Learning to Map Frequent Phrases to Sub-Structures of Meaning Representation for Neural Semantic Parsing

Bo Chen,[1,2] **Xianpei Han,**[1,2,*] **Ben He,**[3,1,*] **Le Sun**[1,2]
[1]Chinese Information Processing Laboratory [2]State Key Laboratory of Computer Science
Institute of Software, Chinese Academy of Sciences, Beijing, China
[3]University of Chinese Academy of Sciences, Beijing, China
{chenbo, xianpei, sunle}@iscas.ac.cn, benhe@ucas.ac.cn

Abstract

Neural semantic parsers usually generate meaning representation tokens from natural language tokens via an encoder-decoder model. However, there is often a vocabulary-mismatch problem between natural language utterances and logical forms. That is, one word maps to several atomic logical tokens, which need to be handled as a whole, rather than individual logical tokens at multiple steps. In this paper, we propose that the vocabulary-mismatch problem can be effectively resolved by leveraging appropriate logical tokens. Specifically, we exploit macro actions, which are of the same granularity of words/phrases, and allow the model to learn mappings from frequent phrases to corresponding sub-structures of meaning representation. Furthermore, macro actions are compact, and therefore utilizing them can significantly reduce the search space, which brings a great benefit to weakly supervised semantic parsing. Experiments show that our method leads to substantial performance improvement on three benchmarks, in both supervised and weakly supervised settings.

Introduction

Semantic parsing aims to transform natural language utterances to meaning representations (Zelle and Mooney 1996; Zettlemoyer and Collins 2005; Wong and Mooney 2007; Lu et al. 2008; Kwiatkowski et al. 2013; Yin et al. 2018). In recent years, encoder-decoder based neural models have achieved significant progress in semantic parsing (Xiao, Dymetman, and Gardent 2016; Dong and Lapata 2016; Jia and Liang 2016; Rabinovich, Stern, and Klein 2017; Chen, Sun, and Han 2018). Neural parsers model semantic parsing as a word sequence to logical token sequence translation task, where an encoder encodes the word sequence of an utterance, and an attention-guided decoder generates its logical form token-by-token. These tokens can be logical tokens of a linearized logical form, or atomic construction actions of a semantic graph. Figure 1 shows an example, where "*Which states border Texas?*" is translated to a logical token sequence [`answer`, `(`, `A` ...] or an atomic action sequence [`add_node:A`, `add_type:state` ...].

*Corresponding authors.

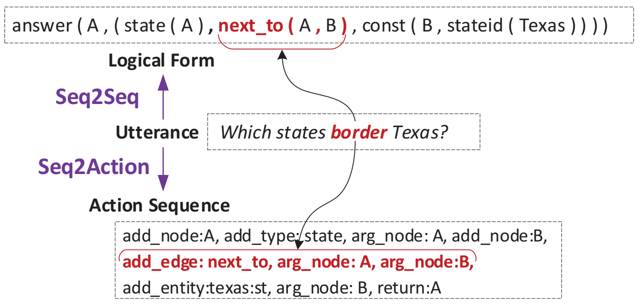

answer (A , (state (A) , next_to (A , B) , const (B , stateid (Texas))))
Logical Form
Seq2Seq
Utterance
Which states border Texas?
Seq2Action
Action Sequence
add_node:A, add_type: state, arg_node: A, add_node:B,
add_edge: next_to, arg_node: A, arg_node:B,
add_entity:texas:st, arg_node: B, return:A

Figure 1: A demonstration of Seq2Seq model and Seq2Action model for semantic parsing.

To adapt to the Seq2Seq framework, most neural semantic parsers decouple meaningful semantic units (e.g., CCG category, lambda term, etc.) into atomic logical tokens, so that the decoder can generate them just like words in neural machine translation. This assumption, unfortunately, often leads to a vocabulary-mismatch problem. For example, in Figure 1 the predicate triggered by word "*border*" will be decoupled to 4 atomic logical tokens [`next_to`, `(`, `,`, `)`] in lambda calculus or 3 actions [`add_edge:next_to`, `arg_node:A`, `arg_node:B`] in semantic graph representation. It is obvious that words and atomic logical tokens are of different granularities, and atomic logical tokens cannot act as separate meaningful semantic units.

The vocabulary-mismatch problem raises both effectiveness and efficiency issues for neural semantic parsing. Firstly, the logical tokens triggered by a word are generated in multiple individual steps, rather than being triggered at once. Therefore, neural parsers need to additionally model the dependency between different logical tokens, in order to prevent syntactically ill-formed logic forms (e.g., missing arguments for predicates, missing "`)`" after "`(`", missing nodes for an edge). Secondly, the granularity of logical tokens is usually too small, which is likely to result in long logical token sequence, and consequently, high time complexity in both training and decoding phases. For instance, in GEO dataset the average CCG category size of its sentences is 7.6, but the average length of linearized logical token sequences is 28.2. Such a length increase significantly enlarges the search space during decoding, making it hard to train neural parsers in weakly-supervised settings, which is critical for building semantic parsers in real world applications.

In this paper, we resolve the vocabulary-mismatch problem by leveraging macro logical tokens, which are of the same granularity of words/phrases. Our idea is motivated by traditional grammar-based semantic parsers, where the logical tokens of a word are not triggered separately, but triggered at once as a whole meaningful term. For example, for word "*border*", the atomic tokens in its CCG category $\lambda x.\lambda y.next\_to(x, y)$ are triggered at once, rather than triggered as 7 individual tokens. Using macro logical tokens, the neural models don't need to capture the dependency between atomic logical tokens, and its efficiency in both training and decoding is expected to be significantly improved, owing to the shorter sequence length.

To this end, this paper proposes the Seq2MacroAction model, which uses Seq2Action as the base model, extended with macro logical tokens – macro actions as demonstrated in Figure 2. In this way, the vocabulary-mismatch problem can be resolved by exploiting macro actions. Concretely, we use a frequent sub-structure mining algorithm to automatically collect macro actions, and choose macro actions based on two intuitions: 1) the macro actions need to be meaningful semantic units which are of the same granularity of words/phrases; 2) the macro actions should be general enough to be applicable to different domains/datasets. For example, using the proposed algorithm, we retain a single macro action for word "*border*" and phrase "*border Texas*", but do not retain a single macro action for phrase "*border state that has a city named*", as the latter is too specific. Additionally, we propose an effective macro action embedding algorithm, so that our macro actions can be adapted to current Seq2Seq based neural parsers without any additional effort.

We evaluate our approach on both supervised and weakly-supervised semantic parsing tasks on three standard datasets—GEO, ATIS and JOBS. Experimental results show that, by collecting macro actions and integrating them in Seq2Seq framework, our Seq2MacroAction is more effective and more efficient. Firstly, because we don't need to model the dependencies between atomic logical tokens, our model can be more easily learned. Experimental results show that our Seq2MacroAction model achieves competitive performances on all three datasets in supervised setting. Secondly, macro actions can significantly reduce the complexity of search space, therefore lead to more efficient decoding. For instance, macro actions can reduce the logical token sequence length from 28 to 5 in GEO dataset. In this way, our macro action based semantic parsers can be directly trained using weakly-supervised techniques, without extra resources or strategies. Experimental results also verify that our method significantly outperforms previous weakly-supervised models, and even competitive to supervised semantic parsers.

The main contributions of this paper are as follows:

- We propose a promising direction for resolving the vocabulary-mismatch problem in neural semantic parsers, i.e., leveraging macro logical tokens which are of the same granularity of words/phrases. This solution can benefit many neural semantic parsers in improving both their effectiveness and efficiency.
- We propose the Seq2MacroAction model, which presents how to automatically collect macro actions – the macro logical tokens for semantic graph representation, and shows how to incorporate macro actions in Seq2Seq framework using an effective macro action embedding algorithm. The Seq2MacroAction achieves competitive performance in both supervised/weakly-supervised settings on three standard datasets.

Base Seq2Action Model

This section briefly describes the base model used in this work –Seq2Action (Chen, Sun, and Han 2018). Seq2Action models semantic parsing as an end-to-end semantic graph generation process, which maps the word sequence of a sentence to a sequence of atomic semantic graph construction actions. This paper selects Seq2Action as our base model because it can simultaneously leverage the advantages of semantic graph representation and the strong prediction ability of Seq2Seq models. In the following we briefly describe the Seq2Action model.

Semantic Graph Representation. Seq2Action uses semantic graphs as meaning representations, where each semantic graph is a sub-graph of a knowledge graph (e.g., Freebase[1]). A semantic graph consists of nodes, edges and some operations (e.g., `count`, `argmax`). A node can be a variable or an entity. An edge corresponds to a relation in knowledge graph. In the original paper (Chen, Sun, and Han 2018), 6 different kinds of atomic semantic graph construction actions are designed, namely `add_node`, `add_type`, `add_entity`, `add_edge`, `add_operation` and `arg_node`.

Semantic Parsing as Semantic Graph Generation. Based on the above atomic actions, Seq2Action parses a sentence by translating it to a sequence of semantic graph construction actions. Specifically, given a sentence $X = x_1, ..., x_{|X|}$, Seq2Action generates a sequence of actions $Y = y_1, ..., y_{|Y|}$ using a constrained decoder (Krishnamurthy, Dasigi, and Gardner 2017; Sun et al. 2018), which leverages structure constraints and semantic constraints to generate well-formed action sequences.

Learning. Given a training corpus $\{(X_1, Y_1), ..., (X_n, Y_n)\}$, where each instance is a sentence X_i paired with its action sequence Y_i, Seq2Action maximizes the likelihood of the generated action sequence given X_i using the objective function:

$$\sum_{i=1}^{n} \log P(Y_i|X_i) \qquad (1)$$

where the conditional probability $P(Y|X)$ used in

[1]https://en.wikipedia.org/wiki/Freebase

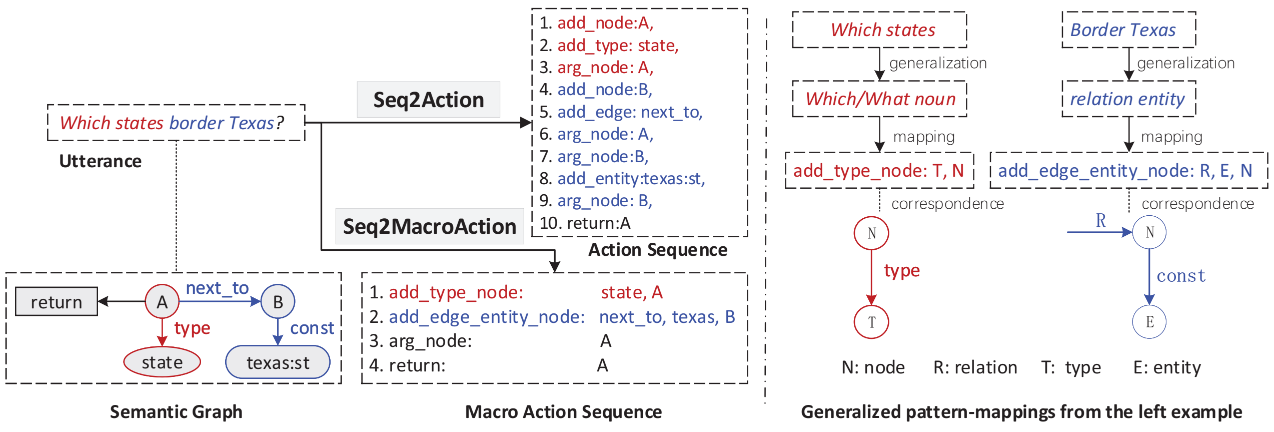


Figure 2: Demonstrations of semantic parsing with macro actions (Compared with atomic action sequence, the length of macro action sequence has been reduced from 10 to 4). These parts with the same color (red and blue) in Utterance, Semantic Graph, Action Sequence and Macro Action Sequence indicate their correspondences. Our parser learns to map frequent language patterns (e.g., *relation entity*) to macro actions (e.g., `add_edge_entity_node`), which denote corresponding sub-structures in a semantic graph.

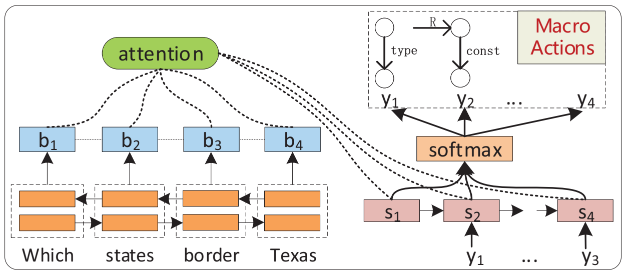


Figure 3: The framework of our Seq2MacroAction model, which encodes an input sentence to a vector and then utilizes an attention-guided decoder to generate macro actions one by one.

Seq2Action is decomposed as follows:

$$P(Y|X) = \prod_{t=1}^{|Y|} P(y_t|y_1, ..., y_{t-1}, X) \tag{2}$$

Macro Actions for Neural Semantic Parsing

This section describes our Seq2MacroAction model, which can resolve the vocabulary-mismatch problem by exploiting macro actions in neural semantic parsing. Figure 3 shows the framework of our Seq2MacroAction model. Compared with Seq2Action, Seq2MacroAction outputs macro actions, rather than atomic actions, and is therefore more effective and efficient due to shorter sequence length. To this end, there are two main issues to be addressed. Firstly, how to obtain meaningful macro actions which are of the same granularity with natural language words/phrases. Secondly, how to effectively incorporate macro actions into the encoder-decoder architecture. Solutions to these two issues are described in details as follows.

Macro Actions

As discussed above, macro actions need to: 1) be of the same granularity of natural language words/phrases, so that the vocabulary-mismatch problem can be resolved; 2) be generalized enough, so that the macro-action based model can fit to different datasets/representations/domains. It is obvious that atomic actions are not appropriate because their granularity is too small, although they are generalized enough. In this paper, we obtain macro actions automatically with a frequent sub-structure mining algorithm (see Algorithm 1)[2], which is described as follows.

Firstly, the meaning representations of all sentences are decomposed into sub-structures with all granularities[3]. For example, for the semantic graph representation in Figure 2, we get sub-structures with granularity of 1: `node:A`, `node:state`, `node:texas`, `edge:next_to`, `edge:type`, `edge:const`. Then we identify frequent sub-structures according to their frequencies. We filter the frequent sub-structures using by only considering sub-structures with their granularities between 3 to 4, since sub-structures with too large granularity are of low generalization ability. Finally, we generalize the extracted frequent sub-structures to get the macro-actions. For example, for frequent sub-structure `next_to.state(A)`, we obtain a macro action `add_edge_type_node`, which is a more general form of the sub-structure by abstracting `state` to `type` and abstracting `next_to` to `edge`. There are in total 5 kinds of macro actions we collected as described as follows and their examples are shown in Table 1:

Add Type Node: This kind of macro actions denotes a variable node with a type, which are mostly triggered by en-

[2]Notice that this algorithm can also be adapted to other meaning representations.

[3]Since a semantic graph usually only has a small number of nodes and edges, so our algorithm is efficient in practice.

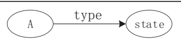
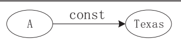
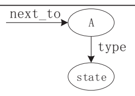
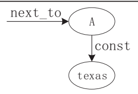
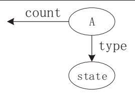
Macro Action Type	Phrase Pattern	Example		
		Phrase	Macro Action	Sub-Structure
Add Type Node	*which/what category*	*which states*	structure: `add_type_node` semantic: `state,A`	
Add Entity Node	*entity*	*Texas*	structure: `add_entity_node` semantic: `Texas,A`	
Add Edge Type Node	*relation category*	*border states*	structure: `add_edge_type_node` semantic: `next_to,state,A`	
Add Edge Entity Node	*relation entity*	*border Texas*	structure: `add_edge_entity_node` semantic: `next_to,texas,A`	
Add Ope Type Node	*count category*	*how many states*	structure: `add_ope_type_node` semantic: `count,state,A`	

Table 1: Macro action types (5 in total) and examples. Each kind of macro action type corresponds to a phrase pattern in natural language, and denotes a sub-structure in knowledge base.

Algorithm 1 Frequent Sub-Structure Mining

```
Input: $\{(x_i, y_i)\}$ pairs, $x_i$ is an utterance, and $y_i$ is its meaning
    representation.
Output: Frequent sub-structure set $\mathcal{FSS}$
1: function FreSubStrucExt($\{(x_i, y_i)\}$)
2:     $\mathcal{SS} \leftarrow \emptyset$
3:     for $i = 0 \rightarrow n - 1$ do
4:         Decompose $y_i$ to get all sub-structures $SS_i$
5:         $\mathcal{SS} \leftarrow \mathcal{SS} \cup SS_i$
6:     end for
7:     Extract frequent sub-structures from $\mathcal{SS}$ according to fre-
    quency
8:     return Frequent sub-structure set $\mathcal{FSS}$
9: end function
```

tity type words such as "*river*", "*person*", etc. We denote this kind of action as `add_type_node:T,N`, where `T` indicates the type, and `N` is the identifier of the variable node.

Add Entity Node: This kind of macro actions denotes an entity node (e.g., *Texas*) and is represented as `add_entity_node:E,N`, where `E` indicates the entity, and `N` is the identifier of the entity node. This kind of actions are mostly triggered by entity names, such as "*Texas*", "*Steve Jobs*", etc.

Add Edge Type Node: This kind of macro actions denotes a sub-structure which adds a relational constraint to a typed variable node. This is a frequent sub-structure for "*relation category*". For example, "*border states*". It consists of an edge and a variable node with a type. We denote this kind of actions as `add_edge_type_node:R,T,N`, where `R` indicates the edge (corresponding to a *relation*), `T` indicates the type, and `N` is the identifier of the variable node.

Add Edge Entity Node: This kind of macro actions denotes a sub-structure which indicates a relational constraint in semantic graph. For example, the phrase "*border Texas*" in "*which states border Texas*" will add a relational constraint to the variable node indicated by "*states*". This kind of macro actions consists of an edge and an entity node. We denote this kind of actions as `add_edge_ent_node:R,E,N`, where `R` indicates the edge, `E` indicates the entity, and `N` is the identifier of the variable node.

Add Ope Type Node: This kind of macro actions denotes a sub-structure which indicates an operation attached to a typed variable node. For example, the phrase "*how many states*" associates a `count` operation attached to a variable node with the type of "*state*". It consists of an operation and a variable node with a type. We denote this kind of actions as `add_ope_type_node:O,T,N`, where `O` indicates the operation (e.g., `count`, `argmax`), `T` indicates the type, and `N` is the identifier of the variable node.

The above macro actions are of almost the same granularity of words or frequent phrases. Meanwhile, the macro actions are generalized enough, as their generalization ability is the same as the lexicons used in traditional grammar-based semantic parsers. Furthermore, by leveraging macro actions for frequent phrases, similar to the phrase based machine translation (Koehn, Och, and Marcu 2003), our macro actions can further reduce the ambiguity of actions. We can automatically generate all macro actions from knowledge base (in weakly-supervised setting) or from training corpus (in supervised settings) by enumerating all above sub-structures.

Seq2MacroAction

To incorporate macro actions into neural semantic parsing, our Seq2MacroAction model first encodes sentences as the same as Seq2Action, then the decoder generates macro action sequence for constructing semantic graph. For example, our method will generate [`add_type_node:state,A`; `add_edge_enti_node:next_to,texas,B`; `arg_node:A`; `return:A`] in Figure 2. We can see that, the main task here is how to embed macro actions, rather than atomic actions.

Macro2Action Embedding. From our experience, macro actions are composed of atomic actions with usually unique

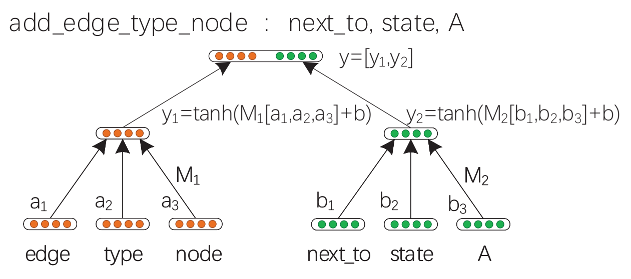


Figure 4: Macro action embedding model.

structures, and therefore, simply embedding each macro action as a whole likely leads to sparsity and redundancy problems. To this end, this paper embeds macro actions by taking their structures into consideration. Intuitively, macro actions usually contain many functional parts, such as the structure part and the semantic part, which are common between different macro actions. Based on this intuition, we embed macro actions as follows.

Concretely, as Figure 4 shows, the embedding of each macro action is the concatenation of the embedding of its structure part (`add_edge_type_node`) and semantic part (`next_to, state, A`). Each structure part or semantic part is composed of primitive elements, e.g., `edge`, `type`, `node` in the structure part `add_edge_type_node`. In order to make parameters more compact and enable more information sharing of macro actions, we first embed every primitive elements, and then use a recursive neural network (Socher, Manning, and Ng 2010) to get the embedding for each structure part and semantic part. Specifically, we combine the embeddings of different elements using the weight matrix M[4].

Weakly-Supervised Learning

Because macro actions can significantly reduce the search complexity of semantic parsing, it is easy to train our model using weakly-supervised learning techniques.

The difference between supervised setting and weakly-supervised setting is the learning algorithm. Specifically, for weakly-supervised learning we use a learning algorithm called *maximum margin average violation reward* (MAVER), proposed by Misra et al. (2018). For each training example (x_i, z_i), this algorithm finds the highest scoring program y_i that evaluates to z_i, as the *reference* program, from the set $\mathcal{K}$ of the programs generated by the search. With a margin function δ, and reference program y, the set of programs $\mathcal{V}$ that violate the margin constraint can thus be defined as:

$$\begin{aligned}\mathcal{V} = \{y'|y' \in \mathcal{Y} \ \textbf{and} \ \ score_\theta(y,x) \\ \leq score_\theta(y',x) + \delta(y,y',z)\}\end{aligned} \tag{3}$$

[4]The matrix for structure part and semantic part are different, but each kind of macro action shares the same matrix for structure and the same matrix for semantic part. These matrices are parameters learned from training data.

The most violating program for the constraint can be written as:

$$\begin{aligned}\hat{y} = \arg\max_{y' \in \mathcal{Y}} \{score_\theta(y',x) + \delta(y,y',z) \\ -score_\theta(y,x)\}\end{aligned} \tag{4}$$

The average-violation margin objective is thus defined as:

$$\begin{aligned}\mathcal{J}_{MAVER} = -max\{0, score_\theta(\hat{y}, x_i) \\ -\frac{1}{|\mathcal{V}|}\sum_{y \in \mathcal{V}} score_\theta(y_i, x_i) + \delta(y_i, \hat{y}, z_i)\}\end{aligned} \tag{5}$$

Using this update strategy, the learning algorithm considers more negative examples during each update, therefore can achieve stable performance.

Experiments

Datasets

ATIS contains natural language questions of a flight database, in which each question is annotated with a lambda calculus query. Following Zettlemoyer and Collins (2007), we use the standard 4473/448 for train/test instance split.

GEO contains natural language questions about US geography paired with corresponding Prolog database queries. A US geography database and the answers for all questions are also provided. Following Zettlemoyer and Collins (2005), we use the standard 600/280 train/test instance split.

JOBS contains natural language questions about jobs paired with Prolog database queries. The jobs database and the answers for each question are also provided[5]. Following Zettlemoyer and Collins (2005), we use the standard 500/140 train/test instance split.

Experiment Setups

For utterances encoder, we use 200 hidden units and 100-dimensional word vectors. For decoder, we tune the dimensions of action embedding on validation datasets. All parameters are initialized by uniformly sampling within the interval [-0.1, 0.1]. We train our model for a total of 30 epochs with an initial learning rate of 0.1, and halve the learning rate every 5 epochs after the first 15 epochs. We use a universal word embedding for words occurring only once.

We conduct experiments on GEO, JOBS, and ATIS for supervised semantic parsing, and conduct experiments on GEO and JOBS for weakly-supervised semantic parsing. We reimplement two base systems. One is Seq2Seq based on Jia and Liang (2016). The other is Seq2Action based on Chen, Sun, and Han (2018). We use the same evaluation strategy as Jia and Liang (2016) for GEO and JOBS, and the same strategy as (Dong and Lapata 2016) for ATIS.

Main Results

Tables 2 & 3 show our main results. From these tables, we can see that:

[5]We use the same jobs database as Liang, Jordan, and Klein (2011).

1. The proposed Seq2MacroAction model achieves competitive performance on all three datasets. On GEO dataset, our model gets the best accuracy of 89.6. On ATIS dataset, our model gets the state-of-the-art accuracy of 88.2. On JOBS dataset, our model gets a competitive accuracy of 92.1, only falls behind Rabinovich, Stern, and Klein (2017) with SUPATT which uses an extra supervised attention mechanism. These results confirm the effectiveness of using logical tokens of larger granularity in neural semantic parsing. For example, in ATIS dataset, macro actions will enable phrase-level mapping for frequent phrases "*from city*", "*to city*", "*earliest flight*", etc. In this way the mapping ambiguity and the decoding complexity can be significantly reduced.

2. By leveraging logical tokens, having the same granularity of words/phrases, our model achieves significant improvement in both supervised and weakly-supervised settings. Compared with atomic action based baseline Seq2Action, our model achieves performance improvements on all three datasets. This verifies the effectiveness of leveraging logical tokens as meaningful semantic units as a whole, rather than individual logical tokens at multiple steps.

3. By significantly reducing search complexity, our method achieves good performance in weakly-supervised setting. Our model outperforms the base systems significantly, and even gets competitive results compared to supervised semantic parsers. On the GEO dataset, our model gets the test accuracy of 86.4, better than the accuracy 82.1 of Seq2Action baseline. On the JOBS dataset, our model gets the test accuracy of 87.1, better than the accuracy of 84.3 of Seq2Action baseline. We argue that the improvement is brought by macro actions' ability in explicitly encoding common dependencies, and thus enable the model to learn better pattern-subgraph mappings, which can greatly benefit weakly-supervised semantic parsers.

4. The proposed macro actions have a generalization ability. Our Seq2MacroAction gets strong results on all three datasets from different domains. This demonstrates that the logical tokens can be effectively obtained by mining frequent sub-structures to enhancing neural semantic parsing.

Detailed Analysis

Length of Macro Action Sequences. Table 4 presents the average lengths of logical forms, action sequences and macro action sequences. We can see that the length of macro action sequence is significantly shorter than other action sequences, and is similar to utterances. We believe short action sequence is an advantage since many long-term dependency problems can be avoided and the decoding can be more efficient.

Effect of Macro Action Embedding Algorithm. Here we conduct experiments to evaluate the effect of our macro action embedding algorithm (RNN+Conca.). The first comparing algorithm is embedding each macro action as a whole (Simple Embedding). The second one is summing up the structure part embedding and semantic part embedding separately, then concatenating them (Summing+Conca.). Table 5 shows the results. We observe a large drop in performance for all datasets when embedding each macro action as a whole, showing that the sparsity problem is serious and can be resolved by sharing sub-part embedding. We also observe performance drops by 2.3 points on average when summing up embedding then concatenating, showing that considering the inner structure of macro action is helpful.

	GEO	JOBS
Supervised Systems		
Zettlemoyer and Collins (2005)	79.3	79.3
Zettlemoyer and Collins (2007)	86.1	-
Kwiatkowksi et al. (2010)	88.9	-
Kwiatkowski et al. (2011)	88.6	-
Zhao and Huang (2015)	88.9	85.0
Jia and Liang (2016)	85.0	-
Jia and Liang (2016)* (+data)	89.3	-
Dong and Lapata (2016): 2Seq	84.6	87.1
Dong and Lapata (2016): 2Tree	87.1	90.0
Rabinovich, Stern, and Klein (2017)	85.7	91.4
Rabinovich, Stern, and Klein (2017)+SUPATT	87.1	**92.9**
Dong and Lapata (2018)	88.2	-
Chen, Sun, and Han (2018)	88.9	-
Seq2MacroAction	**89.6**	92.1
Weakly-Supervised Systems		
Liang, Jordan, and Klein (2011)*	87.9	90.7
Seq2Seq	78.6	77.1
Seq2Action	82.1	84.3
Seq2MacroAction	86.4	87.1

Table 2: Test accuracies on GEO and JOBS datasets in both supervised and weakly-supervised settings, where * indicates systems with extra-resources used.

	ATIS
Zettlemoyer and Collins (2007)	84.6
Kwiatkowksi et al. (2010)	71.4
Kwiatkowski et al. (2011)	82.8
Poon (2013)	83.5
Zhao and Huang (2015)	84.2
Dong and Lapata (2016): 2Seq	84.2
Dong and Lapata (2016): 2Tree	84.6
Jia and Liang (2016)	76.3
Jia and Liang (2016)* (+data)	83.3
Rabinovich, Stern, and Klein (2017)	85.3
Rabinovich, Stern, and Klein (2017)+SUPATT	85.9
Chen, Sun, and Han (2018)	85.5
Dong and Lapata (2018)	87.7
Seq2MacroAction	**88.2**

Table 3: Test accuracies on ATIS, where * indicates systems with extra-resources used.

	GEO	ATIS	JOBS
Utterance	7.60	11.10	9.80
Logical Form	28.20	28.40	22.90
Action	18.20	25.80	16.24
Macro Action	**5.51**	**8.36**	**7.02**

Table 4: Average length of utterance, logical forms, action sequences and macro action sequences on three datasets.

Attention Heatmap based on Macro Actions. Figure 5 shows the heatmap of a Seq2MacroAction parsing example. Notice that, our model can learn ef-

	GEO	ATIS	JOBS
RNN+Conca.	**89.6**	**88.2**	**92.1**
Simple Embedding	83.9	84.2	87.8
Summing+Conca.	87.9	86.0	89.2

Table 5: Results for using different macro action embedding algorithms.

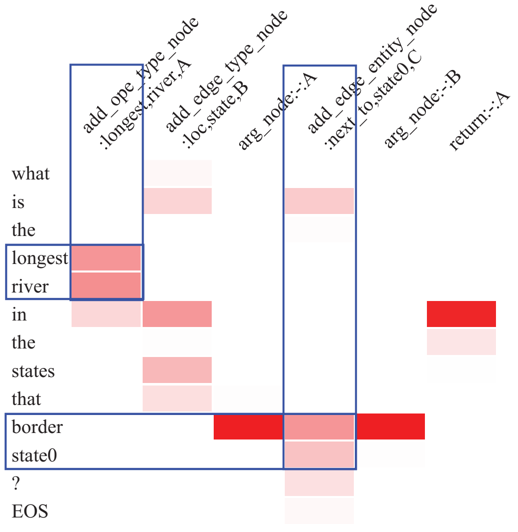


Figure 5: An attention heatmap example of our Seq2MacroAction model (darker color indicates higher attention score). Alignments are showed with blue rectangles.

fective mapping patterns from phrases to sub-meaning structures. For instance, there is a strong soft alignment between phrase "*longest river*" and macro actions `add_ope_type_node:longest,river,A` in the heatmap.

Related Work

Traditional semantic parsers are usually lexicon-grammar based models (Zettlemoyer and Collins 2005; Liang, Jordan, and Klein 2011; Cai and Yates 2013; Berant et al. 2013). These parsers need to learn a lexicon and to define compositional grammars. In recent years, Seq2Seq models have achieved significant progress (Dong and Lapata 2016; Jia and Liang 2016; Chen, Sun, and Han 2018; Herzig and Berant 2018; Guo et al. 2019; Bogin, Berant, and Gardner 2019; Shaw et al. 2019). Meanwhile, there are some studies focusing on using other meaning representations (e.g., semantic graphs) or designing effective learning algorithm for different situations (e.g., weakly-supervised semantic parsing).

Semantic Parsing using Semantic Graph. Compared to traditional logical forms, semantic graph representations have a tight-coupling with the knowledge base, and shares many commonalities with syntactic structures. Therefore structure and semantic constraints from knowledge base can be easily exploited during parsing. Using semantic graphs as meaning representation, semantic parsing can be formulated as a semantic graph generation task. Reddy, Lapata, and Steedman (2014) constructs semantic graphs by transforming dependency graph from CCG parse. Yih et al. (2015) constructs semantic graphs based on three heuristic steps. Bast and Haussmann (2015) uses three templates to construct semantic graphs. Chen, Sun, and Han (2018) generates action sequences, which can be used to construct semantic graphs.

Weakly Supervised Semantic Parsing. Supervised approaches often suffer from the lack of training data, because it is expensive to annotate an utterance with its logical form. Many weakly supervised techniques have been investigated, e.g., supervised using denotations for utterances (Liang et al. 2017; Guu et al. 2017; Cheng and Lapata 2018; Misra et al. 2018; Goldman et al. 2018; Liang et al. 2018; Agrawal et al. 2019). One main challenge here is the large search space of potential programs needing to be explored. Misra et al. (2018) defines a new learning algorithm to address this challenge. Goldman et al. (2018) alleviates this problem by utilizing an abstract representation, where tokens in both the language utterance and program are lifted to an abstract form. However the search space of these methods is still not reduced. The number of actions to build target programs is large. Compared to these methods, our method focuses on reducing the number of actions, which can reduce search space substantially.

Macro Grammars for Semantic Parsing. In order to speed up semantic parsing, Zhang, Pasupat, and Liang (2017) proposed a new learning algorithm using macro grammars and holistic triggering. The idea of macro grammars is that macros can capture the overall shape of computations in a way that can generalize across different utterances and knowledge bases. However, their base grammar must be general enough to build programs for complex utterances, leading to a huge search space. And their grammars are used for traditional lexicon-grammar based semantic parsing. Our idea of macro actions is partially inspired by their idea. The main difference is that our method can not only reduce the search space, but can also be easily incorporated to Seq2Seq neural semantic parsing.

Conclusions

This paper presents a new algorithm – Seq2MacroAction, which can resolve the vocabulary-mismatch problem in neural semantic parsing. Compared to previous studies, macro actions can model word/phrase level mappings, thus the ambiguity of logical tokens is significantly reduced and the search space for generating gold programs is greatly reduced. Experimental results show that our model not only outperforms base models significantly, but also obtains strong performance in weakly supervised semantic parsing settings. We believe this work explores a promising direction for enhancing neural semantic parsers – by leveraging more appropriate output logical tokens, and this will benefit and can be used in many other neural semantic parsers.

Acknowledgments

This work is supported by the National Natural Science Foundation of China under Grants no. 61433015, 61572477 and 61772505; and Huawei Noah's Ark Lab; and Beijing Academy of Artificial Intelligence(BAAI). Moreover, we sincerely thank the reviewers for their valuable comments.

References

Agrawal, P.; Dalmia, A.; Jain, P.; Bansal, A.; Mittal, A.; and Sankaranarayanan, K. 2019. Unified semantic parsing with weak supervision. In *Proceedings of ACL*, 4801–4810.

Bast, H., and Haussmann, E. 2015. More accurate question answering on freebase. In *Proceedings of CIKM*, 1431–1440.

Berant, J.; Chou, A.; Frostig, R.; and Liang, P. 2013. Semantic parsing on Freebase from question-answer pairs. In *Proceedings of EMNLP*, 1533–1544.

Bogin, B.; Berant, J.; and Gardner, M. 2019. Representing schema structure with graph neural networks for text-to-SQL parsing. In *Proceedings of ACL*, 4560–4565.

Cai, Q., and Yates, A. 2013. Large-scale semantic parsing via schema matching and lexicon extension. In *Proceedings of ACL*, 423–433.

Chen, B.; Sun, L.; and Han, X. 2018. Sequence-to-action: End-to-end semantic graph generation for semantic parsing. In *Proceedings of ACL*, 766–777.

Cheng, J., and Lapata, M. 2018. Weakly-supervised neural semantic parsing with a generative ranker. In *CoNLL*, 356–367.

Dong, L., and Lapata, M. 2016. Language to logical form with neural attention. In *Proceedings of ACL*, 33–43.

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

Goldman, O.; Latcinnik, V.; Nave, E.; Globerson, A.; and Berant, J. 2018. Weakly supervised semantic parsing with abstract examples. In *Proceedings of ACL*, 1809–1819.

Guo, J.; Zhan, Z.; Gao, Y.; Xiao, Y.; Lou, J.-G.; Liu, T.; and Zhang, D. 2019. Towards complex text-to-SQL in cross-domain database with intermediate representation. In *ACL*, 4524–4535.

Guu, K.; Pasupat, P.; Liu, E.; and Liang, P. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Proceedings of ACL*, 1051–1062.

Herzig, J., and Berant, J. 2018. Decoupling structure and lexicon for zero-shot semantic parsing. In *EMNLP*, 1619–1629.

Jia, R., and Liang, P. 2016. Data recombination for neural semantic parsing. In *Proceedings of ACL*, 12–22.

Koehn, P.; Och, F. J.; and Marcu, D. 2003. Statistical phrase-based translation. In *Proceedings of NAACL*.

Krishnamurthy, J.; Dasigi, P.; and Gardner, M. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of EMNLP*, 1516–1526.

Kwiatkowksi, T.; Zettlemoyer, L.; Goldwater, S.; and Steedman, M. 2010. Inducing probabilistic CCG grammars from logical form with higher-order unification. In *Proceedings of EMNLP*, 1223–1233.

Kwiatkowski, T.; Zettlemoyer, L.; Goldwater, S.; and Steedman, M. 2011. Lexical generalization in ccg grammar induction for semantic parsing. In *Proceedings of EMNLP*, 1512–1523.

Kwiatkowski, T.; Choi, E.; Artzi, Y.; and Zettlemoyer, L. 2013. Scaling semantic parsers with on-the-fly ontology matching. In *Proceedings of EMNLP*, 1545–1556.

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 *Proceedings of ACL*, 23–33.

Liang, C.; Norouzi, M.; Berant, J.; Le, Q. V.; and Lao, N. 2018. Memory augmented policy optimization for program synthesis and semantic parsing. In *Proceedings of NeurIPS*, 9994–10006.

Liang, P.; Jordan, M.; and Klein, D. 2011. Learning dependency-based compositional semantics. In *Proceedings of ACL*, 590–599.

Lu, W.; Ng, H. T.; Lee, W. S.; and Zettlemoyer, L. S. 2008. A generative model for parsing natural language to meaning representations. In *Proceedings of EMNLP*, 783–792.

Misra, D.; Chang, M.-W.; He, X.; and Yih, W.-t. 2018. Policy shaping and generalized update equations for semantic parsing from denotations. In *Proceedings of EMNLP*, 2442–2452.

Poon, H. 2013. Grounded unsupervised semantic parsing. In *Proceedings of ACL*, 933–943.

Rabinovich, M.; Stern, M.; and Klein, D. 2017. Abstract syntax networks for code generation and semantic parsing. In *Proceedings of ACL*, 1139–1149.

Reddy, S.; Lapata, M.; and Steedman, M. 2014. Large-scale semantic parsing without question-answer pairs. *TACL* 2:377–392.

Shaw, P.; Massey, P.; Chen, A.; Piccinno, F.; and Altun, Y. 2019. Generating logical forms from graph representations of text and entities. In *Proceedings of ACL*, 95–106.

Socher, R.; Manning, C. D.; and Ng, A. Y. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*.

Sun, Y.; Tang, D.; Duan, N.; Ji, J.; Cao, G.; Feng, X.; Qin, B.; Liu, T.; and Zhou, M. 2018. Semantic parsing with syntax- and table-aware sql generation. In *Proceedings of ACL*, 361–372.

Wong, Y. W., and Mooney, R. 2007. Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of ACL*, 960–967.

Xiao, C.; Dymetman, M.; and Gardent, C. 2016. Sequence-based structured prediction for semantic parsing. In *Proceedings of ACL*, 1341–1350.

Yih, W.-t.; Chang, M.-W.; He, X.; and Gao, J. 2015. Semantic parsing via staged query graph generation: Question answering with knowledge base. In *Proceedings of the ACL*, 1321–1331.

Yin, P.; Zhou, C.; He, J.; and Neubig, G. 2018. Structvae: Tree-structured latent variable models for semi-supervised semantic parsing. In *Proceedings of ACL*, 754–765.

Zelle, J. M., and Mooney, R. J. 1996. Learning to parse database queries using inductive logic programming. In *AAAI/IAAI*, 1050–1055.

Zettlemoyer, L. S., and Collins, M. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *Proceedings of UAI*, 658–666.

Zettlemoyer, L., and Collins, M. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of EMNLP*, 678–687.

Zhang, Y.; Pasupat, P.; and Liang, P. 2017. Macro grammars and holistic triggering for efficient semantic parsing. In *Proceedings of EMNLP*, 1214–1223.

Zhao, K., and Huang, L. 2015. Type-driven incremental semantic parsing with polymorphism. In *Proceedings of NAACL*, 1416–1421.