 | 71.9 | 2372 | 2667 | 83.5 | 77.7 |
| 4 | BART-CopyPtr | 4788 | 2526 | 71.9 | 2372 | 2667 | 84.9 | 78.3 |
| *Low-Resource models with 25 SPIS* | | | | | | | | |
| 5 | LSTM-CopyPtr (FT only) | 493 | 337 | 21.5 | 176 | 147 | 46.2 | 33.8 |
| 6 | BART-CopyPtr (FT only) | 493 | 337 | 55.7 | 176 | 147 | 71.6 | 63.6 |
| 7 | BART-CopyPtr (JT) | 493 | 337 | 57.1 | 176 | 147 | 71.0 | 64.1 |
| 8 | BART-CopyPtr (JT 10x) | 493 | 337 | 59.2 | 176 | 147 | 73.3 | 66.2 |
| 9 | BART-CopyPtr (JT 100x) | 493 | 337 | 58.9 | 176 | 147 | 74.7 | 66.8 |
| 10 | LSTM-CopyPtr (ST+FT) | 493 | 337 | 45.8 | 176 | 147 | 65.1 | 55.4 |
| 11 | RoBERTa-CopyPtr (ST+FT) | 493 | 337 | 63.7 | 176 | 147 | 76.0 | 69.9 |
| 12 | BART-CopyPtr (ST+FT) | 493 | 337 | 68.0 | 176 | 147 | 75.9 | 72.0 |
| 13 | BART-CopyPtr (Reptile+FT) | 493 | 337 | **70.5** | 176 | 147 | **77.7** | **74.1** |

Table 2: Results on the TOPv2 dataset. Accuracy: Exact Match Accuracy; ST: Source Training; FT: Fine-Tuning (Target Training); JT: Joint Training; 10x: 10x Target Domain Upsampling. See more details in §5.3.

### 5.2 Experimental Setup

To evaluate our model on low-resource domain scaling for both complex and simple (flat) domains, we use *reminder* and *weather* as the target domains. The remaining 6 domains are used as source domains. As mentioned in §2, to study how much data is needed to achieve a good performance on the target domain, we adopt the SPIS strategy (samples per intent and slot) instead of selecting a fixed number of training samples for each target domain. In particular, we focus on a low-resource setting of 25 SPIS (see §5.4), where samples are randomly selected to ensure each intent and slot appears in at least 25 training instances. On the other hand, supervised models are trained with 500 or 1000 SPIS to assess the performance of our low-resource domain scaling model. For the source domains, all available training data is utilized.

**Validation Set** To perform model selection and early stopping, a validation set is adopted, which is also set to 25 SPIS for simplicity. In contrast, the supervised models utilize the entire validation set as shown in Table 2.

**Data Preprocessing** We first perform standard preprocessing such as lower-casing and tokenization. For models initialized with pre-trained language representations, BPE (Sennrich et al., 2016) tokenization is done to match that used by the pre-trained model. We do not tokenize ontology tokens (intents and slot labels) into BPE, but instead treat them as atomic tokens which are appended to the BPE vocabulary.

We then perform two additional preprocessing (canonicalization) steps, consistent across all models. First of all, note that certain utterance tokens do not contribute to the semantics of the query. For instance, in Figure 1, the phrase *Driving directions to* under `IN:GET_DIRECTIONS` and *the* under `IN:GET_EVENT` can be omitted as their semantics are already captured by the intent labels. Therefore, we only retain utterance tokens under *leaf slots* (*Eagles* and *game* in Figure 1) while removing all others. Furthermore, we sort the children of each node in the semantic parse tree in alphabetical order of the label, since the order of the children does not alter its semantic meaning. In the case of Figure 1, `SL:NAME_EVENT` and `SL:CAT_EVENT` will be reordered. We call the final semantic parse after these two preprocessing steps the **canonical form**, which will be used in all experiments.

### 5.3 Results and Discussions

Our main experimental results are summarized in Table 2. LSTM-CopyPtr utilizes BiLSTMs as both the encoder and decoder, which are commonly adopted in practice due to their smaller model size, faster inference time, and sometimes better performance when training data is sufficient (Rongali et al., 2020). RoBERTa-CopyPtr is the most simi-

lar to the model proposed by Rongali et al. (2020)[4], which uses the RoBERTa encoder and a randomly initialized transformer decoder. BART-CopyPtr is our proposed model (§3) which leverages BART to initialize both the encoder and decoder.

On the other hand, FT in the table refers fine-tuning or target training, and a FT only model trains solely on the 25 SPIS training data on a target domain. In contrast, ST+FT models first go through source training in which the models are trained on all source domain data, and are then fine-tuned on the target domain. JT stands for joint training, where the training data of the source and target domains are concatenated to jointly train the model. Since the target domains have very few samples (25 SPIS) compared to the source domains, upsampling can be conducted. For instance, JT 100x indicates that the target domain samples are duplicated 100 times before concatenated with the source domains. Finally, Reptile+FT is our meta-learning approach, where standard source training is replaced with Reptile, as described in §4.

**Pre-trained language representations** As demonstrated in Table 2, pre-trained representations is crucial in the low-resource setting with a very small amount of training data, where the knowledge encoded in these representations can dramatically improve the model's generalization. In our experiments, BART outperforms LSTM by 17% in the ST+FT setting (row 10 & 12), and 30% in the FT only setting (row 5 & 6).

Furthermore, we demonstrate that BART is a superior choice over RoBERTa to initialize our Seq2Seq-CopyPtr model, indicating that pre-training both the encoder and decoder works well for semantic parsing, even when the pretraining was based on denoising English sentences without using any logical forms. Comparing row 11 and 12, we observe a performance gap of 4.3% between BART and RoBERTa on the *reminder* domain, which is more complex with more labels and deep compositional structures. A closer look on the reminder domain also reveals that BART outperforms RoBERTa by a larger margin on compositional queries than flat ones (8.7% relative improvement on compositional queries vs. 6.3% on flat; numbers not shown in table).

[4]Our implementation (§3) is not identical to Rongali et al. (2020); please refer to their paper for the differences.

**Importance of source training** As shown in Table 2, source training also plays a critical role in low-resource domain scaling. With the non-pretrained LSTM model, source training can improve the performance from 33.8% to 55.4% (row 5 & 10), showing that source training can teach the model important inductive biases for the semantic parsing task. With BART, one hypothesis might be that source training is no longer important as BART learns a sufficiently good representation to provide model generalization. Nevertheless, this is not the case as revealed by row 6 and 12, where source training improves the BART model's performance by 8.4%.

One possible explanation is that the model can learn useful knowledge about the semantic spaces (tree structures) as well as certain intents and slots during source training. For instance, the improved accuracy on *reminder* may be explained in part by its similarity to various source domains such as *alarm* and *timer*. In addition, the target domains share some common slots with the source domains, such as `SL:DATE_TIME` and `SL:LOCATION`. When exposed to many more instances of these slot values on the source domains, the model can learn to better capture the semantics of the slots, leading to enhanced performance.

**Joint training vs. fine-tuning** In this paper, we adopt the source training + fine-tuning strategy. An alternative is joint training where the training data is combined from the source and target domains. Optionally, we can also upsample the target domains training data to increase model exposure. As shown in Table 2 (row 7-9, 12), however, joint training performs worse than ST+FT. It is a consistent empirical observation yet a curious one that fine-tuning achieves superior performance than joint training, which may deserve further investigation. Nonetheless, joint training does not suffer from the *forgetting* issue on the source domains, and may be the preferred choice for building a single model for both the source and target domains.

**Meta-learning** Finally, we show that meta-learning can improve model training for transferring to low-resource domains (row 12 & 13). When standard source training is replaced with Reptile, the accuracy of the BART model is substantially improved on both target domains and the best performance is achieved across all low-resource models (+2.1%). Even compared to supervised mod-

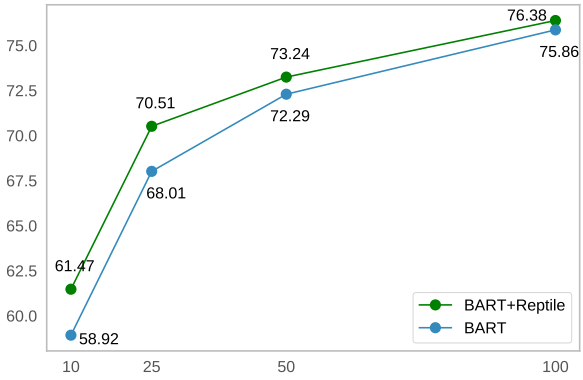


Figure 3: Performance of BART-CopyPtr (ST+FT) and BART-CopyPtr (Reptile+FT) at 10, 25, 50 and 100 SPIS on the *reminder* domain.

els trained with 500 SPIS, a 10-fold data increase, our Reptile+BART model outperforms the LSTM-based model, and is only 3.6% away from the state-of-the-art RoBERTa-based one.

### 5.4 Accuracy vs. SPIS

Figure 3 shows the performance on the *reminder* domain of our two best models, BART-CopyPtr (ST+FT) and BART-CopyPtr (Reptile+FT), at 10, 25, 50 and 100 SPIS. For each SPIS setting, we take the model with the best validation set accuracy over 5 runs. Similar to the main experiment, the validation set is selected using the same SPIS setting as the training set.

First, we observe that meta-learning proves to be beneficial for the extremely low resource setting and improves the performance of BART by about 2.5% at both 10 and 25 SPIS. The performance gap is gradually reduced as the amount of training data increases. Furthermore, we notice a steeper performance drop for both models when we go below 25 SPIS, which gives us an idea of the amount of annotated data required to achieve an acceptable performance on a new domain. In future work, we plan to push the boundary further to learn effective models with even less training data.

### 5.5 Implementation Details

For the LSTM models, both the encoder and decoder have 2 layers and a hidden size of 512. Dropout ($p = 0.4$) (Srivastava et al., 2014) is applied. Adam (Kingma and Ba, 2015) is used for optimization, with a learning rate of $10^{-3}$ for source training and $5 \times 10^{-4}$ for low-resource fine-tuning. For RoBERTa models, the encoder is a 12-layer transformer with an embedding size of 768, while the decoder is a smaller transformer model with 3 layers and an embedding size of 256. For BART models, both the encoder and decoder follow the size of the pre-trained model with 12 layers and an embedding size of 1024. For all transformer-based models, Dropout ($p = 0.3$) is applied. Adam is again used for optimization, with a learning rate of $10^{-4}$ for source training and $5 \times 10^{-5}$ for fine-tuning. In addition, the inverse square-root learning rate schedule is employed with a warmup period of 4000 updates for source training, and 2000 for fine-tuning. For meta-learning, we select $k = 5$ and a batch size of 32 for Reptile, with both inner ($\eta$) and outer ($\alpha$) learning rates being $5 \times 10^{-5}$.

All models are trained for 100 epochs on the source domains with a batch size of 128 (except Reptile), using early stopping if the validation accuracy does not improve in the last 10 epochs. Fine-tuning is done for 2000 epochs, with a batch size of either 64 (LSTM and RoBERTa) or 32 (BART and meta-learning). Model validation is performed once every 10 epochs during fine-tuning, and stops early after 20 consecutive validations with no improvements. Our model is implemented with the *fairseq* framework (Ott et al., 2019) and trained on a Nvidia Telsa P100 GPU with 16GB memory.

## 6 Related Work

**Task-Oriented Semantic Parsing** has attracted attention from the research community since 1990s with the advent of the ATIS dataset (Price, 1990). Traditionally, the task is formulated as a joint text classification (intent prediction) and sequence tagging (slot filling) problem, and can be tackled with sequence labeling models such as RNNs (Mesnil et al., 2013; Liu and Lane, 2016). These models can only parse *flat* queries with one intent and non-nested slots. More recently, a number of studies propose alternative approaches for handling the more complex *compositional* queries using neural shift-reduce parsers (Gupta et al., 2018; Einolghozati et al., 2018) or seq2seq models (Jia and Liang, 2016; Rongali et al., 2020).

On the other hand, there have been research efforts on scaling task-oriented parsers to new domains with less training data (Jaech et al., 2016; Bapna et al., 2017; Fan et al., 2017; Goyal et al., 2018; Lee and Jha, 2019). These methods, however, only focus on the simpler flat queries. Our proposed method, in contrast, can effectively parse both flat and compositional queries for low-

resource target domains.

**Meta-Learning** (Lake et al., 2015), or learning to learn, aims to learn a model that can quickly adapt to new tasks with a small amount of training data. In particular, Finn et al. (2017) propose MAML, an optimization-based meta-learning method, which learns a good parameter initialization suitable for faster adaptation to new tasks. As MAML requires to compute second derivatives, which are computation and memory intensive, there have been studies to use either first-order approximation such as first-order MAML and Reptile (Nichol et al., 2018), or implicit differentiation (Rajeswaran et al., 2019). Furthermore, meta-learning has also been applied to a number of NLP tasks lately (Gu et al., 2018; Dou et al., 2019; Mi et al., 2019; Qian and Yu, 2019; Sun et al., 2019).

## 7 Conclusion

In this work, we study the low-resource domain scaling problem for task-oriented semantic parsing. In particular, we focus on the 25 SPIS setting to investigate whether a model can effectively adapt to new *domains* with a very limited amount of training data. Our approach distinguishes itself from previous methods on two fronts. First of all, we argue the encoder-only pre-trained representations used in existing work are not ideal for the seq2seq model employed in task-oriented semantic parsing, and instead propose to use BART, a pre-trained model with an encoder-decoder architecture. More importantly, we adopt optimization-based meta-learning to improve the model's generalization to new target domains with very few training samples.

Our experiments show that our proposed method significantly outperforms all competing methods and achieves the best performance in the low-resource setting. Even when compared with supervised models trained with 500 SPIS, a 10-fold data increase, our best performing model remains competitive, and outperforms a state-of-the-art LSTM-based Pointer Generator Network (Rongali et al., 2020). Last but not least, we collect the `TOPv2` dataset, a large-scale multi-domain task-oriented semantic parsing dataset with 8 domains and more than $180k$ annotated samples to evaluate our models, which we release to the research community.

## References


Armen Aghajanyan, Jean Maillard, Akshat Shrivastava, Keith Diedrick, Michael Haeger, Haoran Li, Yashar Mehdad, Veselin Stoyanov, Anuj Kumar, Mike Lewis, and Sonal Gupta. 2020. Conversational semantic parsing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

Antreas Antoniou, Harrison Edwards, and Amos Storkey. 2019. How to train your MAML. In *International Conference on Learning Representations*.

Ankur Bapna, Gökhan Tür, Dilek Z. Hakkani-Tür, and Larry Heck. 2017. Towards zero-shot frame semantic parsing for domain scaling. In *INTERSPEECH*.

Alice Coucke, Alaa Saade, Adrien Ball, Théodore Bluche, Alexandre Caulier, David Leroy, Clément Doumouro, Thibault Gisselbrecht, Francesco Caltagirone, Thibaut Lavril, Maël Primet, and Joseph Dureau. 2018. Snips voice platform: an embedded spoken language understanding system for private-by-design voice interfaces. *CoRR*, abs/1805.10190.

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

Zi-Yi Dou, Keyi Yu, and Antonios Anastasopoulos. 2019. Investigating meta-learning algorithms for low-resource natural language understanding tasks. 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 1192–1197, Hong Kong, China. Association for Computational Linguistics.

Arash Einolghozati, Panupong Pasupat, Sonal Gupta, Rushin Shah, Mrinal Mohit, Mike Lewis, and Luke Zettlemoyer. 2018. Improving semantic parsing for task oriented dialog. In *32nd Conference on Neural Information Processing Systems - Conversational AI Workshop*.

Xing Fan, Emilio Monti, Lambert Mathias, and Markus Dreyer. 2017. Transfer learning for neural semantic parsing. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 48–56, Vancouver, Canada. Association for Computational Linguistics.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1126–1135. JMLR.org.

Anuj Kumar Goyal, Angeliki Metallinou, and Spyros Matsoukas. 2018. Fast and scalable expansion of

natural language understanding functionality for intelligent agents. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 145–152, New Orleans - Louisiana. Association for Computational Linguistics.

Jiatao Gu, Yong Wang, Yun Chen, Victor O. K. Li, and Kyunghyun Cho. 2018. Meta-learning for low-resource neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3622–3631, Brussels, Belgium. Association for Computational Linguistics.

Sonal Gupta, Rushin Shah, Mrinal Mohit, Anuj Kumar, and Mike Lewis. 2018. Semantic parsing for task oriented dialog using hierarchical representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2787–2792, Brussels, Belgium. Association for Computational Linguistics.

Aaron Jaech, Larry Heck, and Mari Ostendorf. 2016. Domain adaptation of recurrent neural networks for natural language understanding. In *INTERSPEECH*, pages 690–694.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22, Berlin, Germany. Association for Computational Linguistics.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. 2015. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338.

Sungjin Lee and Rahul Jha. 2019. Zero-shot adaptive transfer for conversational language understanding. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*, pages 6642–6649. AAAI Press.

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.

Bing Liu and Ian Lane. 2016. Attention-based recurrent neural network models for joint intent detection and slot filling. In *Interspeech 2016*, pages 685–689.

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

Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. 2013. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Interspeech 2013*.

Fei Mi, Minlie Huang, Jiyong Zhang, and Boi Faltings. 2019. Meta-learning for low-resource natural language generation in task-oriented dialogue systems. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 3151–3157. International Joint Conferences on Artificial Intelligence Organization.

Alex Nichol, Joshua Achiam, and John Schulman. 2018. On first-order meta-learning algorithms. *CoRR*, abs/1803.02999.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

P. J. Price. 1990. Evaluation of spoken language systems: the ATIS domain. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*.

Kun Qian and Zhou Yu. 2019. Domain adaptive dialog generation via meta learning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2639–2649, Florence, Italy. Association for Computational Linguistics.

Aravind Rajeswaran, Chelsea Finn, Sham M Kakade, and Sergey Levine. 2019. Meta-learning with implicit gradients. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 113–124. Curran Associates, Inc.

Subendhu Rongali, Luca Soldaini, Emilio Monti, and Wael Hamza. 2020. Don't parse, generate! a sequence to sequence architecture for task-oriented semantic parsing. In *Proceedings of The Web Conference 2020*, WWW '20, page 2962–2968, New York, NY, USA. Association for Computing Machinery.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Yibo Sun, Duyu Tang, Nan Duan, Yeyun Gong, Xiaocheng Feng, Bing Qin, and Daxin Jiang. 2019. Neural semantic parsing in low-resource settings with back-translation and meta-learning. *arXiv e-prints*, page arXiv:1909.05438.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.