Few-Shot Semantic Parsing for New Predicates

Zhuang Li, Lizhen Qu*, Shuo Huang, Gholamreza Haffari
Faculty of Information Technology
Monash University
firstname.lastname@monash.edu
shua0043@student.monash.edu

Abstract

In this work, we investigate the problems of semantic parsing in a few-shot learning setting. In this setting, we are provided with k utterance-logical form pairs per new predicate. The state-of-the-art neural semantic parsers achieve less than 25% accuracy on benchmark datasets when $k = 1$. To tackle this problem, we proposed to i) apply a designated meta-learning method to train the model; ii) regularize attention scores with alignment statistics; iii) apply a smoothing technique in pre-training. As a result, our method consistently outperforms all the baselines in both one and two-shot settings.

1 Introduction

Semantic parsing is the task of mapping natural language (NL) utterances to structured meaning representations, such as logical forms (LF). One key obstacle preventing the wide application of semantic parsing is the lack of task-specific training data. New tasks often require new predicates of LFs. Suppose a personal assistant (e.g. Alexa) is capable of booking flights. Due to new business requirement it needs to book ground transport as well. A user could ask the assistant "*How much does it cost to go from Atlanta downtown to airport?*". The corresponding LF is as follows:

(lambda $0 e (exists $1 (and (ground_transport $1)
(to_city $1 atlanta:ci)(from_airport $1 atlanta:ci)
(=(ground_fare $1) $0))))

where both *ground_transport* and *ground_fare* are new predicates while the other predicates are used in flight booking, such as *to_city*, *from_airport*. As manual construction of large parallel training data is expensive and time-consuming, we consider the *few-shot* formulation of the problem, which requires only a handful of utterance-LF training pairs for each new predicate. The cost of preparing few-shot training examples is low, thus the corresponding techniques permit significantly faster prototyping and development than supervised approaches for business expansions.

Semantic parsing in the few-shot setting is challenging. In our experiments, the accuracy of the state-of-the-art (SOTA) semantic parsers drops to less than *25%*, when there is only one example per new predicate in training data. Moreover, the SOTA parsers achieve less than 32% of accuracy on five widely used corpora, when the LFs in the test sets do not share LF *templates* in the training sets (Finegan-Dollak et al., 2018). An LF template is derived by *normalizing* the entities and attribute values of an LF into typed variable names (Finegan-Dollak et al., 2018). The few-shot setting imposes two major challenges for SOTA neural semantic parsers. First, it lacks sufficient data to learn effective representations for new predicates in a supervised manner. Second, new predicates bring in new LF templates, which are mixtures of known and new predicates. In contrast, the tasks (e.g. image classification) studied by the prior work on few-shot learning (Snell et al., 2017; Finn et al., 2017) considers an instance exclusively belonging to either a known class or a new class. Thus, it is non-trivial to apply conventional few-shot learning algorithms to generate LFs with mixed types of predicates.

To address above challenges, we present `ProtoParser`, a transition-based neural semantic parser, which applies a sequence of parse actions to transduce an utterance into an LF template and fills the corresponding slots. The parser is pre-trained on a training set with known predicates, followed by fine-tuning on a *support set* that contains few-shot examples of new predicates. It extends the attention-based sequence-to-sequence architecture (Sutskever et al., 2014) with the following

*corresponding author

Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics, pages 1281–1291
April 19 - 23, 2021.

novel techniques to alleviate the specific problems in the few-shot setting:

- *Predicate-droput*. Predicate-droput is a meta-learning technique to improve representation learning for both known and new predicates. We empirically found that known predicates are better represented with supervisely learned embeddings, while new predicates are better initialized by a metric-based few-shot learning algorithm (Snell et al., 2017). In order to let the two types of embeddings work together in a single model, we devised a training procedure called *predicate-dropout* to simulate the testing scenario in pre-training.

- *Attention regularization*. In this work, new predicates appear approximately once or twice during training. Thus, it is insufficient to learn reliable attention scores in the Seq2Seq architecture for those predicates. In the spirit of supervised attention (Liu et al., 2016), we propose to regularize them with alignment scores estimated by using co-occurrence statistics and string similarity between words and predicates. The prior work on supervised attention is not applicable, because it requires either large parallel data (Liu et al., 2016), significant manual effort (Bao et al., 2018; Rabinovich et al., 2017), or it is designed only for applications other than semantic parsing (Liu et al., 2017; Kamigaito et al., 2017).

- *Pre-training smoothing*. The vocabulary of predicates in fine-tuning is higher than that in pre-training, which leads to a distribution discrepancy between the two training stages. Inspired by Laplace smoothing (Manning et al., 2008), we achieve significant performance gain by applying a smoothing technique during pre-training to alleviate the discrepancy.

Our extensive experiments on three benchmark corpora show that `ProtoParser` outperforms the competitive baselines with a significant margin. The ablation study demonstrates the effectiveness of each individual proposed technique. The results are statistically significant with $p \leq 0.05$ according to the Wilcoxon signed-rank test (Wilcoxon, 1992).

2 Related Work

Semantic parsing There is ample of work on machine learning models for semantic parsing. The recent surveys (Kamath and Das, 2018; Zhu et al., 2019) cover a wide range of work in this area. The semantic formalism of meaning representations range from lambda calculas (Montague, 1973), SQL, to abstract meaning representation (Banarescu et al., 2013). At the core of most recent models (Chen et al., 2018; Cheng et al., 2019; Lin et al., 2019; Zhang et al., 2019b; Yin and Neubig, 2018) is SEQ2SEQ with attention (Bahdanau et al., 2014) by formulating the task as a machine translation problem. COARSE2FINE (Dong and Lapata, 2018) reports the highest accuracy on GEOQUERY (Zelle and Mooney, 1996) and ATIS (Price, 1990) in a supervised setting. IRNET (Guo et al., 2019) and RATSQL (Wang et al., 2019) are two best performing models on the Text-to-SQL benchmark, SPIDER (Yu et al., 2018). They are also designed to be able to generalize to unseen database schemas. However, supervised models perform well only when there is sufficient training data.

Data Sparsity Most semantic parsing datasets are small in size. To address this issue, one line of research is to augment existing datasets with automatically generated data (Su and Yan, 2017; Jia and Liang, 2016; Cai and Yates, 2013). Another line of research is to exploit available resources, such as knowledge bases (Krishnamurthy et al., 2017; Herzig and Berant, 2018; Chang et al., 2019; Lee, 2019; Zhang et al., 2019a; Guo et al., 2019; Wang et al., 2019), semantic features in different domains (Dadashkarimi et al., 2018; Li et al., 2020), or unlabeled data (Yin et al., 2018; Kočiskỳ et al., 2016; Sun et al., 2019). Those works are orthogonal to our setting because our approach aims to efficiently exploit a handful of labeled data of new predicates, which are not limited to the ones in knowledge bases. Our setting also does not require involvement of humans in the loop such as active learning (Duong et al., 2018; Ni et al., 2019) and crowd-sourcing (Wang et al., 2015; Herzig and Berant, 2019). We assume availability of resources different than the prior work and focus on the problems caused by new predicates. We develop an approach to generalize to unseen LF templates consisting of both known and new predicates.

Few-Shot Learning Few-shot learning is a type of machine learning problems that provides a handful of labeled training examples for a specific task. The survey (Zhu et al., 2019) gives a comprehensive overview of the data, models, and algorithms

t	Actions
t_1	*GEN* [(ground_transport v_a)]
t_2	*GEN* [(to_city v_a v_e)]
t_3	*GEN* [(from_airport v_a v_e)]
t_4	*GEN* [(= (ground_fare v_a) v_a)]
t_5	REDUCE [and :- NT NT NT NT]
t_6	REDUCE [exists :- v_a NT]
t_7	REDUCE [lambda :- v_a e NT]

Table 1: An example action sequence.

proposed for this type of problems. It categorizes the models into multitask learning (Hu et al., 2018), embedding learning (Snell et al., 2017; Vinyals et al., 2016), learning with external memory (Lee and Choi, 2018; Sukhbaatar et al., 2015), and generative modeling (Reed et al., 2017) in terms of what prior knowledge is used. (Lee et al., 2019) tackles the problem of poor generalization across SQL templates for SQL query generation in the one-shot learning setting. In their setting, they assume all the SQL templates on test set are shared with the templates on support set. In contrast, we assume only the sharing of new predicates between a support set and a test set. In our one-shot setting, only around 10% of LF templates on test set are shared with the ones in the support set of GEOQUERY dataset.

3 Semantic Parser

`ProtoParser` follows the SOTA neural semantic parsers (Dong and Lapata, 2018; Guo et al., 2019) to map an utterance into an LF in two steps: *template generation* and *slot filling*[1]. It implements a designated transition system to generate templates, followed by filling the slot variables with values extracted from utterances. To address the challenges in the few-shot setting, we proposed three training methods, detailed in Sec. 4.

Many LFs differ only in mentioned atoms, such as entities and attribute values. An LF template is created by replacing the atoms in LFs with typed slot variables. As an example, the LF template of our example in Sec. 1 is created by substituting i) a typed atom variable v_e for the entity "atlanta:ci"; ii) a shared variable name v_a for all variables "$0" and "$1".

(lambda v_a e (exists v_a (and (ground_transport v_a) (to_city v_a v_e)(from_airport v_a v_e) (=(ground_fare v_a) v_a))))

[1]Code and datasets can be found in this repository: https://github.com/zhuang-li/few-shot-semantic-parsing

Formally, let $\boldsymbol{x} = \{x_1, ..., x_n\}$ denote an NL utterance, and its LF is represented as a semantic tree $\boldsymbol{y} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, ..., v_m\}$ denotes the node set with $v_i \in \mathcal{V}$, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is its edge set. The node set $\mathcal{V} = \mathcal{V}_p \cup \mathcal{V}_v$ is further divided into a template predicate set $\mathcal{V}_p$ and a slot value set $\mathcal{V}_v$. A template predicate node represents a predicate symbol or a term, while a slot value node represents an atom mentioned in utterances. Thus, a semantic tree $\boldsymbol{y}$ is composed of an abstract tree $\tau_{\boldsymbol{y}}$ representing a template and a set of slot value nodes $\mathcal{V}_{v,\boldsymbol{y}}$ attaching to the abstract tree.

In the few-shot setting, we are provided with a train set $\mathcal{D}_{train}$, a support set $\mathcal{D}_s$, and a test set $\mathcal{D}_{test}$. Each example in either of those sets is an utterance-LF pair $(\boldsymbol{x}_i, \boldsymbol{y}_i)$. The new predicates appear only in $\mathcal{D}_s$ and $\mathcal{D}_{test}$ but *not* in $\mathcal{D}_{train}$. For K-shot learning, there are K $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ per each new predicate p in $\mathcal{D}_s$. Each new predicate appears also in the test set. The goal is to maximize the accuracy of estimating LFs given utterances in $\mathcal{D}_{test}$ by using a parser trained on $\mathcal{D}_{train} \cup \mathcal{D}_s$.

3.1 Transition System

We apply the transition system (Cheng et al., 2019) to perform a sequence of transition actions to generate the template of a semantic tree. The transition system maintains partially-constructed outputs using a *stack*. The parser starts with an empty stack. At each step, it performs one of the following transition actions to update the parsing state and generate a tree node. The process repeats until the stack contains a complete tree.

- ***GEN*** **[y]** creates a new leaf node y and pushes it on top of the stack.
- **REDUCE [r].** The reduce action identifies an implication rule $head :- body$. The rule body is first popped from the stack. A new subtree is formed by attaching the rule head as a new parent node to the rule body . Then the whole subtree is pushed back to the stack.

Table 1 shows such an action sequence for generating the above LF template. Each action produces *known* or *new* predicates.

3.2 Base Parser

`ProtoParser` generates an LF in two steps: i) template generation, ii) slot filling. The base architecture largely resembles (Cheng et al., 2019).

Template Generation Given an utterance, the task is to generate a sequence of actions $\mathbf{a} = a_1, ..., a_k$ to build an abstract tree $\tau_{\boldsymbol{y}}$.

We found out LFs often contain idioms, which are frequent subtrees shared across LF templates. Thus we apply a *template normalization* procedure in a similar manner as (Iyer et al., 2019) to preprocess all LF templates. It collapses idioms into single units such that all LF templates are converted into a compact form.

The neural transition system consists of an encoder and a decoder for estimating action probabilities.

$$P(\mathbf{a}|\boldsymbol{x}) = \prod_{t=1}^{|\mathbf{a}|} P(a_t|\boldsymbol{a}_{<t}, \boldsymbol{x}) \quad (1)$$

Encoder We apply a bidirectional Long Short-term Memory (LSTM) network (Gers et al., 1999) to map a sequence of n words into a sequence of contextual word representations $\{\mathbf{e}\}_{i=1}^{n}$.

Template Decoder The decoder applies a stack-LSTM (Dyer et al., 2015) to generate action sequences. A stack-LSTM is an unidirectional LSTM augmented with a pointer. The pointer points to a particular hidden state of the LSTM, which represents a particular state of the stack. It moves to a different hidden state to indicate a different state of the stack.

At time t, the stack-LSTM produces a hidden state $\mathbf{h}_t^d$ by $\mathbf{h}_t^d = \text{LSTM}(\mu_t, \mathbf{h}_{t-1}^d)$, where μ_t is a concatenation of the embedding of the action $\mathbf{c}_{a_{t-1}}$ estimated at time $t-1$ and the representation $\mathbf{h}_{y_{t-1}}$ of the partial tree generated by history actions at time $t-1$.

As a common practice, $\mathbf{h}_t^d$ is concatenated with an attended representation $\mathbf{h}_t^a$ over encoder hidden states to yield $\mathbf{h}_t$, with $\mathbf{h}_t = \mathbf{W}\begin{bmatrix}\mathbf{h}_t^d \\ \mathbf{h}_t^a\end{bmatrix}$, where $\mathbf{W}$ is a weight matrix and $\mathbf{h}_t^a$ is created by soft attention,

$$\mathbf{h}_t^a = \sum_{i=1}^{n} P(\mathbf{e}_i|\mathbf{h}_t^d)\mathbf{e}_i \quad (2)$$

We apply dot product to compute the normalized attention scores $P(\mathbf{e}_i|\mathbf{h}_t^d)$ (Luong et al., 2015). The *supervised attention* (Rabinovich et al., 2017; Yin and Neubig, 2018) is also applied to facilitate the learning of attention weights. Given $\mathbf{h}_t$, the probability of an action is estimated by:

$$P(a_t|\mathbf{h}_t) = \frac{\exp(\mathbf{c}_{a_t}^\intercal \mathbf{h}_t)}{\sum_{a' \in \mathcal{A}_t} \exp(\mathbf{c}_{a'}^\intercal \mathbf{h}_t)} \quad (3)$$

where $\mathbf{c}_a$ denotes the embedding of action a, and $\mathcal{A}_t$ denotes the set of applicable actions at time t. The initialization of those embeddings will be explained in the following section.

Slot Filling A tree node in a semantic tree may contain more than one slot variables due to template normalization. Since there are two types of slot variables, given a tree node with slot variables, we employ a LSTM-based decoder with the same architecture as the *Template* decoder to fill each type of slot variables, respectively. The output of such a decoder is a value sequence of the same length as the number of slot variables of that type in the given tree node.

4 Few-Shot Model Training

The few-shot setting differs from the supervised setting by having a support set in testing in addition to train/test sets. The support set contains k utterance-LF pairs per new predicate, while the training set contains only known predicates. To evaluate model performance on new predicates, the test set contains LFs with both known and new predicates. Given the support set, we can tell if a predicate is known or new by checking if it only exists in the train set.

We take two steps to train our model: i) pre-training on the training set, ii) fine-tuning on the support set. Its predictive performance is measured on the test set. We take the two-steps approach because i) our experiments show that this approach performs better than training on the union of the train set and the support set; ii) for any new support sets, it is computationally more time efficient than training from scratch on the union of the train set and the support set.

There is a distribution discrepancy between the train set and the support set due to new predicates, the meta-learning algorithms (Snell et al., 2017; Finn et al., 2017) suggest to simulate the testing scenario in pre-training by splitting each batch into a meta-support set and a meta-test set. The models utilize the information (e.g. prototype vectors) acquired from the meta-support set to minimize errors on the meta-test set. In this way, the meta-support and meta-test sets simulate the support and test sets sharing new predicates.

However, we cannot directly apply such a training procedure due to the following two reasons. First, each LF in the support and test sets is a mixture of both known predicates and new predicates. To simulate the support and test sets, the meta-support and meta-test sets should include

both types of predicates as well. We cannot assume that there are only one type of predicates. Second, our preliminary experiments show that if there is sufficient training data, it is better off training action embeddings of known predicates $\mathbf{c}$ (Eq. (3)) in a supervised way, while action embeddings initialized by a metric-based meta-learning algorithm (Snell et al., 2017) perform better for rarely occurred new predicates. Therefore, we cope with the differences between known and new predicates by using a customized initialization method in fine-tuning and a designated pre-training procedure to mimic fine-tuning on the train set. In the following, we introduce fine-tuning first because it helps understand our pre-training procedure.

4.1 Fine-tuning

During fine-tuning, the model parameters and the action embeddings in Eq. (3) for known predicates are obtained from the pre-trained model. The embedding of actions that produce new predicates $\mathbf{c}_{a_t}$ are initialized using prototype vectors as in prototypical networks (Snell et al., 2017). The prototype representations act as a type of regularization, which shares the similar idea as the deep learning techniques using pre-trained models.

A prototype vector of an action a_t is constructed by using the hidden states of the *template* decoder collected at the time of predicting a_t on a support set. Following (Snell et al., 2017), a prototype vector is built by taking the mean of such a set of hidden states $\mathbf{h}_t$.

$$\mathbf{c}_{a_t} = \frac{1}{|M|} \sum_{\mathbf{h}_t \in M} \mathbf{h}_t \tag{4}$$

where M denotes the set of all hidden states at the time of applying the action a_t. After initialization, the whole model parameters and the action embeddings are further improved by fine-tuning the model on the support set with a supervised training objective $\mathcal{L}_f$.

$$\mathcal{L}_f = \mathcal{L}_s + \lambda\Omega \tag{5}$$

where $\mathcal{L}_s$ is the cross-entropy loss and Ω is an attention regularization term explained below. The degree of regularization is adjusted by $\lambda \in \mathbb{R}^+$.

Attention Regularization We address the poorly learned attention scores $P(\mathbf{e}_i|\mathbf{h}_t^d)$ of infrequent actions by introducing a novel attention regularization. We observe that the probability $P(a_j|x_i) = \frac{\text{count}(a_j, x_i)}{\text{count}(x_i)}$ and the character similarity between the predicates generated by action a_j and the token x_i are often strong indicators of their alignment. The indicators can be further strengthened by manually annotating the predicates with their corresponding natural language tokens. In our work, we adopt $1 - dist(a_j, x_i)$ as the character similarity, where $dist(a_j, x_j)$ is normalized Levenshtein distance (Levenshtein, 1966). Both measures are in the range $[0, 1]$, thus we apply $g(a_j, x_i) = \sigma(\cdot)P(a_j|x_i) + (1 - \sigma(\cdot))char\_sim(a_j, x_i)$ to compute alignment scores, where the sigmoid function $\sigma(\mathbf{w}_p^\intercal \mathbf{h}_t^d)$ combines two constant measures into a single score. The corresponding normalized attention scores is given by

$$P'(x_i|a_k) = \frac{g(a_k, x_i)}{\sum_{j=1}^{n} g(a_k, x_j)} \tag{6}$$

The attention scores $P(x_i|a_k)$ should be similar to $P'(x_i|a_k)$. Thus, we define the regularization term as $\Omega = \sum_{ij} |P(x_i|a_j) - P'(x_i|a_j)|$ during training.

4.2 Pre-training

The pre-training objective are two-folds: i) learn action embeddings for known predicates in a supervised way, ii) ensure our model can quickly adapt to the actions of new predicates, whose embeddings are initialized by prototype vectors before fine-tuning.

Predicate-dropout Starting with randomly initialized model parameters, we alternately use one batch for the meta-loss $\mathcal{L}_m$ and one batch for optimizing the supervised loss $\mathcal{L}_s$.

In a batch for $\mathcal{L}_m$, we split the data into a meta-support set and a meta-test set. In order to simulate existence of new predicates, we randomly select a subset of predicates as "new", thus their action embeddings $\mathbf{c}$ are replaced by prototype vectors constructed by applying Eq. (4) over the meta-support set. The actions of remaining predicates keep their embeddings learned from previous batches. The resulted action embedding matrix $\mathbf{C}$ is the combination of both.

$$\mathbf{C} = (1 - \mathbf{m}^\intercal)\mathbf{C}_s + \mathbf{m}^\intercal \mathbf{C}_m \tag{7}$$

where $\mathbf{C}_s$ is the embedding matrix learned in a supervised way, and $\mathbf{C}_m$ is constructed by using prototype vectors on the meta-support set. The mask vector $\mathbf{m}$ is generated by setting the indices of actions of the "new" predicates to ones and the other

Algorithm 1: Predicate-Dropout

Input : Training set $\mathcal{D}$, supervisely trained action embedding $\mathcal{C}_s$, number of meta-support examples k, number of meta-test examples n per one support example, predicate-dropout ratio r
Output : The loss $\mathcal{L}_m$.
Extract a template set $\mathcal{T}$ from the training set $\mathcal{D}$
Sample a subset $\mathcal{T}_i$ of size k from $\mathcal{T}$
$S := \emptyset$ # *meta-support set*
$Q := \emptyset$ # *meta-test set*
for t *in* $\mathcal{T}_i$ **do**
  Sample a meta-support example s' with template t from $\mathcal{D}$ without replacement
  Sample a meta-test set Q' of size n with template t from $\mathcal{D}$
  $S = S \cup s'$
  $Q = Q \cup Q'$
end
Build a prototype matrix $\mathcal{C}_m$ on S
Extract a predicate set $\mathcal{P}$ from S
Sample a subset $\mathcal{P}_s$ of size $r \times |\mathcal{P}|$ from $\mathcal{P}$ as new predicates
Build a mask $\mathbf{m}$ using $\mathcal{P}_s$
With $\mathcal{C}_s$, $\mathcal{C}_m$ and $\mathbf{m}$, apply Eq. (7) to compute $\mathbf{C}$
Compute $\mathcal{L}_m$, the cross-entropy on Q with $\mathbf{C}$

to zeros. We refer to this operation as *predicate-dropout*. The training algorithm for the meta-loss is summarised in Algorithm 1.

In a batch for $\mathcal{L}_s$, we update the model parameters and all action embeddings with a cross-entropy loss $\mathcal{L}_s$, together with the attention regularization. Thus, the overall training objective becomes

$$\mathcal{L}_p = \mathcal{L}_m + \mathcal{L}_s + \lambda\Omega \tag{8}$$

Pre-training smoothing Due to the new predicates, the number of candidate actions during the prediction of fine-tuning and testing is larger than the one during pre-training. That leads to distribution discrepancy between pre-training and testing. To minimize the differences, we assume a prior knowledge on the number of actions for new predicates by adding a constant k to the denominator of Eq. (3) when estimating the action probability $P(a_t|\mathbf{h}_t)$ during pre-training.

$$P(a_t|\mathbf{h}_t) = \frac{\exp(\mathbf{c}_{a_t}^\intercal \mathbf{h}_t)}{\sum_{a' \in \mathcal{A}_t} \exp(\mathbf{c}_{a'}^\intercal \mathbf{h}_t) + k} \tag{9}$$

We do not consider this smoothing technique during fine-tuning and testing. Despite its simplicity, the experimental results show a significant performance gain on benchmark datasets.

5 Experiments

Datasets. We use three semantic parsing datasets: JOBS, GEOQUERY, and ATIS. JOBS contains 640 question-LF pairs in Prolog about job listings. GEOQUERY (Zelle and Mooney, 1996) and ATIS (Price, 1990) include 880 and 5,410 utterance-LF pairs in lambda calculas about US geography and flight booking, respectively. The number of predicates in JOBS, GEOQUERY, ATIS is 15, 24, and 88, respectively. All atoms in the datasets are anonymized as in (Dong and Lapata, 2016).

For each dataset, we randomly selected m predicates as the new predicates, which is 3 for JOBS, and 5 for GEOQUERY and ATIS. Then we split each dataset into a train set and an *evaluation* set. And we removed the instances, the template of which is unique in each dataset. The number of such instances is around 100, 150 and 600 in JOBS, GEOQUERY, and ATIS. The ratios between the evaluation set and the train set are 1:4, 2:5, and 1:7 in JOBS, GEOQUERY, and ATIS, respectively. Each LF in an evaluation set contains at least a new predicate, while an LF in a train set contains only known predicates. To evaluate k-shot learning, we build a support set by randomly sampling k pairs per new predicate without replacement from an *evaluation* set, and keep the remaining pairs as the test set. To avoid evaluation bias caused by randomness, we repeat the above process six times to build six different splits of support and test set from each evaluation set. One for hyperparameter tuning and the rest for evaluation. We consider at most 2-shot learning due to the limited number of instances per new predicate in each evaluation set.

Training Details. We pre-train our parser on the training sets for {80, 100} epochs with the Adam optimizer (Kingma and Ba, 2014). The batch size is fixed to 64. The initial learning rate is 0.0025, and the weights are decayed after 20 epochs with decay rate 0.985. The predicate dropout rate is 0.5. The smoothing term is set to {3, 6}. The number of meta-support examples is 30 and the number of meta-test examples per support example is 15. The coefficient of attention regularization is set to 0.01 on JOBS and 1 on the other datasets. We employ the 200-dimensional GLOVE embedding (Pennington et al., 2014) to initialize the word embeddings for utterances. The hidden state size of all LSTM models (Hochreiter and Schmidhuber, 1997) is 256. During fine-tuning, the batch size is 2, the learning rates and the epochs are selected from {0.001, 0.0005} and {20, 30, 40, 60, 120}, respectively.

	JOBS	GEOQUERY	ATIS	JOBS	GEOQUERY	ATIS	p-values
SEQ2SEQ (*pt*)	11.27	20.00	17.23	14.58	33.01	18.76	3.32e-04
SEQ2SEQ (*cb*)	11.70	7.64	2.25	21.49	14.36	7.91	6.65e-06
SEQ2SEQ (*os*)	14.18	11.38	4.45	30.46	33.59	10.17	5.30e-05
COARSE2FINE (*pt*)	10.91	24.07	17.44	13.83	35.63	21.08	1.48e-04
COARSE2FINE (*cb*)	9.28	14.50	0.42	19.61	28.93	9.25	2.35e-06
COARSE2FINE (*os*)	6.73	10.35	5.26	16.08	28.55	17.73	1.13e-05
IRNET (*pt*)	16.00	20.00	17.12	19.06	35.05	20.11	2.86e-05
IRNET (*cb*)	19.67	21.90	5.60	28.22	44.08	15.73	2.76e-03
IRNET (*os*)	14.91	18.78	4.95	30.84	40.97	18.05	2.47e-04
DA	18.91	9.67	4.29	21.31	20.88	17.18	1.13e-06
PT-MAML	11.64	9.76	6.83	17.76	22.52	12.28	1.73e-06
Ours	**27.09**	**27.49**	**19.27**	**32.5**	**48.45**	**22.48**	

Table 2: Evaluation of learning results on three datasets. (Left) The one-shot results. (Right) The two-shot results.

Baselines. We compared our methods with five competitive baselines, SEQ2SEQ with attention (Luong et al., 2015), COARSE2FINE (Dong and Lapata, 2018), IRNET (Guo et al., 2019), PT-MAML (Huang et al., 2018) and DA (Li et al., 2020). COARSE2FINE is the best performing supervised model on the standard split of GEOQUERY and ATIS datasets. PT-MAML is a few-shot learning semantic parser that adopts Model-Agnostic Meta-Learning (Finn et al., 2017). We adapt PT-MAML in our scenario by considering a group of instances that share the same template as a pseudo-task. DA is the most recently proposed neural semantic parser applying domain adaptation techniques. IRNET is the strongest semantic parser that can generalize to unseen database schemas. In our case, we consider a list of predicates in support sets as the columns of a new database schema and incorporate the schema encoding module of IRNET into the encoder of our base parser. We choose IRNET over RATSQL (Wang et al., 2019) because IRNET achieves superior performance on our datasets.

We consider three different supervised learning settings. First, we pre-train a model on a train set, followed by fine-tuning it on the corresponding support set, coined *pt*. Second, a model is trained on the combination of a train set and a support set, coined *cb*. Third, the support set in *cb* is over-sampled by 10 times and 5 times for one-shot and two-shot respectively, coined *os*.

Evaluation Details. The same as prior work (Dong and Lapata, 2018; Li et al., 2020), we report accuracy of exactly matched LFs as the main evaluation metric.

To investigate if the results are statistically significant, we conducted the Wilcoxon signed-rank test, which assesses whether our model consistently performs better than another baseline across *all* evaluation sets. It is considered superior than t-test in our case, because it supports comparison across different support sets and does not assume normality in data (Demšar, 2006). We include the corresponding p-values in our result tables.

5.1 Results and Discussion

Table 2 shows the average accuracies and significance test results of all parsers compared on all three datasets. Overall, `ProtoParser` outperforms all baselines with at least 2% on average in terms of accuracy in both one-shot and two-shot settings. The results are statistically significant w.r.t. the strongest baselines, IRNET (*cb*) and COARSE2FINE (*pt*). The corresponding p-values are 0.00276 and 0.000148, respectively. Given one-shot example on JOBS, our parser achieves 7% higher accuracy than the best baseline, and the gap is 4% on GEOQUERY with two-shots examples. In addition, none of the SOTA baseline parsers can consistently outperform other SOTA parsers when there are few parallel data for new predicates. In one-shot setting, the best supervised baseline IRNET (*cb*) can achieve the best results on GEOQUERY and JOBS among all baselines, and on two-shot setting, it performs best only on GEOQUERY. It is also difficult to achieve good performance by adapting the existing meta-learning or transfer learning algorithms to our problem, as evident by the moderate performance of PT-MAML and DA on all datasets.

The problems of few-shot learning demonstrate the challenges imposed by infrequent predicates. There are significant proportions of infrequent predicates on the existing datasets. For example, on GEOQUERY, there are 10 predicates contributing to only 4% of the total frequency of all 24 predicates, while the top two frequent predicates amount

	JOBS	GEOQUERY	ATIS	JOBS	GEOQUERY	ATIS	p-values
Ours	27.09	**27.49**	**19.27**	**32.50**	**48.45**	**22.48**	
- sup	23.63	18.86	12.91	26.91	39.51	14.89	1.44e-05
- proto	22.91	18.77	13.24	29.16	38.93	16.81	1.77e-05
- reg	**29.27**	18.10	13.66	31.03	39.61	18.58	9.60e-04
- strsim	22.18	19.62	10.14	28.41	47.09	19.98	9.27e-04
- cond	23.27	19.05	9.63	27.66	40.97	17.50	4.37e-05
- smooth	24.36	23.60	15.23	30.84	44.95	18.71	3.27e-03

Table 3: Ablation study results. (Left) The one-shot learning results. (Right) The two-shot learning results.

to 42%. As a result, the SOTA parsers achieve merely less than 25% and 44% of accuracy with one-shot and two-shots examples, respectively. In contrast, those parsers achieve more than 84% accuracy on the standard splits of the same datasets in the supervised setting.

Infrequent predicates in semantic parsing can also be viewed as a class imbalance problem, when support sets and train sets are combined in a certain manner. In this work, the ratio between the support set and the train set in JOBS, GEOQUERY, and ATIS is 1:130, 1:100, and 1:1000, respectively. Different models prefer different ways of using the train sets and support sets. The best option for COARSE2FINE and SEQ2SEQ is to pre-train on a train set followed by fine-tuning on the corresponding support set, while IRNET favors oversampling in two-shot setting.

Ablation Study We examine the effect of different components of our parser by removing each of them individually and reporting the corresponding average accuracy. As shown in Table 3, removing any of the components almost always leads to statistically significant drop of performance. The corresponding p-values are all less than 0.00327.

To investigate predicate-dropout, we exclude either supervised-loss during pre-training (-sup) or initialization of new predicate embeddings by prototype vectors before fine-tuning (-proto). It is clear from Table 3 that ablating either supervisely trained action embeddings or prototype vectors hurts performance severely.

We further study the efficacy of attention regularization by removing it completely (-reg), removing only the string similarity feature (-strsim), or conditional probability feature (-cond). Removing the regularization completely degrades performance sharply except on JOBS in the one-shot setting. Our further inspection shows that model learning is easier on JOBS than on the other two datasets. Each predicate in JOBS almost always aligns to the same word across examples, while a predicate can align with different word/phrase in different examples in GEOQUERY and ATIS. The performance drop with -strsim and -cond indicates that we cannot only reply on a single statistical measure for regularization. For instance, we cannot always find predicates take the same string form as the corresponding words in input utterances. In fact, the proportion of predicates present in input utterances is only 42%, 38% and 44% on JOBS, ATIS, and GEOQUERY, respectively.

Furthermore, without pre-training smoothing (-smooth), the accuracy drops at least 1.6% in terms of mean accuracy on all datasets. Smoothing enables better model parameter training by more accurate modelling in pre-training.

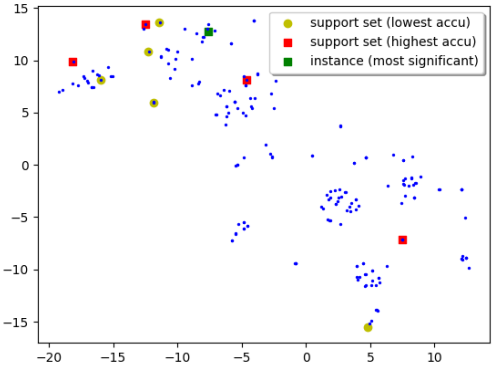

Figure 1: (Round) The support set with the lowest accuracy. (Box) The support set with the highest accuracy.

Support Set Analysis We observe that all models consistently achieve high accuracy on certain support sets of the same dataset, while obtaining low accuracies on the other ones. We illustrate the reasons of such effects by plotting the evaluation set of GEOQUERY. Each data point in Figure 1 depicts an representation, which is generated by the encoder of our parser after pre-training. We applied T-SNE (Maaten and Hinton, 2008) for dimension reduction. We highlight two support sets used in the one-shot setting on GEOQUERY. All exam-

ples in the highest performing support set tend to scatter evenly and cover different dense regions in the feature space, while the examples in the lowest performing support set are far from a significant number of dense regions. Thus, the examples in good support sets are more representative of the underlying distribution than the ones in poor support sets. When we leave out each example in the highest performing support set and re-evaluate our parser each time, we observe that the good ones (e.g. the green box in Figure 1) locate either in or close to some of the dense regions.

6 Conclusion and Future Work

We propose a novel few-shot learning based semantic parser, coined `ProtoParser`, to cope with new predicates in LFs. To address the challenges in few-shot learning, we propose to train the parser with a pre-training procedure involving predicate-dropout, attention regularization, and pre-training smoothing. The resulted model achieves superior results over competitive baselines on three benchmark datasets.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186.

Yujia Bao, Shiyu Chang, Mo Yu, and Regina Barzilay. 2018. Deriving machine attention from human rationales. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1903–1913.

Qingqing Cai and Alexander Yates. 2013. Semantic parsing freebase: Towards open-domain semantic parsing. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 328–338.

Shuaichen Chang, Pengfei Liu, Yun Tang, Jing Huang, Xiaodong He, and Bowen Zhou. 2019. Zero-shot text-to-sql learning with auxiliary task. *arXiv preprint arXiv:1908.11052*.

Bo Chen, Le Sun, and Xianpei Han. 2018. Sequence-to-action: End-to-end semantic graph generation for semantic parsing. *arXiv preprint arXiv:1809.00773*.

Jianpeng Cheng, Siva Reddy, Vijay Saraswat, and Mirella Lapata. 2019. Learning an executable neural semantic parser. *Computational Linguistics*, 45(1):59–94.

Javid Dadashkarimi, Alexander Fabbri, Sekhar Tatikonda, and Dragomir R Radev. 2018. Zero-shot transfer learning for semantic parsing. *arXiv preprint arXiv:1808.09889*.

Janez Demšar. 2006. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. *arXiv preprint arXiv:1601.01280*.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. *arXiv preprint arXiv:1805.04793*.

Long Duong, Hadi Afshar, Dominique Estival, Glen Pink, Philip Cohen, and Mark Johnson. 2018. Active learning for deep semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 43–48.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based dependency parsing with stack long short-term memory. *arXiv preprint arXiv:1505.08075*.

Catherine Finegan-Dollak, Jonathan K Kummerfeld, Li Zhang, Karthik Ramanathan, Sesh Sadasivam, Rui Zhang, and Dragomir Radev. 2018. Improving text-to-sql evaluation methodology. *arXiv preprint arXiv:1806.09029*.

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*, pages 1126–1135. JMLR. org.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 1999. Learning to forget: Continual prediction with lstm.

Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards complex text-to-sql in cross-domain database with intermediate representation. *arXiv preprint arXiv:1905.08205*.

Jonathan Herzig and Jonathan Berant. 2018. Decoupling structure and lexicon for zero-shot semantic parsing. *arXiv preprint arXiv:1804.07918*.

Jonathan Herzig and Jonathan Berant. 2019. Don't paraphrase, detect! rapid and effective data collection for semantic parsing. *arXiv preprint arXiv:1908.09940*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Zikun Hu, Xiang Li, Cunchao Tu, Zhiyuan Liu, and Maosong Sun. 2018. Few-shot charge prediction with discriminative legal attributes. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 487–498.

Po-Sen Huang, Chenglong Wang, Rishabh Singh, Wentau Yih, and Xiaodong He. 2018. Natural language to structured query generation via meta-learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 732–738.

Srinivasan Iyer, Alvin Cheung, and Luke Zettlemoyer. 2019. Learning programmatic idioms for scalable semantic parsing. *arXiv preprint arXiv:1904.09086*.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. *arXiv preprint arXiv:1606.03622*.

Aishwarya Kamath and Rajarshi Das. 2018. A survey on semantic parsing. *CoRR*, abs/1812.00978.

Hidetaka Kamigaito, Katsuhiko Hayashi, Tsutomu Hirao, Masaaki Nagata, Hiroya Takamura, and Manabu Okumura. 2017. Supervised attention for sequence-to-sequence constituency parsing. *IJCNLP 2017*, page 7.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Tomáš Kočiskỳ, Gábor Melis, Edward Grefenstette, Chris Dyer, Wang Ling, Phil Blunsom, and Karl Moritz Hermann. 2016. Semantic parsing with semi-supervised sequential autoencoders. *arXiv preprint arXiv:1609.09315*.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1526.

Dongjun Lee. 2019. Clause-wise and recursive decoding for complex and cross-domain text-to-sql generation. 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 6047–6053.

Dongjun Lee, Jaesik Yoon, Jongyun Song, Sanggil Lee, and Sungroh Yoon. 2019. One-shot learning for text-to-sql generation. *arXiv preprint arXiv:1905.11499*.

Yoonho Lee and Seungjin Choi. 2018. Gradient-based meta-learning with learned layerwise metric and subspace. *arXiv preprint arXiv:1801.05558*.

Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710.

Zechang Li, Yuxuan Lai, Yansong Feng, and Dongyan Zhao. 2020. Domain adaptation for semantic parsing. *arXiv preprint arXiv:2006.13071*.

Kevin Lin, Ben Bogin, Mark Neumann, Jonathan Berant, and Matt Gardner. 2019. Grammar-based neural text-to-sql generation. *arXiv preprint arXiv:1905.13326*.

Lemao Liu, Masao Utiyama, Andrew Finch, and Eiichiro Sumita. 2016. Neural machine translation with supervised attention. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3093–3102.

Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. 2017. Exploiting argument information to improve event detection via supervised attention mechanisms. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1789–1798.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605.

Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to information retrieval*. Cambridge university press.

Richard Montague. 1973. The proper treatment of quantification in ordinary english. In *Approaches to natural language*, pages 221–242. Springer.

Ansong Ni, Pengcheng Yin, and Graham Neubig. 2019. Merging weak and active supervision for semantic parsing. *arXiv preprint arXiv:1911.12986*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Patti 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*.

Maxim Rabinovich, Mitchell Stern, and Dan Klein. 2017. Abstract syntax networks for code generation

and semantic parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1139–1149.

Scott Reed, Yutian Chen, Thomas Paine, Aäron van den Oord, SM Eslami, Danilo Rezende, Oriol Vinyals, and Nando de Freitas. 2017. Few-shot autoregressive density estimation: Towards learning to learn distributions. *arXiv preprint arXiv:1710.10304*.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems*, pages 4077–4087.

Yu Su and Xifeng Yan. 2017. Cross-domain semantic parsing via paraphrasing. *arXiv preprint arXiv:1704.05974*.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

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 preprint arXiv:1909.05438*.

I Sutskever, O Vinyals, and QV Le. 2014. Sequence to sequence learning with neural networks. *Advances in NIPS*.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. 2016. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2019. Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers. *arXiv preprint arXiv:1911.04942*.

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)*, volume 1, pages 1332–1342.

Frank Wilcoxon. 1992. Individual comparisons by ranking methods. In *Breakthroughs in statistics*, pages 196–202. Springer.

Pengcheng Yin and Graham Neubig. 2018. Tranx: A transition-based neural abstract syntax parser for semantic parsing and code generation. *arXiv preprint arXiv:1810.02720*.

Pengcheng Yin, Chunting Zhou, Junxian He, and Graham Neubig. 2018. StructVAE: Tree-structured latent variable models for semi-supervised semantic parsing. In *The 56th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887*.

John M Zelle and Raymond J Mooney. 1996. Learning to parse database queries using inductive logic programming. In *Proceedings of the national conference on artificial intelligence*, pages 1050–1055.

Rui Zhang, Tao Yu, He Yang Er, Sungrok Shim, Eric Xue, Xi Victoria Lin, Tianze Shi, Caiming Xiong, Richard Socher, and Dragomir Radev. 2019a. Editing-based sql query generation for cross-domain context-dependent questions. *arXiv preprint arXiv:1909.00786*.

Sheng Zhang, Xutai Ma, Kevin Duh, and Benjamin Van Durme. 2019b. Broad-coverage semantic parsing as transduction. *arXiv preprint arXiv:1909.02607*.

Q. Zhu, X. Ma, and X. Li. 2019. Statistical learning for semantic parsing: A survey. *Big Data Mining and Analytics*, 2(4):217–239.