Neural Semantic Parsing in Low-Resource Settings with Back-Translation and Meta-Learning

Yibo Sun,[*,1] **Duyu Tang,**[2] **Nan Duan,**[2] **Yeyun Gong,**[2] **Xiaocheng Feng,**[1] **Bing Qin,**[1] **Daxin Jiang**[3]
[1]Harbin Institute of Technology, Harbin, China
[2]Microsoft Research Asia, Beijing, China
[3]Microsoft Search Technology Center Asia, Beijing, China
{ybsun, xcfeng, qinb}@ir.hit.edu.cn {dutang, nanduan, yegong, djiang}@microsoft.com

Abstract

Neural semantic parsing has achieved impressive results in recent years, yet its success relies on the availability of large amounts of supervised data. Our goal is to learn a neural semantic parser when only prior knowledge about a limited number of simple rules is available, without access to either annotated programs or execution results. Our approach is initialized by rules, and improved in a back-translation paradigm using generated question-program pairs from the semantic parser and the question generator. A phrase table with frequent mapping patterns is automatically derived, also updated as training progresses, to measure the quality of generated instances. We train the model with model-agnostic meta-learning to guarantee the accuracy and stability on examples covered by rules, and meanwhile acquire the versatility to generalize well on examples uncovered by rules. Results on three benchmark datasets with different domains and programs show that our approach incrementally improves the accuracy. On WikiSQL, our best model is comparable to the state-of-the-art system learned from denotations.

Introduction

Semantic parsing aims to map natural language questions to the logical forms of their underlying meanings, which can be regarded as programs and executed to yield answers, aka denotations (Berant et al. 2013). In the past few years, neural network based semantic parsers have achieved promising performances (Liang et al. 2017), however, their success is limited to the setting with rich supervision, which is costly to obtain. There have been recent attempts at low-resource semantic parsing, including data augmentation methods which are learned from a small number of annotated examples (Guo et al. 2018), and methods for adapting to unseen domains while only being trained on annotated examples in other domains.

This work investigates neural semantic parsing in a low-resource setting, in which case we only have our prior knowledge about a limited number of simple mapping rules, including a small amount of domain-independent word-level matching tables if necessary, but have no access to either annotated programs or execution results. Our key idea is to use these rules to collect modest question-programs pairs as the starting point, and then leverage automatically generated examples to improve the accuracy and generality of the model. This presents three challenges including how to generate examples in an efficient way, how to measure the quality of generated examples which might contain errors and noise, and how to train a semantic parser that makes robust predictions for examples covered by rules and generalizes well to uncovered examples.

We address the aforementioned challenges with a framework consisting of three key components. The **first** component is a data generator. It includes a neural semantic parsing model, which maps a natural language question to a program, and a neural question generation model, which maps a program to a natural language question. We learn these two models in a back-translation paradigm using pseudo parallel examples, inspired by its big success on unsupervised neural machine translation (Sennrich, Haddow, and Birch 2016; Lample et al. 2018). The **second** component is a quality controller, which is used for filtering out noise and errors contained in the pseudo data. We construct a phrase table with frequent mapping patterns, therefore noise and errors with low frequency can be filtered out. A similar idea has been worked as posterior regularization in neural machine translation (Zhang et al. 2017; Ren et al. 2019). The **third** component is a meta learner. Instead of transferring a model pretrained with examples covered by rules to the generated examples, we leverage model-agnostic meta-learning (Finn, Abbeel, and Levine 2017), an elegant meta-learning algorithm which has been successfully applied to a wide range of tasks including few-shot learning and adaptive control. We regard different data sources as different tasks, and use outputs of the quality controller for stable training.

We test our approach on three tasks with different programs, including SQL (and SQL-like) queries for both single-turn and multi-turn questions over web tables (Zhong, Xiong, and Socher 2017; Iyyer, Yih, and Chang 2017), and subject-predicate pairs over a large-scale knowledge graph (Bordes et al. 2015). The program for SQL queries for single-turn questions and subject-predicate pairs over

[*]This work was done during an internship at MSRA.

Algorithm 1 Low-Resource Neural Semantic Parsing with Back-Translation and MAML

```
Require: $Q$: a set of natural language questions. $LF$: a collection of sampled logical forms.
Require: $r$: a rule, if satisfied, maps $q$ to $lf$. $\alpha$, $\beta$: step size hyperparameters.
 1: Apply $r$ to $Q$, obtain training data $D_0$
 2: Use $D_0$ to initialize $\theta_{q\to lf}$ and $\theta_{lf\to q}$
 3: while not done do
 4:     Apply $f_{q\to lf}$ to $Q$, apply $f_{lf\to q}$ to $LF$
 5:     Use $f_{q\to lf}(Q)$ and $f_{lf\to q}(LF)$ to update the phrase table of the quality controller $qc$
 6:     Update $\theta_{lf\to q}$ using $D_0$ and $qc(f_{q\to lf}(Q))$
 7:     for Task $\mathcal{T}_i \in \{r=T, r=F\}$ do
 8:         Sample $\mathcal{D}_i$ from $\{D_0, qc(f_{q\to lf}(Q)), qc(f_{lf\to q}(LF))\}$ for task $\mathcal{T}_i$
 9:         Compute gradients using $\mathcal{D}_i$ and $\mathcal{L}$, update learner $\theta_i = \theta - \alpha\nabla_\theta\mathcal{L}(f_\theta)$
10:         Sample $\mathcal{D}'_i$ from $\{D_0, qc(f_{q\to lf}(Q)), qc(f_{lf\to q}(LF))\}$ for the meta-update
11:         Compute gradients using $\mathcal{D}'_i$ and $\mathcal{L}$, update meta-learner $\theta = \theta - \beta\nabla_\theta\mathcal{L}(f_{\theta_i})$
12:     end for
13:     Update $\theta^{r=T}_{q\to lf}$ and $\theta^{r=F}_{q\to lf}$ with $\theta$ as initialization, respectively
14: end while
```

knowledge graph is simple while the program for SQL queries for multi-turn questions have top-tier complexity among currently proposed tasks. Results show that our approach yields large improvements over rule-based systems, and incorporating different strategies incrementally improves the overall performance. On WikiSQL, our best performing system achieves execution accuracy of 72.7%, comparable to a strong system learned from denotations (Agarwal et al. 2019) with an accuracy of 74.8%.

Problem Statement

We focus on the task of executive semantic parsing. The goal is to map a natural language question/utterance q to a logical form/program lf, which can be executed over a world Wld to obtain the correct answer a.

We consider three tasks. The **first** task is single-turn table-based semantic parsing, in which case q is a self-contained question, lf is a SQL query in the form of *"SELECT agg col$_1$ WHERE col$_2$ = val$_2$ AND ..."*, and Wld is a web table consisting of multiple rows and columns. We use WikiSQL (Zhong, Xiong, and Socher 2017) as the testbed for this task. The **second** task is multi-turn table-based semantic parsing. Compared to the first task, q could be a follow-up question, the meaning of which depends on the conversation history. Accordingly, lf in this task supports additional operations that copy previous turn lf to the current turn. We use SequentialQA (Iyyer, Yih, and Chang 2017) for evaluation. In the **third** task, we change Wld to a large-scale knowledge-graph (i.e. Freebase) and consider knowledge-based question answering for single-turn questions. We use SimpleQuestions (Bordes et al. 2015) as the testbed, where the lf is in the form of a simple λ-calculus like $\lambda x.predicate(subject, x)$, and the generation of lf is equivalent to the prediction of the predicate and the subject entity.

We study the problem in a low-resource setting. In the training process, we don't have annotated logical forms lf or execution results a. Instead, we have a collection of natural language questions for the task, a limited number of simple mapping rules based on our prior knowledge about the task, and may also have a small amount of domain-independent word-level matching tables if necessary. These rules are not perfect, with low coverage, and can even be incorrect for some situations. For instance, when predicting a SQL command in the first task, we have a prior knowledge that (1) WHERE values potentially have co-occurring words with table cells; (2) the words "*more*" and "*greater*" tend to be mapped to WHERE operator ">"; (3) within a WHERE clause, header and cell should be in the same column; and (4) the word "*average*" tends to be mapped to aggregator "*avg*". Similarly, when predicting a λ-calculus in the third task, the entity name might be present in the question, and among all the predicates connected to the entity, the predicate with maximum number of co-occurred words might be correct. We would like to study to what extent our model can achieve if we use rules as the starting point.

Learning Algorithm

We describe our approach for low-resource neural semantic parsing in this section.

We propose to train a neural semantic parser using back-translation and meta-learning. The learning process is summarized in Algorithm 1. We describe the three components in this section, namely back-translation, quality control, and meta-learning.

Back-Translation

Following the back-translation paradigm (Sennrich, Haddow, and Birch 2016; Lample et al. 2018), we have a semantic parser, which maps a natural language question q to a logical form lf, and a question generator, which maps lf to q. The semantic parser works for the primary task, and the question generator mainly works for generating pseudo datapoints. We start the training process by applying the rule r to a set of natural language questions Q. The resulting dataset is considered as the training data to initialize both the semantic parser and the question generator. Afterwards, both

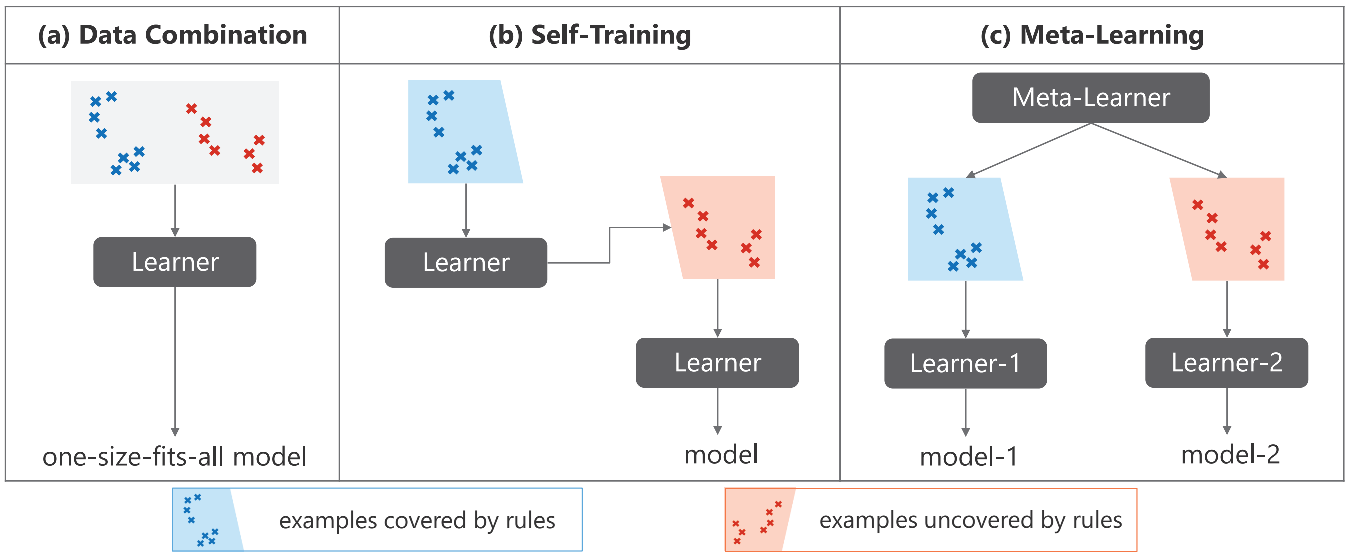


Figure 1: An illustration of the difference between (a) data combination which learns a monolithic, one-size-fits-all model, (b) self-training which learns from predictions which the model produce and (c) meta-learning that reuse the acquired ability to learn.

models are improved following the back-translation protocol that target sequences should follow the real data distribution, yet source sequences can be generated with noises. This is based on the consideration that in an encoder-decoder model, the decoder is more sensitive to the data distribution than the encoder. We use datapoints from both models to train the semantic parser because a logical form is structural which follows a grammar, whose distribution is similar to the ground truth.

Quality Controller

Directly using generated datapoints as supervised training data is not desirable because those generated datapoints contain noises or errors. To address this, we follow the application of posterior regularization in neural machine translation (Zhang et al. 2017), and implement a dictionary-based discriminator which is used to measure the quality of a pseudo data. The basic idea is that although these generated datapoints are not perfect, the frequent patterns of the mapping between a phrase in q to a token in lf are helpful in filtering out noise in the generated data with low frequency (Ren et al. 2019). There are multiple ways to collect the phrase table information, such as using statistical phrase-level alignment algorithms like Giza++ or directly counting the co-occurrence of any question word and logical form token. We use the latter one in this work.

Meta-Learning

A simple way to update the semantic parser is to merge the datapoints in hand and train a one-size-fits-all model (Guo et al. 2018). However, this will hurt model's stability on examples covered by rules, and examples of the same task may vary widely (Huang et al. 2018). Dealing with different types of examples requires the model to possess different abilities. For example, tackling examples uncovered by rules in WikiSQL requires the model to have the additional ability to map a column name to a totally different utterance, such as "*country*" to "*nation*". Another simple solution is self-training (McClosky, Charniak, and Johnson 2006). One can train a model with examples covered by rules, and use the model as a teacher to make predictions on examples uncovered by rules and update the model on these predictions. However, self-training is somewhat tautological because the model is learned to make predictions which it already can produce.

We learn the semantic parser with meta-learning, regarding learning from examples covered by rules or uncovered by rules as two (pseudo) tasks. Compared to the aforementioned strategies, the advantage of exploring meta-learning here is two-fold. First, we learn a specific model for each task, which provides guarantees about its stability on examples covered by rules. In the **test phase**, we can use the rule to detect which task an example belongs to, and use the corresponding task-specific model to make predictions. When dealing with examples covered by rules, we can either directly use rules to make predictions or use the updated model, depending on the accuracy of the learned model on the examples covered by rules on development set. Second, latent patterns of examples may vary widely in terms of whether or not they are covered by rules. Meta-learning is more desirable in this situation because it learns the model's ability to learn, improving model's versatility rather than mapping the latent patterns learned from datapoints in one distribution to datapoints in another distribution by force. Figure 1 is an illustration of data combination, self-training, and meta-learning.

Meta-learning includes two optimizations: the learner that learns new tasks, and the meta-learner that trains the

Methods	Supervision	Rule Covered	Rule Uncovered	Overall
Dong and Lapata (2018)	logical form	-	-	78.5%
Shi et al. (2018)	logical form	-	-	87.1%
Liang et al. (2018)	denotation	-	-	72.4%
Agarwal et al. (2019)	denotation	-	-	74.8%
Base (full supervision)	logical form	85.0%	71.7%	82.3%
Rule	rule	77.6%	0.0%	61.8%
Base (trained on rule-covered data)	rule	75.9%	48.4%	70.3%
Base + Self Training	rule	75.6%	49.3%	70.3%
Base + Question Generation	rule	77.7%	51.9%	72.5%
Base + BT	rule	77.6%	51.6%	72.3%
Base + BT + QC	rule	77.8%	52.1%	72.6%
Base + BT + QC + MAML	rule	77.9%	52.1%	72.7%

Table 1: Results on WikiSQL testset. BT stands for back-translation. QC stands for quality control.

learner. In this work, the meta-learner is optimized by finding a good initialization that is highly adaptable. Specifically, we use model-agnostic meta-learning, MAML (Finn, Abbeel, and Levine 2017), a powerful meta-learning algorithm with desirable properties including introducing no additional parameters and making no assumptions of the form of the model. In MAML, task-specific parameter θ_i is initialized by θ, and updated using gradient decent based on the loss function $\mathcal{L}_i$ of task $\mathcal{T}_i$. In this work, the loss functions of two tasks are the same. The updated parameter θ_i is then used to calculate the model's performance across tasks to update the parameter θ. In this work, following the practical suggestions given by Antoniou, Edwards, and Storkey (2019), we update θ in the *inner-loop* and regard the outputs of the quality controller as the input of both tasks.

If we only have examples covered by rules, such as those used in the initialization phase, meta-learning learns to learn a good initial parameter that is evaluated by its usefulness on the examples from the same distribution. In the training phase, datapoints from both tasks are generated, and meta-learning learns to learn an initialization parameter which can be quickly and efficiently adapted to examples from both tasks.

Experiment

We conduct experiments on three tasks to test our approach, including generating SQL (or SQL-like) queries for both single-turn and multi-turn questions over web tables (Zhong, Xiong, and Socher 2017; Iyyer, Yih, and Chang 2017), and predicting subject-predicate pairs over a knowledge graph (Bordes et al. 2015). We describe task definition, base models, experiments settings and empirical results for each task, respectively.

Table-Based Semantic Parsing

Task and Dataset Given a natural language q and a table t with n columns and m rows as the input, the task is to output a SQL query y, which could be executed on table t to yield the correct answer of q. We conduct experiments on WikiSQL (Zhong, Xiong, and Socher 2017), which provides 87,726 annotated question-SQL pairs over 26,375 web tables. In this work, we do not use either SQL queries or answers in the training process. We use execution accuracy as the evaluation metric, which measures the percentage of generated SQL queries that result in the correct answer.

SQL Token	NL Word
sum	sum
$count$	how many, total number
max	maximum
min	minimum
avg	average
$>$	more, greater, higher, taller, longer, older, larger, after
$<$	less, smaller, lower, fewer, nearer, shorter, before

Table 2: Token-level dictionary for aggregators (upper group) and operators (lower group) in WikiSQL.

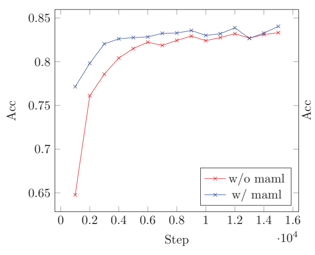
Rules We describe our rules for WikiSQL here. We first detect WHERE values, which exactly match to table cells. After that, if a cell appears at more than one column, we choose the column name with more overlapped words with the question, with a constraint that the number of co-occurred words is larger than 1. By default, a WHERE operator is $=$, except for the case that surrounding words of a value contain keywords for $>$ and $<$. Then, we deal with the SELECT column, which has the largest number of co-occurred words and cannot be same with any WHERE column. By default, the SELECT AGG is NONE, except for matching to any keywords in Table 2. The coverage of our rule on training set is 78.4%, with execution accuracy of 77.9%.

Base Model We implement a neural network modular approach as the base model, which includes different modules to predict different SQL constituents. This approach is based on the understanding of the SQL grammar in WikiSQL, namely "*SELECT \$agg \$column WHERE \$column \$op \$value (AND \$column \$op \$value)**", where tokens starting with "\$" are the slots to be predicted (Xu, Liu,

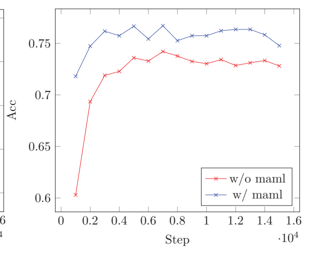
Methods	Supervision	Rule Covered	Rule Uncovered	Overall
Petrochuk and Zettlemoyer (2018)	logical form	-	-	78.1%
Base (full supervision)	logical form	81.8%	67.6%	72.4%
Rule	rule	76.5%	0.0%	25.7%
Base (trained on rule-covered data)	rule	74.2%	34.8%	48.0%
Base + Self Training	rule	74.4%	46.5%	55.9%
Base + Question Generation	rule	73.9%	42.1%	52.8%
Base + BT	rule	75.5%	47.7%	57.1%
Base + BT + QC	rule	75.3%	48.6%	57.6%
Base + BT + QC + MAML	rule	76.8%	48.6%	58.1%

Table 3: Results on SimpleQuestions testset. BT stands for back-translation. QC stands for quality control.

and Song 2017). In practice, modular approaches typically achieve higher accuracy than end-to-end learning approach. Specifically, at the first step we implement a sequential labeling module to detect WHERE values and link them to table cells. Advantages of starting from WHERE values include that WHERE values are less ambiguous compared to other slots, and that the number of WHERE clauses can be naturally detected. After that, for each WHERE value, we use the preceding and following contexts in the question to predict its WHERE column and the WHERE operator through two unidirectional LSTM. Column attention (Xu, Liu, and Song 2017) is used for predicting a particular column. Similar LSTM-based classifiers are used to predict SELECT column and SELECT aggregator.



(a) Rule-covered



(b) Rule-uncovered

Figure 2: Learning curve of the WHERE column prediction model on WikiSQL devset.

Settings According to whether the training data can be processed by our rules, we divide it into two parts: rule covered part and rule uncovered part. For the rule covered part we could get rule covered training data using our rules. For the rule uncovered part we could also get training data using the trained Base model we have, we refer to these data as self-inference training data. Furthermore, we could get more training data by back translation, we refer to these data as question-generation training data. For all the settings, the Base Model is initialized with rule covered training data. In Base + Self Training Method, we finetune the Base model with self-inference training data. In Base + Question Generation Method, we use question-generation training data to finetune our model. In Base + BT Method, we use both self-inference and question-generation data to finetune our model. In Base + BT + QC, we add our quality controller. In Base + BT + QC + MAML, we further add meta-learning.

Results and Analysis Results are given in Table 1. We can see that back-translation, quality control and MAML incrementally improves the accuracy. Question generation is better than self-training here because the logical form in WikiSQL is relatively simple, so the distribution of the sampled logical forms is similar to the original one. In the back-translation setting, generated examples come from both self-training and the question generation model. The model performs better than rules on rule-covered examples, and improves the accuracy on uncovered examples. Figure 2 shows the learning curves of the COLUMN prediction model with or without using MAML. The model using MAML has a better starting point during training, which reflects the effectiveness of the pre-trained parameter.

Knowledge-Based Question Answering

We test our approach on question answering over another genre of environment: knowledge graph consisting of subject-relation-object triples.

Task and Dataset Given a natural language question and a knowledge graph, the task aims to correctly answer the question with evidences from the knowledge graph. We do our study on SimpleQuestions (Bordes et al. 2015), which includes 108,442 simple questions, each of which is accompanied by a subject-relation-object triple. Questions are constructed in a way that subject and relation are mentioned in the question, and that object is the answer. The task requires predicting the entityId and the relation involved in the question.

Rulse Our rule for KBQA is simple without using a curated mapping dictionary. First, we detect an entity from the question using strict string matching, with the constraint that only one entity from the KB has the same surface string and that the question contains only one entity. After that, we get the connected relations of the detected entity, and assign the relation as the one with maximum number of co-occurred words. The coverage of our rule on training set is 16.0%, with an accuracy of 97.3% for relation prediction.

Methods	Supervision	Rule Covered	Rule Uncovered	Overall
Pasupat and Liang (2015)	denotation	-	-	33.2%
Neelakantan et al. (2016)	denotation	-	-	40.2%
Iyyer, Yih, and Chang (2017)	denotation	-	-	44.7%
Misra et al. (2018)	denotation	-	-	49.7%
Base (full supervision)	denotation	50.6%	33.5%	45.9%
Rule	rule	51.2%	0.0%	37.2%
Base (trained on rule-covered data)	rule	43.7%	9.3%	34.3%
Base + Self Training	rule	42.3%	10.7%	33.7%
Base + Question Generation	rule	34.7%	9.1%	27.7%
Base + BT	rule	39.2%	10.1%	31.3%
Base + BT + QC	rule	41.0%	10.1%	32.6%
Base + BT + QC + MAML	rule	41.2%	9.7%	32.7%
Base + BT (w/o QG) + MAML	rule	43.7%	11.3%	34.6%

Table 4: Results on SequentialQA testset. BT stands for back-translation. QC stands for quality control.

Base Model We follow Petrochuk and Zettlemoyer (2018), and implement a KBQA pipeline consisting of three modules in this work. At the first step, we use a sequence labeling model, i.e. LSTM-CRF, to detect entity mention words in the question. After that, we use an entity linking model with BM25 built on Elasticsearch. Top-K ranked similar entities are retrieved as candidate list. Then, we get all the relations connected to entities in the candidate list as candidate relations, and use a relation prediction model, which is based on Match-LSTM (Wang and Jiang 2016), to predict the relation. Finally, from all the entities connected to the predicted relation, we choose the one with highest BM25 score as the predicted entity. We use FB2M as the KB, which includes about 2 million triples.

Settings The settings are the same as those described in table-based semantic parsing.

SQL Token	NL Word
$\geq$	no less than, greater or equal, at least
$\leq$	no more than, less or equal, at most
$followup$	of those, which ones, which one
NEG	do not, does not, did not, have not, has not, had not, was not, were not, is not, are not, not have, not has

Table 5: Token-level dictionary used for additional actions in SequentialQA.

Results and Analysis Results are given in Table 3, which are consistent with the numbers in WikiSQL. Using back-translation, quality control and MAML incrementally improves the accuracy, and our approach generalizes well to rule-uncovered examples.

Conversational Table-Based Semantic Parsing

We consider the task of conversational table-based semantic parsing in this part. Compared to single-turn table-based semantic parsing as described in subsection , the meaning of a natural language may also depends on questions of past turns, which is the common ellipsis and co-reference phenomena in conversational agents.

Task and Dataset Given a natural language question at the current turn, a web table, and previous turn questions in a conversation as the input, the task aims to generate a program (i.e. logical form), which can be executed on the table to obtain the correct answer of the current turn question.

We conduct experiments on SequentialQA (Iyyer, Yih, and Chang 2017) which is derived from the WikiTableQuestions dataset(Pasupat and Liang 2015). It contains 6,066 question sequences covering 17,553 question-answer pairs. Each sequence includes 2.9 natural language questions on average. Different from WikiSQL which provides the correct logical form for each question, SequentialQA only annotates the correct answer. This dataset is also harder than the previous two, since it requires complex, highly compositional logical forms to get the answer. Existing approaches are evaluated by question answering accuracy, which measures whether the predicted answer is correct or not.

Rules The pipeline of rules in SequentialQA is similar to that of WikiSQL. Compared to the grammar of WikiSQL, the grammar of SequentialQA has additional actions including copying the previous-turn logical form, no greater than, no more than, and negation. Table 5 shows the additional word-level mapping table used in SequentialQA. The coverage of our rule on training set is 75.5%, with an accuracy of 38.5%.

Base Model We implement a modular approach on top of a grammar of derivation rules (actions) as the base model. Similar to Iyyer, Yih, and Chang (2017), our grammar consists of predefined actions used for predicting SELECT column, WHERE column, WHERE operator, WHERE value, and determining whether it is required to copy the entire action sequence of the previous turn questions. After encoding a question and previous turn questions into vectors, we first use a controller module to predict an action sequence consisting of slots, and then use specific modules to predict the argument of each slot. Similar to Iyyer, Yih, and Chang (2017), we use a recurrent structure as the backbone of each module and use the softmax layer for making prediction.

Settings The settings are the same as those described in table-based semantic parsing.

Results and Analysis From Table 4, we can see that question generation does not work well on this task. This is because the difficulty in generating sequential questions and complex target logical forms. Applying MAML to examples not coming from question generation performs best. We leave contextual question generation as a future work.

Related Work

Semantic Parsing Semantic parsing aims to transform a natural language utterance to a program/logical form, which is executable on an environment, or a world, to obtain the result. There are a variety of semantic parsing tasks with different types of environments (including large-scale knowledge base (Berant et al. 2013), web table (Pasupat and Liang 2015), image, 3D environment, etc.), and different types of programs (such as λ-calculus, dependency-based compositional semantics, SQL query (Zhong, Xiong, and Socher 2017), Bash command, source code, etc.). The majority of existing works study the problem in a supervised or weak-supervised setting, in which case either annotated programs or execution results are available during training. Earlier works on unsupervised semantic parsing do not ground to an environment such a KB or a web table (Poon and Domingos 2009). Goldwasser et al. (2011) align words to predicates, then do compositions and self training by iteratively adding self-annotated examples. Krishnamurthy and Mitchell (2012) identify relation instances from KB and produce parses that syntactically agree with the dependency parses. Reddy, Lapata, and Steedman (2014) parse sentences with CCG, and map results to ungrounded and grounded graph by regarding semantic parsing as graph matching. There are recent attempts at combining a limited number of supervised datapoints and artificially generated programs (Guo et al. 2018) with generative models.

More recently, Cheng, Reddy, and Lapata (2018) share the same motivation with this work that learns semantic parser from templates. We differ from them in two aspects. First, we only need domain-general rules but they also require domain-specific lexicon containing mapping from natural language expressions to database predicates/entities. Second, our method innovates the coupling of back-translation and model-agnostic meta-learning.

Meta-Learning Meta-learning, also known as learning to learn, is one of the potential techniques for enabling an artificial agent to mimic a human's ability in using past experience to quickly adapt to unseen situations. With the rapid progress of deep neural network models (Yin et al. 2018; Feng, Qin, and Liu 2018), existing approaches largely fall into two categories. In the first category, neural network models are trained to learn from datapoints that are passed in. Different neural architectures including LSTM, convolution, or memory-augmented ones have been explored. Methods in this category either use additional datapoints as a part of the input to predict the label for the new example, or optimizes the meta-learning based on the performance of the updated parameter of the learner (Finn, Abbeel, and Levine 2017). The second category aims to learn a metric space, where examples with the same class are close with each other. Notable works include Matching Networks and Prototypical Networks, both of which are tested on low-resource image classification. Matching networks use attention mechanisms to use the similarity between datapoints to predict the label of a test example. Prototypical Networks approach considers the representation of each class, which is obtained by averaging representation of examples belonging to that class. We use MAML (Finn, Abbeel, and Levine 2017), which has been recently used in low-resource multilingual neural machine translation and sequence-to-SQL generation (Huang et al. 2018) in supervised settings.

Back-Translation Back-translation is introduced for improving neural machine translation (NMT) by injecting monolingual data (Sennrich, Haddow, and Birch 2016), and is the key contributor to drive the recent success on unsupervised NMT (Lample et al. 2018). The training process involves a generative source-to-target translator $f_{s\to t}$ and a generative target-to-source translator $f_{t\to s}$. Because generative models are sensitive to the target distribution while tolerant to the noises from input, target sequences typically come from the real distribution during NMT training. Namely, examples generated by $f_{s\to t}$ are used to train $f_{t\to s}$, and examples generated by $f_{t\to s}$ are used to train $f_{s\to t}$. In semantic parsing, we do not regard sampled programs as noisy because they follow a certain grammar which can be guaranteed to be correctly executed.

Conclusion and Future Directions

We present an approach to learn neural semantic parser from simple domain-independent rules, instead of annotated logical forms or denotations. Our approach starts from examples covered by rules, which are used to initialize a semantic parser and a question generator in a back-translation paradigm. Generated examples are measured and filtered based on statistic analysis, and then used with model-agnostic meta-learning, which guarantees model's accuracy and stability on rule-covered examples, and acquires the versatility to generalize well on rule-uncovered examples. We conduct experiments on three datasets for table-based and knowledge-based question answering tasks. Results show that incorporating different strategies incrementally improves the performance. Our best model on WikiSQL achieves comparable accuracy to the system learned from denotation. In the future, we plan to focus on more complex logical forms.

Acknowledgements

Xiaocheng Feng is the corresponding author of this work. The authors would like to thank Shujie Liu, Shuo Ren and Yaming Sun, as well as the anonymous reviewers for discussions and comments. Yibo Sun, Xiaocheng Feng and Bing Qin are supported by the National Key R&D Program of China (No. 2018YFB1005103), National Natural Science Foundation of China (No. 61632011 and No.61772156).

References

Agarwal, R.; Liang, C.; Schuurmans, D.; and Norouzi, M. 2019. Learning to generalize from sparse and underspecified rewards. *CoRR* abs/1902.07198.

Antoniou, A.; Edwards, H.; and Storkey, A. 2019. How to train your maml. *ICLR*.

Berant, J.; Chou, A.; Frostig, R.; and Liang, P. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*, number 5, 6.

Bordes, A.; Usunier, N.; Chopra, S.; and Weston, J. 2015. Large-scale simple question answering with memory networks. *CoRR* abs/1506.02075.

Cheng, J.; Reddy, S.; and Lapata, M. 2018. Building a neural semantic parser from a domain ontology. *arXiv preprint arXiv:1812.10037*.

Dong, L., and Lapata, M. 2018. Coarse-to-fine decoding for neural semantic parsing. In *ACL*, 731–742. Association for Computational Linguistics.

Feng, X.; Qin, B.; and Liu, T. 2018. A language-independent neural network for event detection. *Science China Information Sciences* 61(9):092106.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. *ICML*.

Goldwasser, D.; Reichart, R.; Clarke, J.; and Roth, D. 2011. Confidence driven unsupervised semantic parsing. In *ACL*.

Guo, D.; Sun, Y.; Tang, D.; Duan, N.; Yin, J.; Chi, H.; Cao, J.; Chen, P.; and Zhou, M. 2018. Question generation from sql queries improves neural semantic parsing. In *EMNLP*, 1597–1607. Association for Computational Linguistics.

Huang, P.-S.; Wang, C.; Singh, R.; tau Yih, W.; and He, X. 2018. Natural language to structured query generation via meta-learning. In *NAACL-HLT*.

Iyyer, M.; Yih, W.-t.; and Chang, M.-W. 2017. Search-based neural structured learning for sequential question answering. In *ACL*, 1821–1831. Vancouver, Canada: Association for Computational Linguistics.

Krishnamurthy, J., and Mitchell, T. M. 2012. Weakly supervised training of semantic parsers. In *EMNLP*, 754–765. Association for Computational Linguistics.

Lample, G.; Ott, M.; Conneau, A.; Denoyer, L.; and Ranzato, M. 2018. Phrase-based & neural unsupervised machine translation. In *EMNLP*.

Liang, C.; Berant, J.; Le, Q.; Forbus, K. D.; and Lao, N. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *ACL*, 23–33. Vancouver, Canada: Association for Computational Linguistics.

Liang, C.; Norouzi, M.; Berant, J.; Le, Q. V.; and Lao, N. 2018. Memory augmented policy optimization for program synthesis and semantic parsing. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc. 10015–10027.

McClosky, D.; Charniak, E.; and Johnson, M. 2006. Effective self-training for parsing. In *HLT-NAACL*.

Misra, D.; Chang, M.-W.; He, X.; and Yih, W.-t. 2018. Policy shaping and generalized update equations for semantic parsing from denotations. *arXiv preprint arXiv:1809.01299*.

Neelakantan, A.; Le, Q. V.; Abadi, M.; McCallum, A.; and Amodei, D. 2016. Learning a natural language interface with neural programmer. *arXiv preprint arXiv:1611.08945*.

Pasupat, P., and Liang, P. 2015. Compositional semantic parsing on semi-structured tables. In *ACL*.

Petrochuk, M., and Zettlemoyer, L. 2018. Simplequestions nearly solved: A new upperbound and baseline approach. In *EMNLP*, 554–558. Association for Computational Linguistics.

Poon, H., and Domingos, P. 2009. Unsupervised semantic parsing. In *EMNLP*, 1–10. Association for Computational Linguistics.

Reddy, S.; Lapata, M.; and Steedman, M. 2014. Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics* 2:377–392.

Ren, S.; Zhang, Z.; Liu, S.; Zhou, M.; and Ma, S. 2019. Unsupervised neural machine translation with smt as posterior regularization. *AAAI*.

Sennrich, R.; Haddow, B.; and Birch, A. 2016. Improving neural machine translation models with monolingual data. In *ACL*, 86–96. Association for Computational Linguistics.

Shi, T.; Tatwawadi, K.; Chakrabarti, K.; Mao, Y.; Polozov, O.; and Chen, W. 2018. Incsql: Training incremental text-to-sql parsers with non-deterministic oracles. *arXiv preprint arXiv:1809.05054*.

Wang, S., and Jiang, J. 2016. Machine comprehension using match-lstm and answer pointer. *CoRR* abs/1608.07905.

Xu, X.; Liu, C.; and Song, D. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436*.

Yin, Q.; Zhang, Y.; Zhang, W.-N.; Liu, T.; and Wang, W. Y. 2018. Deep reinforcement learning for chinese zero pronoun resolution. In *ACL*, volume 1, 569–578.

Zhang, J.; Liu, Y.; Luan, H.; Xu, J.; and Sun, M. 2017. Prior knowledge integration for neural machine translation using posterior regularization. In *ACL*, 1514–1523.

Zhong, V.; Xiong, C.; and Socher, R. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.