BIG DATA MINING AND ANALYTICS
ISSN 2096-0654 01/08 pp217–239
Volume 2, Number 4, December 2019
DOI: 10.26599/BDMA.2019.9020011

Statistical Learning for Semantic Parsing: A Survey

Qile Zhu, Xiyao Ma, and Xiaolin Li*

Abstract: A long-term goal of Artificial Intelligence (AI) is to provide machines with the capability of understanding natural language. Understanding natural language may be referred as the system must produce a correct response to the received input order. This response can be a robot move, an answer to a question, etc. One way to achieve this goal is semantic parsing. It parses utterances into semantic representations called logical form, a representation of many important linguistic phenomena that can be understood by machines. Semantic parsing is a fundamental problem in natural language understanding area. In recent years, researchers have made tremendous progress in this field. In this paper, we review recent algorithms for semantic parsing including both conventional machine learning approaches and deep learning approaches. We first give an overview of a semantic parsing system, then we summary a general way to do semantic parsing in statistical learning. With the rise of deep learning, we will pay more attention on the deep learning based semantic parsing, especially for the application of Knowledge Base Question Answering (KBQA). At last, we survey several benchmarks for KBQA.

Key words: deep learning; semantic parsing; Knowledge Base Question Answering (KBQA)

1 Introduction

Enabling machines to understand natural language with big challenge and huge promising future, has attracted so many attention in the last few decades. Meanwhile, semantic parsing, as a subfield of Natural Language Processing (NLP), aims to map utterances into a precise representation of its semantic meaning with a formal language, which is executable and understandable by machines. Semantic parsing has been widely adopted for language reasoning and question answering with knowledge base[1]. Before we introduce the formal language for semantic parsing, let us give some examples of some pairs of utterance-action pairs (Table 1)[2]. We are interested in utterances listed in Table 1, which may require understanding and reasoning for natural language. The formal language for representing semantic meaning is called *logical forms*. For example, the first sentence in Table 1 would map onto the logical form $max(primes \cap (-\infty, 10))$. We can treat the logical form as an executable program to get the desired behavior (e.g., answers to questions, queries from a database, or an invocation of an API).

Semantic parsing, similar to machine translation, involves in translating from one semantic representation into another one. Both of them are dealing with complex structures, often related in complex ways. However, the target of semantic parsing is machine-readable while machine translation is human-readable. Logical forms play a foundational role in natural language understanding systems and building an interface for human and machine. Luna[3], a natural language

Table 1 Examples for utterance-action pairs.

Utterance	Action
What is the largest prime less than 10?	7
What is the highest mountain in the world?	Mount Everest
Call the number 1234567	Open the phone APP and make a call

• Qile Zhu, Xiyao Ma, and Xiaolin Li are with National Science Foundation Center for Big Learning, University of Florida, Gainesville, FL 32608, USA. E-mail: valder@ufl.edu; maxiy@ufl.edu; andyli@ece.ufl.edu.

* To whom correspondence should be addressed.
Manuscript received: 2018-09-17; accepted: 2019-04-29

interface to a database about moon rocks, and SHRDLU, a toy blocks world environment, could both answer questions and perform actions[4]. These systems achieved significant achievements in the early time. They could handle complex linguistic phenomena, integrate syntax-semantics, and reason in an end-to-end fashion. Take an example from SHRDLU, although it was able to process "find a book which is taller than the one you are holding and put it into the box"[2], it becomes extremely difficult to generalize beyond the specific domain and handle general language, since these systems are based on hand-crafted rules.

Inspired by the rise of statistical learning techniques in the speech recognition community, more and more researchers tended to apply statistical algorithms to NLP problems. It offered a new paradigm: collect examples of the desired input-output pairs and fit them into a statistical model. This new paradigm provides a new way to learn a more general NLP system. Tasks like documentation classification, part-of-speech tagging, and syntactic parsing had made significant improvements. Fields which need more semantic understanding like question answering also achieved much progress.

Statistical machine learning approach requires a labeled dataset of natural language utterances paired with annotated logical forms, as shown in Table 1. Over the last few years, many attempts have been applied to reduce the amount of supervision from annotated logical forms[5,6].

The first is transforming the supervision from annotated logical forms to answers. Take the first question in Table 1 as an example, we use the answer 7 as the ground truth label instead of logical form $max(primes \cap (-\infty, 10))$. Compared with the logical form, this form of data is much easier to obtain view crowd-sourcing, which may result in a more difficult learning problem. But Liang et al.[6] showed a possible way to solve it with modern machine learning algorithms. Learning semantic parsers from answers is called weak supervision.

The second is to apply semantic parsers to more complex domains, e.g., answering questions from broad-coverage knowledge bases such as Freebase[7] and Wikidata[8]. Systems leverage knowledge bases to answer questions are called KBQA system. With the weak supervision technique, it is easy to collect datasets via crowdsourcing, and semantic parsers have even been extended beyond fixed knowledge bases to semi-structured tables[1]. Besides, semantic parsers have been applied to a large number of applications out of KBQA, such as navigation systems[9,10], identifying objects in scene[11,12], and converting natural language to regular expressions[13].

Deep learning demonstrates state-of-the-art performance in many areas[14] including speech recognition[15], image classification[16], image segmentation[17], Part-Of-Speech (POS) tagging[18], Named Entity Recognition (NER)[18–20], and Malware Detection[21]. For example, fully connected neural network is used[22] to effectively identify entities in a newswire corpus. The application of character and word embeddings in Bi-directional Long Short-Term Memory (LSTM)[18,19] achieved state-of-the-art performance in several sequence-to-sequence datasets, such as CoNLL03[23] for NER and Penn Treebank WSJ[24] for POS tagging. In the field of semantic parsing, it is also boosted by the strength of deep learning[25].

We select several representative approaches for semantic parsing and discuss them in detail covering from conventional machine learning to deep learning including both supervised and weak supervised learning. We also provide a taxonomy for these methods shown in Fig. 1.

This paper is structured as follows: we will present a general framework for statistical semantic parsing in Section 2[2]. This framework is gratifying modular and can be easy to be extended for conventional statistical learning algorithms with hand-crafted features (Section 3). Section 4 introduces novel deep learning approaches for semantic parsing. We further discuss the datasets in

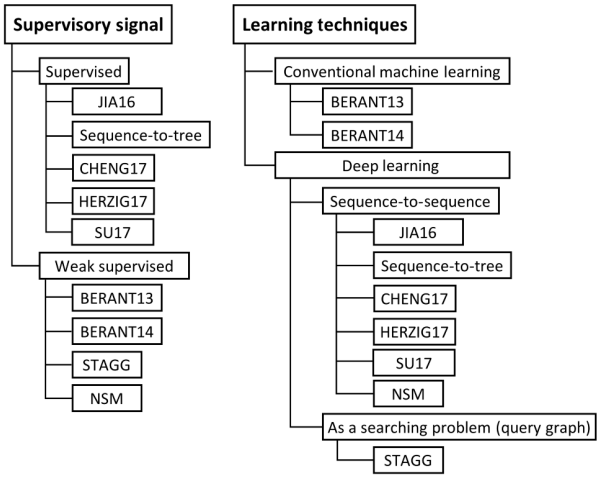


Fig. 1 Semantic parsing taxonomies by supervisory signal and techniques (names are corresponding to Section 4.1).

Section 5 and analyze the challenge and tendency in Section 6. Section 7 concludes the paper.

2 Framework

In this section, we introduce the components of a statistical semantic parsing system. The key of a semantic parser is positing an intermediate logical form d that links utterance x and action y. In particular, d captures the semantics of the utterance x and it also executes to the action y based on the context of c (from a knowledge base or other information source).

Our semantic parsing system consists of five components[2]:

• **Executor.** Compute the action y given a logical form d and context c.

• **Grammar.** Rules G that used to produce the candidates derived from logical forms.

• **Model.** A statistical model to model the distribution $p_\theta(d|x,c)$ parametrized by θ.

• **Parser.** Inferences from the model and get high probability results.

• **Learn.** Tune the parameters θ from the given example pairs.

We now use an example to instantiate each component: "What is the largest prime less than 10?"

Executor. The executor will give us the answer 7.

Grammar. The grammar parses utterances to possible derivations of logical forms. There are plenty of grammar formalisms to represent the semantic meaning[6,26–28]. One is *lambda calculus* with Combinatory Categorial Grammar (CCG)[29]. Below is an example of a CCG derivation[2]:

$$\dfrac{\dfrac{\underline{\textit{prime}} \qquad \underline{\qquad\textit{less than}\qquad} \qquad \underline{10}}{N[\lambda x.prime(x)] \quad (N\backslash N)/NP[\lambda f.\lambda x.f(x)\wedge less(x,y)] \quad NP[10]}\,(>)}{N\backslash N[\lambda f.\lambda x.f(x)\wedge less(x,10)]}\,(<)}{N[\lambda x.prime(x)\wedge less(x,10)]}.$$

A CCG can be defined by lexicons and a set of combinators[30]. The lexicon links words and phrases with their categorial meaning (usually composed of pairs (*words*, *logical constant*)). In CCG trees, each node is a category. Nones (N) denote sets and None Phrases (NP) denote objects, here N and NP are categories in CCG instead of POS tags. It also has composite categories ($(N\backslash N)/NP$), which represent functions of multiple arguments. Rules in CCG are mostly lexical (e.g., $[fun \Rightarrow N[\lambda(x).fun(x)]]$), which map particular words in utterances. Also, we have a forward application and backward application rule:

$$(>) \quad A\backslash B[f] \quad B[x] \quad \Rightarrow A[f(x)],$$
$$(<) \quad B[x] \quad A\backslash B[f] \quad \Rightarrow A[f(x)].$$

CCG can be extended to use other combinations which both handle more complex linguistic phenomena and ungrammatical languages[31]. These issues can also be solved by having a more expansive lexicon[32].

An alternative approach which is less complicated than CCG is to adopt a cruder grammar whose rules correspond directly to the *lambda Dependency-based Composition Semantics* (*lambda DCS*)[33], join and intersect. Here is an example to parse the sentence "prime less than 10" in lambda DCS:

$$\begin{array}{ccc} \textit{prime} & \textit{less than} & 10 \\ N[prime] & N|N[less] & N[10] \\ & N[less.10]_{join} & \\ N[prime \sqcap less.10]_{intersect}. & & \end{array}$$

The join and intersect grammar are

$$(Join) \quad N|N[r]\ N[z] \Rightarrow N[r.z],$$
$$(Intersect) \quad N[z_1]\ N[z_2] \Rightarrow N[z_1 \sqcap z_2].$$

However, these crude rules may result in the derivation of other logical forms such as $10 \sqcap less.prime$. These incorrect logical forms can be figured out by the model.

The choice of grammar is the most important part of a semantic parser because it directly controls the tradeoff between the precision of derived logical forms and statistical model's complexity. CCG is natural to capture the complicated linguistic features from well-structured sentences. For applications with noisy utterance but simple logical forms, a more flexible grammar like lambda DCS is preferred. To answer the question above, we can use CCG to get the logical form $\lambda x.performBy(x, natalie\ potman) \wedge characterIn(x, star\ wars)$.

When answering questions with a knowledge base, another issue is how to map the entities in the utterances to the entities in the knowledge base. This problem is referred as entity linking problem[34]. Although entity linking is not strictly as the same as grammars, it is also a mapping rule. Table 2 shows an example of entity linking result from the question "what character did Natalie Portman play in Star Wars?" A novel entity

Table 2 Examples of entity linking.

Entity in utterance	Entity in knowledge base
NataliePortman	Natalie Portman
Star Wars	Star Wars Episode I: The Phantom Menace

linking system[35] used a tree-based structured learning framework based on multiple additive regression trees to model entity linking as a structured learning problem.

Model. Due to the lexical ambiguity and different parsing decisions, there are many possible logical forms derived from an input utterance. The model produces the scores of candidate logical forms generated by the grammar. Before deep learning, log-linear model (generalizations of logistic regressions) is the common choice for virtually all semantic parsers. The log-linear model defines a feature vector $\phi(x, c, d) \subset \mathbb{R}^F$ for each possible logical form d. We can regard each feature as voting for different derivations d based on some property of the utterance and the derivation.

Let $\theta \subset \mathbb{R}^F$ represent the *parameter vector*, in which every dimension defines a weight for each feature in the feature vector $\phi(x, c, d)$. The score of every generated logical form is $\phi(x, c, d) \cdot \theta$. This score gives an measure how good the logical form is. Normally, we use *softmax* function to obtain the distribution probabilities over all logical forms:

$$p_\theta(d|x, c) = \frac{\exp(\text{score}(x, c, d))}{\sum_{d' \subset D_{x,c}} \exp(\text{score}(x, c, d'))}.$$

Parser. With a trained model p_θ, a semantic parser computes the logical form with the highest probability for an utterance x. Normally, an utterance consists of a sequence of tokens (words). A standard approach is to use a *chart parser*[36], which recursively builds logical forms for each span of the utterance. The parser takes each category and span $[i:j]$ (where $0 \leqslant i < j \leqslant \text{length}(x)$), then loops over all the rules in the grammar to apply each one to build new derivations of each category over $[i:j]$. The final logical forms for the utterance x are collected in the root category over span $[0 : \text{length}(x)]$.

All the derivations generated by Chart Parser could be exponentially large. However, we only care about the derivations with high probability under our model p_θ. Therefore, *beam search* is often applied, where we only keep top-k sub-derivations with top-k scores given by our model at each step. Although beam search is not guaranteed to return the K highest scores, it is still an effective heuristic.

In addition, chart parser suffers from incremental contextual interpretation: the features of a span may only depend on the sub-derivations of that span instead of on the parts constructed before. This problem makes anaphora (resolution of pronoun) challenging to model. A possible solution is to use shift-reduce parsing[37], which scans the utterance from left to right without backing up, so new sub-derivations can depend on the staff before.

Learner. Learner also refers to the optimization problem. The dominant paradigm in machine learning is to set up an objective function and optimize it; while another way is to use reinforcement learning[38]. A standard way is to maximize the likelihood of training data $(x_i, c_i, y_i)_{i=1}^n$. One important thing is that we only have the labeled action y_i rather than the correct logical form d. Under this assumption, we must consider all logical forms d whose action based on context c_i is y_i. The corresponding log-likelihood is

$$\mathcal{O}_i(\theta) = \log \sum_{d \subset D \wedge |d|_{c_i} = y_i} p_\theta(d|x_i, c_i).$$

There are numerous approaches to maximize $\mathcal{O}(\theta)$. Among them, Stochastic Gradient Descent (SGD) is the simplest way, which is an iterative algorithm that updates the parameters θ over the training data.

$$\theta \leftarrow \theta + \eta \bigtriangledown \mathcal{O}_i(\theta),$$

where η is the step size, known as learning rate that decides how aggressively we want to update parameters with the gradients. Due to the concavity of the objection function $\mathcal{O}(\theta)$, SGD is best guaranteed to converge to a local maxim instead of a global one.

Until now, we have covered the components of a semantic parser. In next sections, we will focus on the learning problem, especially on the model and learner part of the whole system.

3 Semantic Parsing with Hand-Crafted Features

3.1 Supervised semantic parsing with CCG

CCG has two key components, the lexicon and combinators. Combinators are a small set of operations that are predefined, while lexicon pair words and phrases with CCG categories. So we have the structure learning problem, where we need to learn the structure of the CCG grammar. Also, the parsing process is often ambiguous, so we need another model to choose the best parse giving these ambiguities. To solve these two problems jointly, we will introduce a simple algorithm named structured perceptron and then an online learning algorithm derived from it. Many supervised semantic parsing algorithms share the same idea so that we will introduce a basic algorithm.

3.1.1 Structured perceptron

A straightforward algorithm of machine learning,

structured perceptron was proposed in Ref. [39]. In this part, we first introduce the generic perceptron[40]. After that, we show how structured perceptron works.

Generic perceptron[40] uses a linear prediction function to do binary classification. More formally, the prediction function is

$$f(x) = \begin{cases} 1, & \text{if } wx + b > 0; \\ 0, & \text{otherwise,} \end{cases}$$

where x is the input vector, w is the weight vector, and b is the bias.

As Algorithm 1 shown, perceptron tunes the weight vector in an online fashion. For each example, perceptron checks whether it is classified correctly. If not, it will update the weight vector by moving w towards the current input vector. The perceptron algorithm is guaranteed to accurately classify each example in a linear separable training set[40].

Reference [41] proposed averaged perceptron that assigns more weights for the examples at the beginning of the training. With this weight averaging, perceptron achieves some kind of *large margin* effect. Binary perceptron can be easily extended to multiclass classification. In multiclass setting, for each input x, the corresponding label y belongs to a finite set $\mathcal{Y}$ and simply chooses the maximum score.

Similar to the multiclass setting, structured perceptron[39] is an extension of the perceptron algorithm. The difference is that structured perceptron deals with the labels which represent a set of structures. These structures can be generated from a given structured input x[42]. The prediction function of the structured perceptron is

$$y = \text{argmax}_{y' \subset \mathcal{Y}(x)} \Phi(x, y') \cdot w,$$

where $\Phi(x, y')$ maps the pair (x, y') to a vector (features). Algorithm 2 shows the learning algorithm for structured perceptron.

Structured perceptron can be connected with a log-linear model naturally. Recall the log-linear model, we define a condition probability on x and y:

$$p(x|y) = \frac{\exp(w \cdot \Phi(x, y))}{\sum_{y'} \exp(w \cdot \Phi(x, y'))}.$$

To maximize the log-likelihood function, we can use gradient descent to solve this optimization problem as follows:

$$\mathcal{L} = \sum_{i=1}^{n} \log p(y^i|x^i),$$

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}w} = \sum_{i=1}^{n} \Phi(x^i, y^i) - E_{p(y|x)} \Phi(x^i, y).$$

When using SGD and Viterbi approximation, we get

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}w} = \Phi(x^i, y^i) - \Phi(x^i, y^*),$$

$$y^* = \text{argmax}_y w \cdot f(x^i, y).$$

To make the model more powerful, we can add latent variables into the log-linear model ($p(y|x) = \sum_h p(y, h|x)$)[31,43]. When applying SGD to optimize the log-likelihood with Viterbi approximation, we have

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}w} = \Phi(x^i, h', y^i) - \Phi(x^i, h^*, y^*),$$

$$h' = \text{argmax}_h w \cdot f(x^i, h, y^i),$$

$$y^*, h^* = \text{argmax}_{y,h} w \cdot f(x^i, h, y^i).$$

Unfortunately, the hidden variable perceptron is not guaranteed to converge due to the log-linear version is non-convex. It is still a good choice because this algorithm is very simple and easy to implement, and works well with careful initialization.

Algorithm 1 Learning for generic perceptron

Input:
Training set $\mathcal{D} = \{(x^1, y^1), (x^2, y^2), \ldots, (x^N, y^N)\}$, number of iterations I.
Output:
Weight vector w.

$w \leftarrow (0, 0, \ldots, 0)$
for $i = 1, 2, \ldots, I$ **do**
  for $j = 1, 2, \ldots, N$ **do**
    if $y^j x^j w \leqslant 0$ **then**
      $w \leftarrow w + y^j x^j$
    end if
  end for
end for

Algorithm 2 Learning for structured perceptron

Input:
Training set $\mathcal{D} = \{(x^1, y^1), (x^2, y^2), \ldots, (x^N, y^N)\}$, map function Φ.
Output:
Weight vector w.

$w \leftarrow (0, 0, \ldots, 0)$
repeat
  for each $(x, y) \subset \mathcal{D}$ **do**
    $z \leftarrow \text{argmax}_{y' \subset \mathcal{Y}(x)} \Phi(x, y') \cdot w$
    if $z \neq y$ **then**
      $w \leftarrow w + \Phi(x, y) - \Phi(x, z)$
    end if
  end for
until Converged

3.1.2 A unified learning algorithm

In this subsection, we investigate a unified learning algorithm[28] to induce semantic parsing which is related to loss-sensitive structured perceptron[44]. This algorithm can be applied to both supervised learning and weak supervision. It jointly estimates the parsing parameters and induces the lexicon structure.

In the unified learning algorithm, there are two learning choices to be made. The first is

$$\mathcal{V} : \mathcal{Y} \Rightarrow (t, f),$$

where the validation function $\mathcal{V}$ indicates the correctness (true of false) of a parse y. The varying function $\mathcal{V}$ allows us to use different forms of supervision. The second choice need to specify the lexical generation procedure (GENLEX[28]), which takes input (*sentence:* x, *validation function:* $\mathcal{V}$ *or labeled logical form* z, *lexicon:* Λ, *parameters:* θ) and produces a overly general set of lexical entries. The generated set is noisy and needs to be pruned later with correct lexical entries. The overall algorithm is shown as Algorithm 3.

With the online learning schema, we have two steps for each training sample. One is *lexical generation* and the other is *update parameters*.

One of the advantages of this algorithm is that it can provide a relatively compact lexicon, which is a subset of possible lexical entries. This can be achieved by alternating between two steps. **Step 1** is aimed to search for a relatively small number of lexical entries, which are sufficient to parse all training samples successfully. **Step 2** is used to update the parameters of the lexical entries that are selected in **Step 1**.

In Algorithm 4, each utterance in one training sample is parsed with the current parameters θ with a new lexicon $\Lambda_0 \cup \text{GENLEX}(x^i)$. This results in a large set of result logical forms. However, correct lexical entries are expected in those parse results with the highest scores. So we extract the top-k parses and combine them with the original set.

Algorithm 3 Unified learning algorithm

Input:
Training set $(x^i, \mathcal{V}^i) : i = 1, \ldots, n$, number of iterations T.
Output:
Parameters θ and lexicon λ.
1: Initialize θ using Λ_0, $\Lambda \leftarrow \Lambda_0$
2: **for** $i = 1, 2, \ldots, T$ **do**
3: **for** $j = 1, 2, \ldots, n$ **do**
4: Step 1: Lexical generation
5: Step 2: Update parameters
6: **end for**
7: **end for**

Algorithm 4 Lexical generation

Input:
Training sample (x^i, L^i), lexicon Λ_0, and parameters θ.
Output:
Lexicon Λ_0.
1: Set $\lambda = \Lambda_0 \cup \text{GENLEX}(x^i, L^i, \Lambda_0, \theta)$
2: Parse x^i with model θ and lexicon λ and get k highest scoring parses Y
3: $\Lambda_t = \Lambda_0 \cup \bigcup_{y \subset Y} \text{LEX}(Y)$

We can apply structured perceptron as discussed in the last subsection to update the parameters θ based on the parsing results and lexicon.

Cycling between **Steps 1 and 2** will keep only the lexical entries that occur in the highest scoring parses, which leads to producing a compact lexicon. In summary, the unified learning algorithm forms a greedy, iterative method for simultaneously finding a compact lexicon and optimizing the log-likelihood of the training data[28].

3.2 Weak supervised semantic parsing

In the supervised learning fashion, we have the labeled logical forms, e.g., "show me flights to Boston ($\lambda x.flight(x) \wedge to(x, BOSTON)$)" and "I need a flight from Baltimore to Seattle ($\lambda(x).flight(x) \wedge from(x, BALTIMORE) \wedge to(x, SEATTLE)$)". In this form of learning, we learn directly from pairs of utterances and logical forms[28]. However, it is hard to label the logical forms when given thousands of utterances. Another problem is that one utterance may correspond to several correct logical forms. A possible way to learn flexibly is *weak supervision*[5,6], which only requires executing logical forms within a system and evaluating the result[45,46]. In this form, both parsing process and logical forms are latent values. Also, it uses the executor directly to evaluate the parser, which can be easily applied to different domains, and it does not suffer from the labeling problem from expertise.

In this section, we will focus on KBQA[47,48], one of the most exciting problems in weak supervising semantic parsing. A *knowledge base* $\mathcal{K}$ is a set of *assertions* $(e1, p, e2)$, where e denotes an *entity* (e.g., BarackObama) and p is a *property* (e.g., BirthDate) in $\mathcal{K}$. There are two challenges for KBQA: (1) knowledge base has a very large set of logical predicates, which is

impossible to store them in the parser, and (2) without labeled logical forms, we cannot generate sufficient lexical entries as in Section 3. An example of KBQA is shown in Fig. 2. Most work of KBQA is based on Freebase[7].

3.2.1 Berant13

Berant13[49] used a large amount of web text and a knowledge base to build a coarse alignment between phrases and predicates. However, this alignment cannot cover all the predicates. Light verbs (e.g., "go") and prepositions are hard due to polysemy. Rare predicates (e.g., "cover price") are difficult even given a large corpus. To improve coverage, Berant13[49] proposed a new *bridging* operation that generates predicates based on adjacent predicates rather than on words. Based on these features, Berant13 used a log-linear model to score the logical forms.

The goal of alignment was to construct a lexicon $\mathcal{L}$, a mapping from natural language phrases to logical predicates accompanied by a set of features. Intuitively, Berant13 aligned a phrase and a predicate based on whether they co-occur with many of the same entities. In summary, Berant13 first constructed a set of typed (Freebase associates each entity with a set of types using the type property) phrases $\mathcal{R}_1$ (e.g., "born in" [Person, Location]) and predicates $\mathcal{R}_2$ (e.g., PlaceOfBirth). For each element $r \subset \mathcal{R}_1 \cup \mathcal{R}_2$, they created a mapping $\mathcal{F}(r)$ from a phrase in $\mathcal{R}_1$ to related predicates $\mathcal{R}_2$ and vice versa (e.g., $\mathcal{F}$ ("*bornin*" [*Person*, *Location*]) $=$ {(*BarackObama*, *Honolulu*), ...}). The lexicon is generated based on the overlap $\mathcal{F}_{r1} \cup \mathcal{F}_{r2}$, for $r1 \subset \mathcal{R}_1$ and $r2 \subset \mathcal{R}_2$. With the alignment, Berant13 computed three types of features. *Alignment* features: (1) log of number of entity pairs that occur with the phrase $r1$ ($|\mathcal{F}(r1)|$), and (2) log of number of entity pairs that occur with the phrase $r2$ ($|\mathcal{F}(r2)|$), and (3) log of number of entity pairs that occur with both $r1$ and $r2$ ($|\mathcal{F}(r1) \cup \mathcal{F}(r2)|$). *Lexicalized* features: conjunction of phrase and predicate. *Text similarity* features: phrase $r1$ is equal/prefix/suffix of $r2$ or $r1$ and $r2$ overlap.

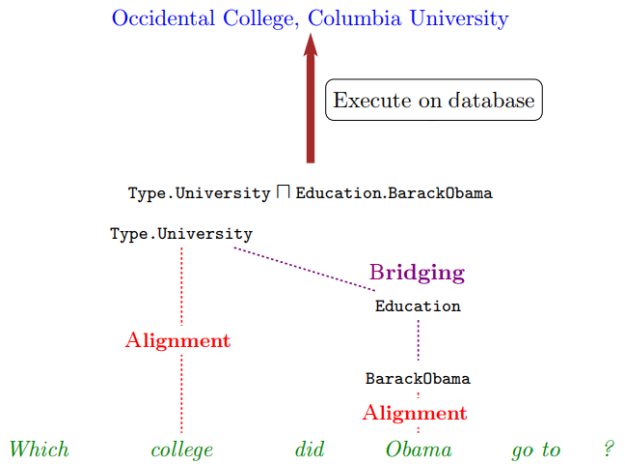


Fig. 2 An example to answer a question through a knowledge base[49]. To narrow down the space of logical predicates, they use (1) course alignment based on Freebase and a text corpus and (2) a bridging operation that generates predicates compatible with neighboring predicates.

The *bridging* operation generates a binary predicate based on neighboring logical predicates instead of explicit lexical material. Considering the question *"what is the cover price of X-men?"*, the correct prediction *ComicBookCoverPrice* is expressed explicitly but is not in the lexicon generated by the alignment. In this situation, given a unary z with type t (X-men), they constructed a logical form $b.z$ for any predicate b with type (*, t). Based on this operation, Berant13 computed the *bridging feature*: log of the number of pairs that occur with bridging predicate b.

Together with the features from alignment, bridging, and other syntax based features, Berant13 used a log-linear model to rank the generated logical forms.

3.2.2 Berant14

One problem of conventional semantic parsers is that they only use data which pairs natural language with the KB. This leaves untapped a vast amount of text not related to the KB. Berant14[50] presented an approach for semantic parsing based on paraphrasing that was able to exploit large amounts of text not covered by the KB (Fig. 3).

This approach consists of three steps: (1) use a

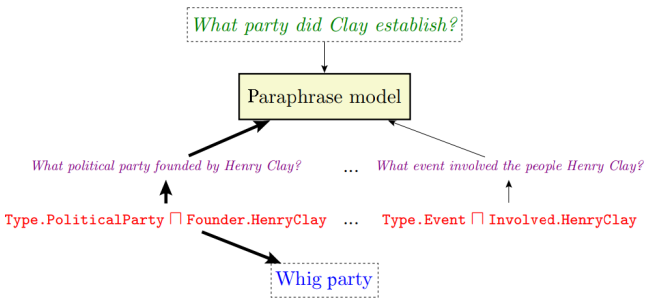


Fig. 3 An example of paraphrasing for semantic parsing[50]. For each candidate logical form (red), it generated canonical utterances (purple). The model is trained to paraphrase the input utterance (green) into the canonical utterances associated with the correct denotation (blue).

simple procedure to construct a manageable set of candidate logical forms based on the input utterance, (2) leverage the generated logical forms to generate canonical utterances based on the text descriptions of entities and predicates from the KB, and (3) choose the generated utterance that best paraphrases and input together with the corresponding logical form.

Before generating logical forms when given utterances, Berant14 defined a set of templates to cover: (1) $p.e$, (2) $p1.pe.2$, (3) $p.(p1.e1 \cup p2.e2)$, (4) support "unary filter", and (5) handle aggregation formulas. From an utterance x, Berant14 found an entity in x and grow the logical form from it, details can be referred in the original paper[50]. With the same strategy, Berant14 also used some rules and templates to generate utterances from the logical forms (e.g., "*What NP is VP by* $d(e)$?", where $d(e)$ is the Freebase descriptions for the entity).

When both logical forms paired with utterances were constructed, the problem was reduced to scoring pairs (c, z) based on a paraphrase model. Berant14 combined traditional association model (determine whether x and c contain phrases that are likely to be paraphrases) and a vector space model (based on word2vec embedding[51]):

$$\phi_{pr}(x,c)^{\mathrm{T}}\theta_{pr} = \phi_{as}(x,c)^{\mathrm{T}}\theta_{as} + \phi_{vs}(x,c)^{\mathrm{T}}\theta_{vs}.$$

The association model links phrases of x and c in multiple overlapping ways with features: (1) lemma $(x_{i:j}) \wedge \text{lemma}(c_{i':j'})$, where $x_{i:j}$ denotes spans from x, (2) $\text{pos}(x_{i:j}) \wedge \text{pos}(c_{i':j'})$, (3) $\text{lemma}(x_{i:j}) = \text{lemma}(c_{i':j'})$?, (4) $\text{pos}(x_{i:j}) = \text{pos}(c_{i':j'})$?, (5) $\text{lemma}(x_{i:j})$ and $\text{lemma}(c_{i':j'})$ are synonyms?, and (6) $\text{lemma}(x_{i:j})$ and $\text{lemma}(c_{i':j'})$ are derivations? ("?" will return true or false of the association model).

In vector space model, Berant14 estimated a paraphrase score via a weighted combination:

$$v_x^{\mathrm{T}} W_{v_c} = \sum_{i,j=1}^{k} w_{ij} v_{x,i} v_{c,j},$$

where $W \subset \mathbb{R}^{k \times k}$ is a parameter matrix. The association model aligned particular phrases to one another, while the vector space model provided a soft representation for utterances.

4 Deep Learning for Semantic Parsing

In this section, we will first give an overview of deep learning, and then focus on the deep learning based semantic parsing algorithms for both supervised and weak supervised learning.

4.1 Overview of deep learning

Deep learning allows computational models to learn representations of data with multiple levels of abstraction through multiple processing layers[14]. These deep learning based methods have dramatically improved the state-of-the-art performance in speech recognition, visual object detection and recognition, language models, and many other fields such as drug discovery and genomics. In this subsection, we will first introduce the feedforward neural network, followed the architecture of Convolution Neural Network (CNN) which has brought breakthroughs in processing images. Then we will discuss Recurrent Neural Network (RNN) and some improvement upon it which has shone a light on sequential data such as text and speech.

4.1.1 Feedforward neural network

Feedforward neural network, which is also known as MultiLayer Perceptron (MLP), is aimed to approximate some function f^*, e.g., for a classifier, $y = f^*(x)$ maps an input x to a category y. To enhance the expressive ability of MLP, we can add non-linear activation function (e.g., *sigmoid* and *ReLU*) after each hidden unit. To train the MLP, we can apply backpropagation with chain-rule. In summary, feedforward neural networks are the quintessential deep learning methods.

4.1.2 CNN

CNN[52] is a specialized kind of neural network for processing data with a known grid-like topology (e.g., time-series data, which can be treated as a 1D grid and image data, which is a 2D grid of pixels). As the name implies, CNN employs a mathematical operation named *convolution*. A convolution layer uses a set of kernels to attain local features over the whole image sharing the same parameters, which means that one channel of the output is a result calculated by one kernel interacts with the whole image. This results in that CNN has fewer parameters than MLP.

A typical layer of CNN consists of three stages: a convolution layer, an activation layer, and a pooling layer. The activation layer performs the same role as in MLPs to provide a nonlinear transformation. The pooling layer applies a pooling function which replaces the output of the net at a certain location with a

summary statistic of the nearby outputs[53]. One popular pooling function is *max pooling*[54], which reports the maximum output within the rectangular neighborhood. Pooling layer helps to make the representation approximately invariant to small translations of the input. Another purpose of pooling is to reduce the dimension of the output and save memory.

4.1.3 RNN

Unlike CNN which is good at processing spatial data like images or speech, RNNs are a family of neural networks for processing sequential data. An RNN deals with a sequence of input $x^1, x^2, \ldots, x^t$ (t is the total number of time steps). The structure of an RNN is based on MLPs with the idea that sharing parameters across different parts of a model and inputs, which makes it possible to generalize across them[53]. RNNs usually produce an output at each time step with connections between last hidden units and current input; and an RNN cell consists of these parts:

$$\begin{aligned} h' &= b + Wh^{t-1} + Ux^t, \\ h^t &= \tanh(h'), \\ o^t &= c + Vh^t, \end{aligned}$$

where b and c are the bias vectors along with the weight matrices U, V, and W, for input-to-hidden, hidden-to-output, and hidden-to-hidden connections, respectively. RNNs capture information only from the past and the present input instead of the whole sequence. In many applications, there is a need to output a prediction that depends on the whole input sequence. Bidirectional RNNs were invented to address the need[55]. Bidirectional RNNs use two separate RNNs to deal with the sequence in two directions (forward and backward) and combine the two outputs together as the final representation.

RNNs suffer from the *gradient vanishing or exploding* problem, so it is hard to learn long-term dependencies. When gradients back propagated over many stages, they tend to either vanish (most of the time[56]) or explode (rarely[57]). An improvement of the basic RNNs is called *gated RNNs* including LSTM[58] and *Gated Recurrent Unit* (GRU[59]). The general idea of gated RNNs is to create paths through time that derivatives neither vanish nor explode.

LSTM leverages a self-loop mechanism to produce paths where the gradient can flow for long durations[58]. An LSTM cell is composed of three multiplicative gates which control the proportions of information to flow to the next time step. Formally, the formulas to update an LSTM cell at time step t are

$$\begin{aligned} i^t &= \sigma(W_i h^{t-1} + U_i x^t + b_i), \\ f^t &= \sigma(W_f h^{t-1} + U_f x^t + b_f), \\ \hat{c}^t &= \tanh(W_c h^{t-1} + U_c x^t + b_c), \\ c^t &= f^t \odot c^{t-1} + i^t \odot \hat{c}^t, \\ o^t &= \sigma(W_o h^{t-1} + U_o x^t + b_o), \\ h^t &= o^t \odot \tanh(c^t), \end{aligned}$$

where σ is the element-wise sigmoid function and $\odot$ is the element-wise product. x^t is the input vector in time step t, and h^t is the hidden state vector. U_i, U_f, U_c, and U_o are the weight matrices of different gates for input, and W_i, W_f, W_c, and W_o are the weight matrices for hidden state. b_i, b_f, b_c, and b_o denote the bias vectors.

Another popular gated RNN is GRU[59], which combines the forget and input gates into one named "update" gate. Besides, it also merges the cell state and hidden state. The update at time step t is

$$\begin{aligned} z^t &= \sigma(W_z[h^{t-1}, x^t]), \\ r^t &= \sigma(W_r[h^{t-1}, x^t]), \\ \hat{h}^t &= \tanh(W_h[r^t \odot h^{t-1}, x^t]), \\ h^t &= (1 - z^t) \odot h^{t-1} + z^t \odot \hat{h}^t. \end{aligned}$$

From the fomulas, we can get that GRU uses a single gating unit simultaneously to control the forgetting factor and the decision to update the hidden state. This results in that fewer parameters needed in a GRU cell.

4.2 Deep learning for semantic parsing

With a basic understanding of different neural nets, we will introduce how the deep learning combined with semantic parsing framework. We will give several elegant algorithms to combine deep learning and semantic parsing: (1) transform the logical form into a query graph, (2) use a sequence-to-sequence model with data recombination to learn logical form directly, (3) use a sequence-to-sequence model and utilize a key-variable memory to handle compositionality with REINFORCE, and (4) a neural semantic parser firstly maps the utterances to an intermediate representation and then induces intermediate representations in the form of predicate-argument structures from data.

4.2.1 STAGG[60]

Inspired by Refs. [61, 62], STAGG[60] proposed a semantic parsing that leverages the knowledge base tightly forming the parse for an input question. First, STAGG defined a query graph that is straightforwardly mapped to a logical form in λ-*calculus*. In this way, they

reduced semantic parsing to query graph generation, which can be formulated as a search problem with staged states and actions. Each state is a candidate parse in the graph representation and each action defines a way to grow the graph. In particular, STAGG staged into three main steps: (1) locate the *topic entity* in the question, (2) find the main *relationship* between the answer and the topic entity, and (3) expand the query graph with additional *constrains* that describe properties the answer needs or the relations between the answer and other entities.

Before we describe the query graph, we first introduce a particular entity category named Compound Value Type (CVT) in Freebase. A CVT node is not a real-world entity but is used to collect multiple fields of an event or a special relationship.

Their query graph consists of four types of nodes: *grounded entity* (rounded rectangle), *existential variable* (circle), *lambda variable* (shaded circle), and *aggregation function* (diamond). Figure 4 shows one possible query graph for the question "who first voiced Meg on Family Guy?" using Freebase. *Meg Griffin* and *Family Guy* are two entities represented by two rounded rectangles. y in a circle is an entity that should exist an entity describing some casting relations (e.g., character, actor), where y should be a CVT entity in this case. The shaded circle x is the answer node, which is the answer to the question. The diamond node argmin constraints that the answer needs to be the earliest actor. This query graph is equivalent as the logical form $\lambda x.\exists y.cast(FamilyGuy, y) \wedge actor(y, x) \wedge character(y, MegGriffin)$ without the aggregation function.

With this new query graph, STAGG generated the graphs with the following properties: (1) there is one root entity referred to the *topic entity* in the graph and (2) there exists only one lambda variable x as the answer node, with a directed path (named *core inference chain*) from the root to it.

Given a question, STAGG formalized the query generation process as a search problem with staged states and actions. Let $\mathcal{S} = \cup\{\varnothing, \mathcal{S}_e, \mathcal{S}_p, \mathcal{S}_c\}$ be the set of states, where each state can be an empty graph,

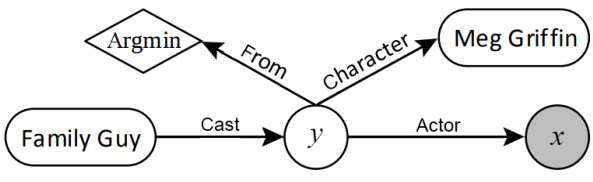


Fig. 4 Query graph of question "who first voiced Meg on Family Guy"[60].

a single node with the topic entity, a core inferential chain, or a more complex query graph with constraints. Let $\mathcal{A} = \cup\{\mathcal{A}_e, \mathcal{A}_p, \mathcal{A}_c, \mathcal{A}_a\}$ be the set of actions. An action can grow a graph by adding some edges and nodes. $\mathcal{A}_e$ picks an entity node; $\mathcal{A}_p$ determines the core inferential chain; and $\mathcal{A}_a$ and $\mathcal{A}_c$ add constraints and aggregation nodes, respectively. A valid action set can be defined in Fig. 5. The order of actions can vary and be viewed as different ways to prune the search space. STAGG defined a reward function using a log-linear model and search is done using the best-first strategy with a priority queue. The three stages are described as follows:

(1) The topic entities are chosen from an entity linking system[35], and to tolerate potential mistakes of the entity linking systems, up to 10 top ranked entities are considered as the topic entities.

(2) To identify core inferential chain, STAGG measured the semantic similarity using CNN between the query and the candidate chain. They proposed Siamese neural networks[63] for identifying the core inferential chain. In this way, STAGG mapped the question to a pattern by replacing the entity mention with a generic symbol $\langle e \rangle$, such as "who first voiced meg on $\langle e \rangle$" vs. cast-actor.

(3) One simple way to derive the constraint set is first issuing the core inferential chain as a query to the KB to find the bindings of variables y' and x, and then enumerating all neighboring nodes of these entities. This often results in an unnecessarily large pool. STAGG employed simple rules to retain only the nodes that have some possibility to be constraints.

To obtain all possible query graphs, STAGG used a log-linear model to learn a reward function based on each stage's reward.

4.2.2 JIA16[64]

STAGG[60] reduced the semantic parsing into a searching problem, this method treated it as a sequence-to-sequence problem (learn logical form from a supervised fashion). Meanwhile, JIA16[64] introduced data recombination, a framework for injecting prior knowledge into a model.

In a sequence-to-sequence model (consists of two RNNs, an encoder and a decoder), the input utterance

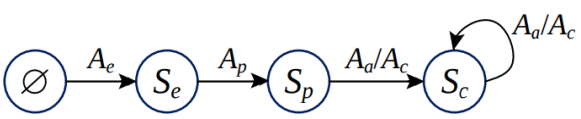


Fig. 5 Legitimate actions to grow a query graph[60].

x is a sequence of words $x_1, \ldots, x_m$ from the input vocabulary; the output is the corresponding logical form y, which is also a sequence of tokens $y_1, \ldots, y_n$ from the output vocabulary. This kind of technique is popular in machine translation[65,66]. One problem of the conventional sequence-to-sequence model is that it has difficulty generalizing to the long tail of entity names commonly found in semantic parsing datasets. To overcome this problem, JIA16 used a copying mechanism based on a Pointer Network[67] to decide whether to write any word in the vocabulary or to copy a word from any input word directly to the output.

Due to the limitation of training data, JIA16 used a data recombination framework to inject prior knowledge and generated new examples to train the sequence-to-sequence model inspired by the data augmentation technique, which is commonly employed in computer vision[68] and speech recognition[69]. The key advantage of this approach is that it allows declaring desired properties for the specific task. In semantic parsing, consider an example "what states border texas?", it should be easy to generalize to questions where "*texas*" is replaced by other state and simply replace the mention of *Texas* in the logical form with the new one.

In general, JIA16 started with a training set $\mathcal{D}$ of (x, y) pairs, which defines the empirical distribution $\hat{p}(x, y)$. Then they fit a generative model $\tilde{p}(x, y)$ to $\hat{p}$ which generalizes beyond the support of $\hat{p}$. Finally, to train the model $p_\theta(y|x)$, they maximized the expected value of $\log p_\theta(y|x)$. JIA16 applied a Synchronous Context-Free Grammar (SCFG) to generate the samples, readers who are interested in the grammar can check the paper[64].

4.2.3 Sequence-to-tree[70]

Sequence-to-sequence model ignores the hierarchical structure of logical forms. To generate well-formed output, it needs to memorize various pieces of auxiliary information (e.g., bracket pairs). Reference [70] presented a hierarchical tree decoder to overcome this problem. The tree decoder generated logical forms in a top-down manner. In order to represent the tree structure, sequence-to-tree[70] defined several special token, e.g., "nonterminal" $\langle n \rangle$ token which indicates subtrees, $\langle s \rangle$ and $\langle (\rangle$ represent the beginning of a sequence and nonterminal sequence, respectively, and $\langle / s \rangle$ is the end of sequence.

After encoding input q, the tree decoder used RNNs to generate the tokens at depth 1 of the subtree. When the predicted token is $\langle n \rangle$, sequence-to-tree decoded the sequence by conditioning on the hidden vector. They introduced a *parent-feeding* connection where the hidden vector of parent nonterminal is concatenated with the inputs and fed into RNN to generate subtrees. Figure 6 shows the decoding tree as an example of the logical form "$AB(C)$", where $y_1, \ldots, y_6$ are predicted tokens, and $t_1, \ldots, t_6$ are time steps. The decoding procedure is as follows: (1) once input has been encoded, the decoder first generates $y_1, \ldots, y_4$ at depth 1 until token $\langle / s \rangle$; (2) use the nonterminal token t_3's hidden vector together with the input to predict the subtree.

After training, sequence-to-tree predicted the logical form for utterance q by

$$\hat{g} = \text{argmax}_{g'} p(g'|q),$$

where g' represents a candidate result. In practice, it was impossible to iterate over all possible candidates to get the optimal output. Sequence-to-tree decomposed the probability $p(g|q)$ and used greedy search (or beam search) to generate tokens one by one. For *parent-feeding*, sequence-to-tree maintained a nonterminal queue to generate the subtrees. Sequence-to-tree also claimed that attention mechanism boosted the performance in all benchmarks they used. In addition, sequence-to-tree developed a simple procedure for data argumentation. They replaced the identify entities and numbers with the type names and unique IDs. Experiments showed the argumentation also improved the performance.

4.2.4 Neural symbolic machines[71]

Neural Symbolic Machine (NSM) contains (1) a sequence-to-sequence model that maps language utterances to programs and utilizes a *key-varivable memory* to handle compositionality and (2) a symbolic "computer", an executor that performs execution

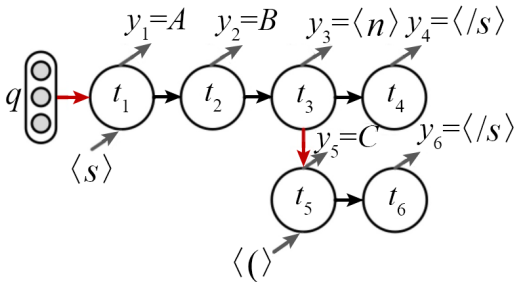


Fig. 6 A SEQ2TREE decoding example for the logical form "*AB*(*C*)"[70].

and helps find good programs by pruning the search space. To train this machine end-to-end, NSMs[71] applied REINFORCE to directly optimize the task reward of this problem. The advantages of this approach are as follows: (1) save and reuse intermediate executions results with the combination of a sequence-to-sequence model and a *key-value memory*, (2) use the computer to execute partial programs and prune the models' search space, and (3) combine REINFORCE with an *iterative maximum likelihood* training process.

Unlike other approaches, NSMs adopted a Lisp interpreter as the "executor" and defined a subset of Lisp programming which is only equivalent to the subset of λ-calculus presented in STAGG[60] as shown in Fig. 7. With the executor, it can help the model to produce a list of valid tokens. First, a valid token should not cause a syntax error, e.g., if the previous token is "(", the next token must be a function name. Second, a valid token should not cause a semantic error, e.g., if the previously generated tokens were "*Hop r*", the next available token is restricted to predicates that are reachable from entities in r.

With the executor, the sequence-to-sequence aims to map natural language into a program, the basic structure is a standard seq2seq model with an attention which is similar with JIA16[64], but extend it with a *key-value* memory that allows the model to learn to represent and refer to program variables. To let the decoder learn to represent and refer to intermediate

$$
\begin{gathered}
(\ Hop\ r\ p\ ) \Rightarrow \{e_2 | e_1 \in r, (e_1, p, e_2) \in \mathbb{K}\} \\
(\ ArgMax\ r\ p\ ) \Rightarrow \{e_1 | e_1 \in r, \exists e_2 \in \mathcal{E} : (e_1, p, e_2) \in \mathbb{K}, \forall e : (e_1, p, e) \in \mathbb{K}, e_2 \geqslant e\} \\
(\ ArgMin\ r\ p\ ) \Rightarrow \{e_1 | e_1 \in r, \exists e_2 \in \mathcal{E} : (e_1, p, e_2) \in \mathbb{K}, \forall e : (e_1, p, e) \in \mathbb{K}, e_2 \leqslant e\} \\
(\ Filter\ r_1\ r_2\ p\ ) \Rightarrow \{e_1 | e_1 \in r_1, \exists e_2 \in r_2 : (e_1, p, e_2) \in \mathbb{K}\}
\end{gathered}
$$

Fig. 7 Interpreter functions of the NSM. r represents a variable, p is a predicate in Freebase. $\leqslant$ and $\geqslant$ are defined on numbers and dates[71]. $\mathbb{K}$ is the knowledge base and $\mathcal{E}$ denotes a set of entities.

variables whose value was saved in the machine after execution, NSMs augmented the seq2seq model with a key-variable memory, where each entry has two components: a continuous embedding key v_i and a corresponding variable token R_i referencing to the value in the machine (Fig. 8). During encoding, NSMs used an entity linker to link text spans (e.g., "US") to KB entities. For each linked entity, NSMs added a memory entry where the key is the average of GRU hidden states over the text span, and the variable token (R_1) is the name of a variable in the machine holding the linked entity as its value. During decoding, when a full sequence is generated, the result is stored as the value of a new variable in the machine (keyed by the hidden state at this step). When a new variable R_1 with key embedding v_1 is added into the memory, the token R_1 is added into the decoder vocabulary with v_1 as its embedding.

NSM executes non-differentiable operations against a KB, and thus end-to-end backpropagation is not possible. Therefore, NSMs used REINFORCE[72] with some full supervision pre-train. To train from weak supervision, NSMs proposed an iterative machine learning algorithm (similar to hard Expectation-Maximization (EM)), where they searched for good programs given fixed parameters, and then optimized the probability of the best program found so far. To do this, NSMs optimized the log-likelyhood objective function:

$$\mathcal{J}_\theta = \sum_x \log P_\theta(A_{\text{best}}(x)|x),$$

where A_{best} is the pseudo-gold program in the current setting. Algorithm 5 summaries the overall training schema. For pre-training, NSMs first ran iterative ML for N_{ML} iterations and recorded the best program found for every input x_i. Then NSMs ran REINFORCE,

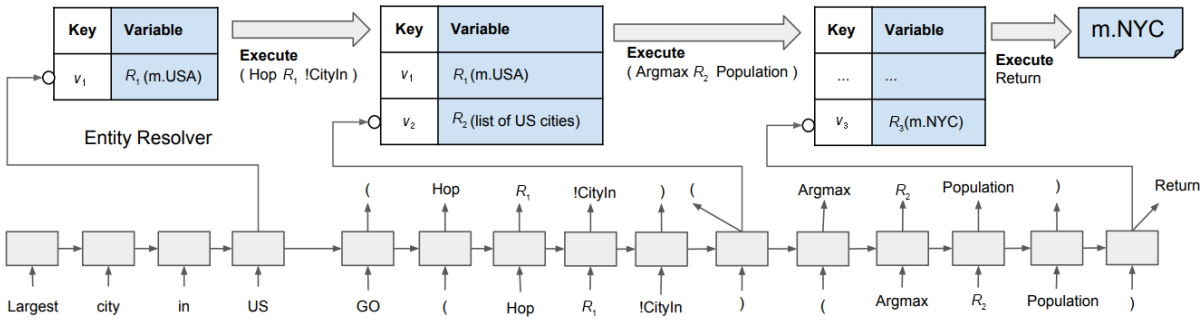


Fig. 8 Semantic parsing with NSM. A special token "GO" indicates the start of decoding, and "Return" indicates the end of decoding. Due to the fact that the decoding model never sees the values in the encoder ("US") here, so it only references them with the name of the variable ("R_1"). The memory bridges these two steps to achieve compositionality[71].

Algorithm 5 IML-REINFORCE

Input:

Quenstion-answer pairs $\mathcal{D} = (x_i, y_i)$, mix ratio α, reward function $R(\cdot)$, training iterations N_{ML}, N_{RL}, and beam sizes B_{ML}, B_{RL}.

Procedure:

Initialize $C_x^* = \varnothing$ the best program so far for x and model θ.

for $n = 1, \ldots, N_{\text{ML}}$ **do**
  for (x, y) in $\mathcal{D}$ **do**
    $\mathcal{C} \leftarrow$ Decode B_{ML} programs given x
    for $j = 1, \ldots, |\mathcal{C}|$ **do**
      if $R_{x,y}(C_j) > R_{x,y}(C_x^*)$ **then**
        $C_x^* \leftarrow C_j$
      end if
    end for
  end for
  $\theta \leftarrow$ ML training with $\mathcal{D}_{\text{ML}} = (x, C_x^*)$
end for
Initialize model for REINFORCE
for $n = 1, \ldots, N_{\text{RL}}$ **do**
  $\mathcal{D}_{\text{RL}} \leftarrow \varnothing$ is the RL training set
  for (x, y) in $\mathcal{D}$ **do**
    $\mathcal{C} \leftarrow$ Decode B_{RL} programs given x
    for $j = 1, \ldots, |\mathcal{C}|$ **do**
      if $R_{x,y}(C_j) > R_{x,y}(C_x^*)$ **then**
        $C_x^* \leftarrow C_j$
      end if
    end for
    for $j = 1, \ldots, |\mathcal{C}|$ **do**
      $\hat{p}_j \leftarrow (1-\alpha) \cdot \frac{p_j}{\sum_{j'} p_{j'}}$, where $p_j = P_\theta(C_j|x)$
      if $C_j = C_x^*$ **then**
        $\hat{p}_j \leftarrow \hat{p}_j + \alpha$
      end if
      $\mathcal{D}_{\text{RL}} \leftarrow \mathcal{D}_{\text{RL}} \cup \{(x, C_j, \hat{p}_j)\}$
    end for
  end for
  $\theta \leftarrow$ REINFORCE training with $\mathcal{D}_{\text{RL}}$
end for

in which they normalized the probabilities of the generated programs in bean to sum to $(1-\alpha)$ and add α to the probability of the best found one $C^*(x_i)$. The REINFORCE model always put a reasonable amount of probability on a program with higher reward during training.

4.2.5 CHENG17[73]

Sequence-to-sequence model for semantic parsing reduces the need for a domain-specific assumption, grammar learning, and more expensive feature engineering. But this modeling flexibility brings a problem that it is no longer possible to interpret how the meaning composition is performed. Meanwhile, this knowledge plays a critical role in understanding modeling limitations. Moreover, without any task-specific knowledge, the learning is unconstrained and may result in ill-formed output. CHENG17[73] proposed a neural semantic parser that alleviates the aforementioned problems. CHENG17[73] first mapped utterances to an intermediate representation containing natural language predicates. However, instead of using an external parser[74,75] or CCG grammars, CHENG17 induced intermediate representations in the form of predicate-argument structures from data with a transition-based approach. This transition-based approach by design yielded recursive semantic structures, avoiding the problem of generating ill-formed meaning representations. Another advantage of transition-based approach is that it does not require feature decomposition over structures and thereby enabling the exploration of rich, non-local features[73]. After CHENG17 got the output from the transition system, they grounded (e.g., to a knowledge base) with a neural mapping model under an assumption that grounded and ungrounded structures are isomorphic. Details of this assumption can be referred in the original paper[73].

In the setting of CHENG17, the task is to learn semantic parser that maps input utterances x to logical form z via an intermediate ungrounded representation U. When G is executed against an executor, it outputs denotation (in weak supervision, optional) y. CHENG17 used FunQL[76] for both grounded and ungrounded meaning representation which is a variable-free query language, where each predicate is treated as a function symbol that modifies an argument list. For example, FunQL grounded representation for an input *which states do not border texas* is

$$answer(exclude(state(all),\ next\_to(texas))),$$

where *next_to* is a domain-specific binary predicate that takes one argument and returns a set of entities as its denotation. The ungrounded meaning representation for the same example is

$$answer(exclude(states(all), border(texas))),$$

where *states* and *border* are natural language predicates. An advantage of FunQL is that abstract predicate can be reused for both grounded and ungrounded meaning representation. They considered five types of domain-general predicates illustrated in Table 3. It is important to notice that domain-general predicates are often implicit and represent extra-sentential knowledge (e.g., *all* in the above utterances represents all states in the domain which is not mentioned in the text).

Table 3 List of domain-general predicates[73].

Predicate	Usage	Sub-category
Answer	Denotation wrapper	–
Type	Entity type checking	Stateid, cityid riverid, etc.
All	Querying for an entire set of entities	–
Aggregation	One-argument meta predicates for sets	Count, largest, smallest, etc.
Logical connectives	Two-argument meta predicates for sets	Intersect, union, exclude

CHENG17 decomposed the semantic parsing task in two stages: (1) convert the utterance to an intermediate representation and (2) ground the intermediate representation to a knowledge base.

A transition-based system generates the representation by following a derivation tree and some canonical generation order. For FunQL, each predicate can be visualized as a non-terminal node of the tree while each entity as a terminal. Some special predicates (e.g., *all*) acted as a terminal directly. CHENG17 proposed a neural transition system based on RNNGs[77]. They used a buffer to store input tokens and a stack to store partially generated trees. A key difference in their approach was that tokens in the buffer were not fetched or removed in a sequential order. They allowed the generation algorithm to pick tokens and combine logical forms in arbitrary orders. They defined three actions in the system: *NT*, *TER*, and *RED*.

(1) $NT(X)$ generates a non-terminal predicate (either an entity or one of the domain-general predicates). The type of predicate is determined by the placeholder X and once generated, it is pushed onto the stack followed by an open bracket. The open bracket will be closed by a REDUCE operation.

(2) $TER(X)$ generates a terminal entity or the special predicate *all*.

(3) *RED* stands for reduce and is used for subtree completion. It recursively pops elements from the stack until an open non-terminal node. Then the composite term representing the entire subtree is pushed back to the stack. The system will terminate when there were no open non-terminals left in the stack.

Figure 9 shows the transition actions used to generate the example. The neural model generates the ungrounded representation U conditioned on utterance x by recursively calling one of the three actions. U is defined by a sequence of actions a and a sequence of term choices (u). The conditional probability $p(U|x)$ is

Sentence: *which states do not border texas*
Non-terminal symbols in buffer: *which, states, do, not, border*
Terminal symbols in buffer: *texas*

Stack	Action	NT choice	TER choice
	NT	*answer*	
answer (	NT	*exclude*	
answer (*exclude* (	NT	*states*	
answer (*exclude* (*states* (	TER		*all*
answer (*exclude* (*states* (*all*	RED		
answer (*exclude* (*states* (*all*)	NT	*border*	
answer (*exclude* (*states* (*all*) , *border* (	TER		*texas*
answer (*exclude* (*states* (*all*) , *border* (*texas*	RED		
answer (*exclude* (*states* (*all*) , *border* (*texas*)	RED		
answer (*exclude* (*states* (*all*) , *border* (*texas*))	RED		
answer (*exclude* (*states* (*all*) , *border* (*texas*)))			

Fig. 9 Actions taken by the transition system for generating the ungrounded meaning representation of the example. Symbols in red indicate domain-general predicates[73].

$$p(U|x) = p(a,u|x) = \prod_{t=1}^{T} p(a_t|a_{<t},x)\,p(u_t|a_{<t},x)^{(a_t \neq RED)}.$$

CHENG17 encoded the input buffer with a bidirectional LSTM and the output stack with a stack-LSTM[78]. At each time step, the model used the representation of the transition system e_t to predict an action, where e_t is the concatenation of the buffer representation b_t and the stack representation s_t. When the predicted action is *NT* or *TER*, ungrounded term u_t needs to be chosen from the candidate list (two types of choices in Fig. 9). To select a domain-general term, CHENG17 used e_t to compute the distribution over candidate terms. For natural language term, CHENG17 computed the distribution over terms in the buffer based on the s_t.

Since CHENG17 constrained the network to learn ungrounded structures that are isomorphic to the target meaning representation, how to convert ungrounded representations to grounded ones becomes a simple lexical mapping problem. To map an ungrounded term u_t to a grounded term g_t, CHENG17 computed the probability of g_t given u_t with a bi-linear neural network:

$$p(g_t|u_t) \propto \exp(\vec{u_t} \cdot W_{ug} \cdot \vec{g_t}^{\mathrm{T}}),$$

where $\vec{u_t}$ is the contextual representation of the ungrounded term given by the Bi-LSTM, $\vec{g_t}$ is the grounded term embedding, and W_{ug} is the weight matrix.

For the transition system, the objective function is

$$\mathcal{L}_a = \sum_{x \subset \mathcal{D}} \log p(a|x) = \sum_{x \subset \mathcal{D}} \sum_{t=1}^{n} \log p(a_t|x),$$

where $\mathcal{D}$ denotes examples in the training data. For the grounded term sequence g, since the ungrounded terms are latent, CHENG17 maximized the expected

log likelihood of the grounded terms for all examples, which is a lower bound of the log likelihood $\log p(g|x)$ by Jensen's Inequality:

$$\mathcal{L}_g = \sum_{x\subset\mathcal{D}}\sum_{u}[p(u|x)\log p(g|u,x)] = \sum_{x\subset\mathcal{D}}\sum_{u}[p(u|x)\sum_{t=1}^{k}\log p(g_t|u_t)] \leqslant \sum_{x\subset\mathcal{D}}\log p(g|x).$$

The final objective is the combination of $\mathcal{L}_a$ and $\mathcal{L}_g$, denoted as $\mathcal{L} = \mathcal{L}_a + \mathcal{L}_g$.

4.2.6 SU17[79]

The majority of existing approaches of semantic parsing focus on in-domain setting, while SU17[79] perform not well in cross-domain circumstance. Due to the diversity of language semantic in different domains, cross-domain still remains a challenging task. For instance, in Fig. 10, in *team* domain, the semantic parser is developed to extract the predicates like *team* and *season*. However, in the social domain, the semantic parser needs to extract some predicates like *employee*.

Although introducing a paraphrasing model is a widely adopted method, it still brings some open challenging problems. For example, OVERNIGHT dataset has eight domains, and 30% to 55% words in each domain never happen in other domains[79]. Due to this out-of-vocabulary problem, the paraphrasing model cannot address with cross-domain well.

To begin with, SU17 defined cross-domain semantic parsing task as follows: suppose U is a set of input utterances$(u_i, z_i)_{i=1}^{N_s}$, and Z is the set of logical forms. Given K source domains, the task is to train a semantic parser to generate a map $f: U \to Z$ for the target domain t, which contains a set of training examples $(u_i, z_i)_{i=1}^{N_t}$. Meanwhile, some attributes are induced as follows:

- Z_s and Z_t are completely non-intersecting.
- Typically, $N_s \gg N_t$.
- Assume the input utterance distributions in source domain and target domain are independent and different.

To deal with the issues listed above, SU17 proposed a paraphrase model and convert cross-domain semantic parsing into a domain adaptation problem. Generally, as shown in Fig. 10, the semantic parser for source domain needs to get canonical utterance from the logical forms. Taken them and input utterance as the input, the paraphrase model incorporates with external language resources to adapt new domains.

SU17 proposed a sequence-to-sequence model with soft-attention for domain adaptation problem. A sequence-to-sequence model is composed with two components: *encoder* and *decoder*. For the encoder, they exploited a bi-directional RNN to encode the input utterance $u(u_1, u_2, \ldots, u_m)$ to a sequence of state vectors $h(h_1, h_2, \ldots, h_m)$. The state vectors of forward RNN and backward RNN are computed respectively as

$$\overrightarrow{h_i} = \mathrm{GRU}_{fw}(\phi(u_i), \overrightarrow{h_{i-1}}),$$
$$\overleftarrow{h_i} = \mathrm{GRU}_{fw}(\phi(u_i), \overrightarrow{h_{i+1}}).$$

SU17 also proposed an RNN model with attention as the $decoder$, which generates one token at a time step[79]. The attention weights are calculated by encoder states, and then they are used to compute the next decoder state d_{j+1} and the probability distribution $p(c_j|u, c_{1:j-1})$ as follows:

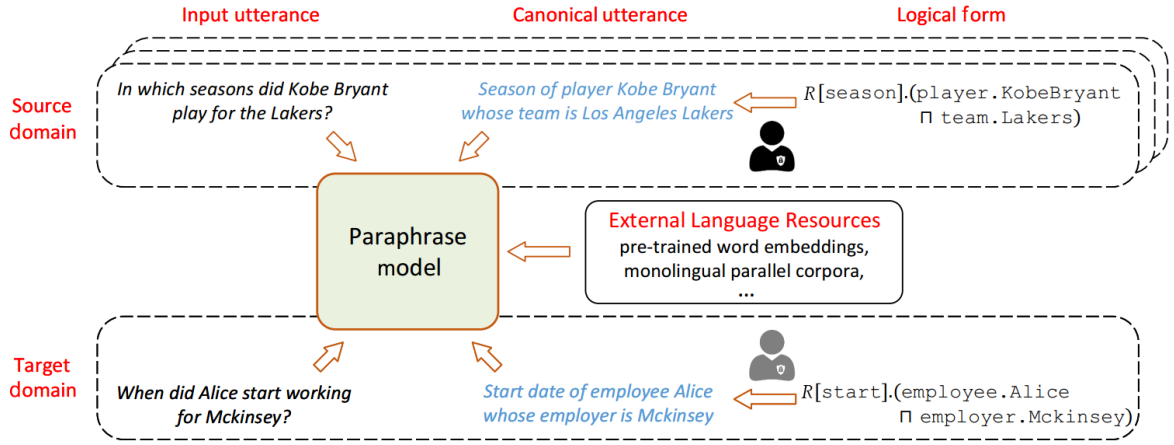


Fig. 10 Framework of semantic parsing via paraphrasing. Firstly, the logical forms are converted deterministically into canonical utterance in natural language. Combing canonical utterance with input utterance as the input, paraphrase model is trained to learn and transfer from the source-domain to the target-domain. External language resources are applied to facilitate domain adaptation[79].

$$
\begin{aligned}
d_0 &= \tanh(W_0[\overrightarrow{h_m}, \overleftarrow{h_1}]),\\
u_{ji} &= v^{\mathrm{T}} \tanh(W_1 H_i + W_2 d_j),\\
\alpha_{ji} &= \frac{U_{ji}}{\sum_{i'=1}^{m} u_{ji'}},\\
h'_j &= \sum_{i=1}^{m} \alpha_{ji} h_i,\\
d_{j+1} &= \mathrm{GRU}([\phi(c_j), h'_j], d_j),\\
p(c_j|u, c_{1:j-1}) &\propto \exp(U[d_j, h'_j]).
\end{aligned}
$$

Given a set of training data $(u_i, c_i)_{i=1}^{N}$, SU17 minimized the loss function $-\frac{1}{N}\sum_{i=1}^{N} \log p(c_i|u_i)$ with cross-entropy. In other words, SU17 maximized the log probability of correct canonical utterances.

Besides, taking advantages of the standardize pre-trained word embedding, SU17 enabled the model to cope with unseen out-of-domain data. Specifically, they initialized the parameters of the word embedding layer with the pre-trained word embedding model in the source domain. By doing this, SU17 addressed with the out-of-vocabulary problems. Based on the observations during the experiments, SU17 claimed that word embedding initialization faces with two dilemmas: one is *small micro variance*, which hinders optimization, and the other is *large macro variance*, which hinders generalization. The comparison between different word embedding initializations (Table 4) shows that the small micro variance brings a poor starting point, while large variance is hard to generalize with the unseen words during training.

Based on the analysis above, SU17 proposed a *unit variance standardization* after initializing word embedding. There are two choices, *per-example standardization* which standardizes every row of the word embedding matrix by simple dividing them with the standard deviation, and *per-feature* which does the same operations on columns instead.

Table 4 Comparison between different word embedding initializations. ES: per-example standardization. FS: per-feature standardization. EN: per-example normalization. Cosine similarity is computed on a randomly selected.

Initialization	L2 norm	Micro variance	Cosine similarity
Random	17.3 ± 0.45	1.00 ± 0.05	0.00 ± 0.06
WORD2VEC	2.04 ± 1.08	0.02 ± 0.02	0.13 ± 0.11
WORD2VEC+ES	17.3 ± 0.05	1.00 ± 0.00	0.13 ± 0.11
WORD2VEC+FS	16.0 ± 8.47	1.09 ± 1.31	0.12 ± 0.10
WORD2VEC+EN	1.00 ± 0.00	0.01 ± 0.00	0.13 ± 0.11

4.2.7 HERZIG17[80]

In parallel of previous approch[79], Ref. [80] explored another direction of semantic parsing with multiple domains. HERZIG17[80] trained a single model for all the domains, and attached a domain-specific encoding to help the parser distinguish between domains. HERZIG17 also employed a sequence-to-sequence model with attention as a base structure. To add the explicit representation of the domain that is being decoded, HERZIG17 encoded the k-th domain by a one-hot vector $d_k \subset \mathbb{R}^K$ as an addition input of the decoder. As an alternate way, HERZIG17 adopted a similar intuition from neural machine translation[81], where they added an artificial token at the beginning of each input sentence to specify the target domain. Since HERZIG17 used one decoder for multiple domains, tokens from different domains could be generated during decoding. HERZIG17 prevented that at test time by excluding out-of-domain tokens before the last softmax function. The experiments showed that training a single model for all domains not only got a better result but also contained much less parameters. In addition, HERZIG17 found the single model performed much better when training data is limited.

5 Datasets

Datasets play a critical role in both conventional statistical learning and deep learning. We survey some of the existing datasets and describe their properties together with the state-of-the-art performance. Table 5 gives an overview of all datasets.

5.1 JOBS

This benchmark dataset[85] has 640 queries to a database of job listing. Each question in it is paired with Prolog-style query. In Refs. [28, 70], JOBS split the dataset into 500 training samples and 140 test instances. Best accuracy of this dataset is 0.9 achieved by Ref. [70].

5.2 Geo880

The Geo880 dataset[26], which is published in 1996, drove nearly a decade of semantic parsing research. It has 880 questions and database query pairs about US geography (e.g., "what is the highest point in the largest state"). The utterances are compositional, but the language is simple and the domain is limited. The majority of questions include at most one entity. On this dataset, learning from logical forms[48] and data[6] both achieve around 90% accuracy. For deep learning approaches, CHENG17[73] achieves 86.7% accuracy

Table 5 An overview of all datasets ("–" means there is not official split for this dataset).

Dataset name	Training set number	Development set number	Test set number	Best result (%)	Supervision form
JOBS	500	–	140	90.0[70]	Supervision
Geo880	880	–	–	91.1[6]	Supervision
ATIS–3	5418	–	–	84.6[31]	Supervision
Regexp824	824	–	–	65.6[13]	Supervision
Free917	917	–	–	68[13]	Supervision
WebQuestions	5810	–	–	58.8[60]	Weak Supervision
WebQuestionSP	3098	–	1639	63.9[60, 82]	Weak Supervision
SPADES	93 319	–	–	39.9[83]	Weak Supervision
SimpleQuestion	75 910	10 845	21 687	78.7[82]	Weak Supervision
WikiTableQuestions	22 033	–	–	37.1[84]	Weak Supervision
OVERNIGHT	13 682	–	–	80.6[79]	Supervision
IFTTT	77 495	5171	4294	74.2[70]	Supervision

which is better compared with JIA16[64] but slightly lower than conventional approaches.

5.3 ATIS-3

Published in 2007, the ATIS-3 dataset[31] consists of 5418 utterances paired with logical forms (e.g., "show me information on american airlines from fort worth texas to philadelphia"). They are extracted from a flight booking system. The standard split has 4480 training instances, 480 development instances, and 450 test instances. The utterances in this dataset contain more disfluencies and flexible word order compared with Geo880, but they are logically simpler. The best-reported result is based on semantic parsing and obtains 84.6% accuracy[31].

5.4 Regexp824

The Regexp824 dataset contains 824 natural language and regular expression pairs published in 2013[13], one example is "three letter words starting with 'X'". One challenge of this dataset is that there are many logically equivalent regular expressions, some aligning better to the natural language than others. Reference [13] obtains the best performance with 65.6% accuracy.

5.5 Free917

The Free917[47] published in 2013 has 917 examples of question-logical form pairs that can be answered directly via Freebase (e.g., "how many works did Mozard dedicate to Joseph Haydn?"). The questions are logically less complex than those in the semantic parsing datasets above, but introduce the new challenge of scaling up to many more predicates (need to deal with a large knowledge base). The state-of-the-art performance is 68% accuracy[13].

5.6 WebQuestions

WebQuestions[49] is another dataset on Freebase published in 2013. It consists of 5810 question-answer pairs (no logical form) such as "what do Australia call their money?" Similar to Free917, the questions are not very complicated, but unlike Free917, the questions in WebQuestions are real questions asked by people over the world on the Web independent from Freebase. These real questions result in that the dataset is more realistic and varied. In this dataset, STAGG[60] achieves the best performance with accuracy 58.8%.

5.7 WebQuestionSP

WebQuestionSP dataset[86] is an extension word of WebQuestions published in 2016. It contains full semantic parses for a subset of the questions from WebQuestions. Because 18.5% of the original dataset were found "not answerable". This dataset has 3098 question-answer pairs for training and 1639 for testing with labeled logical form, which was collected using Google Suggest API. The state-of-the-art result of this dataset is 63.9% achieved by STAGG and Ref. [82].

5.8 SPADES

Datasets above have a limitation on the number of training samples. SPADES[87] consists of 93 319 questions derived from CLUEWEB09[88] sentences published in 2016. Specifically, the questions were created by randomly removing an entity, and producing sentence-denotation pairs. The sentences include two or more entities and although they are not very compositional, it is still a large-scale dataset for neural network training. Reference [83] achieves the state-of-the-art performance with an F1 score 39.9%. Reference [83] leverages external unstructured text information to do answer selection, which is not directly related to

semantic parsing.

5.9 SimpleQuestion

Another dataset for a large scale training schema is SimpleQuestion[89] published in 2015. It consists of a total 108 442 questions written in natural language with corresponding fact and answer. Facts are extracted from Freebase. The dataset is split into a training set (75 910 samples), a development set (10 845 samples), and a test set (21 687 samples). The current best result is achieved by Ref. [82] with accuracy 78.7%.

5.10 WikiTableQuestions

The goal of WikiTableQuestions is to extend question answering beyond knowledge based on HTML tables on Wikipedia, which are semi-structured[84]. The dataset consists of 22 033 question-table-answer pairs (e.g., "how many runners took 2 minutes at the most to run 1500 meters?"). Each question can be answered based on the information inside a given table. People need to aggregate information across the whole table with the reasoning based on the question. At test time, new questions together with new tables are provided to be answered. The result of this dataset is 37.1%.

5.11 OVERNIGHT

OVERNIGHT[90] dataset consists of 8 domains, each of them is based on a separate knowledge base with logical forms written in λ-*DCS*. The logical forms are converted into canonical utterances via a simple grammar. The input utterances are collected by crowd sourcing. Different domains are aimed to stress different linguistic phenomena, which requires the parser more flexible and general. References [79, 80] both achieved best accuracy around 0.8 on average of the 8 domains.

5.12 IFTTT

Reference [91] created this dataset in 2015. This dataset extracted a large number of if-this-then-that recipes from the IFTTT website (http://www.ifttt.com). Recipes are the programs that users specify on the site with exactly one trigger and one action (whenever the trigger is activated, the action is performed). Different from other datasets, in IFTTT, each utterance pairs with one trigger and on action, e.g., an utterance *"turn on heater when temperature drops below 58 degree"*, a TRIGGER: *Weather — Current temperature drops below* — ((*Temperature* (58)) (*Degrees in* (f))), and an ACTION: *WeMo Insight Switch — Turn on* — ((*Which switch?* (" "))). The original split which contains 77 495 training samples, 5171 development samples, and 4294 test samples. Best F1 score is 74.2% achieved by Ref. [70].

6 Challenges and Future Tendency

In this section, we discuss the current challenges and a future tendency of semantic parsing. Although a lot of effort has been applied in recent decades, the performance is still far from satisfaction. We conclude three challenges and several potential directions for semantic parsing.

6.1 Challenges

First, unlike other structure prediction problems, the number of labels in semantic parsing is much more. However, labeling the semantic parsing data costs a lot. In supervised learning, datasets need to be labeled as pairs of utterance and logic form while in weak supervised learning, the corresponding action is needed. Meanwhile, although semantic parser can handle utterances in a narrow and closed domain setting, it still lacks the ability to parse and understand open-domain queries from human[92].

Second, there is no universal logic form for all tasks. Syntax and complexity of different logic forms vary a lot. To learn one model for all is a very challenging problem. Although there are several models trained in a multi-task way, it still lacks the ability of generalization. This is an open problem for all machine learning problems. Fortunately, researchers have proposed several ways for it.

Third, is it a good way to represent semantic or language in a probability way? Logic forms are connected with rules and discrete in a natural way and so does language. Combining logic and probability may be a potential direction for solving this problem, and it still needs a lot of effort.

6.2 Current and future tendency

Although semantic parsing is facing a bunch of problems, researchers still try to settle them with some other methods. We discuss some potential directions on semantic parsing as follows.

6.2.1 Transfer learning

There are some approaches for semantic parsing via transfer learning, trained from a source task with a large amount of labeled data together with a target task with smaller labeled data. Reference [93] proposed to train a single multi-task deep learning model with shared implicit features learning across domains. The approach

showed huge improvement in semantic template level performance. Similar algorithms[94] are proposed, which leveraged a shared feature extraction layer for slot-filling across multiple domains. In Reference [95], the authors achieved good performance of a seq-to-seq model with three multi-task architectures. Transfer learning can also be extended as lifelong learning, whose goal is to sequentially retain learned knowledge and to selectively transfer that knowledge when learning a new task to develop more accurate hypotheses or policies[96].

6.2.2 Zero-shot learning

Zero-shot learning aims to recognize objects whose instances may not have been seen during training along with some side information. Some work based on zero-shot learning has been applied to semantic parsing. Reference [92] proposed a slot-filling approach with natural language slot label descriptions. With multi-task stacked Bi-LSTM, this method can alleviate the need for sufficient labeled training data and bootstrap a model for domain scaling problems. Similar ideas are exploited in Ref. [97] to establish a general slot or concept tagger over multiple domains. Different from lifelong learning, the zero-shot learning is more strict.

6.2.3 Model-agnostic meta-learning

Model-Agnostic Meta-Learning (MAML)[98] aims to learn the learners (for the tasks) and the meta-learner in the few-shot meta-learning setup. It considers a model represented by parameters θ. When the model adapts to a new task, the model changes its parameters θ to $\theta_{i'}$. In this way, the model can quickly adapt to a new task given very few training samples. Reference [99] reduced the supervised learning problem to a few-shot learning problem by treating each training example as a unique *pseudo-task* and achieved faster converge and better performance when generating Structured Query Language (SQL) from wiki queries.

7 Conclusion

We have presented a general semantic parsing framework for the problem of natural language understanding, and then given an introduction of some representative statistical learning algorithms including deep learning for semantic parsing. Going forward, we also collected the datasets used to train semantic parsing models and provided the state-of-the-art performance. At last, we analyzed several challenges for semantic parsing and pointed out several potential ways for further research.

Acknowledgment

This work was partially supported by National Science Foundation (No. CNS-1842407), National Institutes of Health (No. R01GM110240), and Industry Members of NSF Center for Big Learning (http://nsfcbl.org/index.php/partners/).

References

[1] P. Pasupat and P. Liang, Compositional semantic parsing on semi-structured tables, in *Proc. 53rd Ann. Meeting of the Association for Computational Linguistics and the 7th Int. Joint Conf. Natural Language Processing*, Beijing, China, 2015.

[2] P. Liang, Learning executable semantic parsers for natural language understanding, *Commun. ACM*, vol. 59, no. 9, pp. 68–76, 2016.

[3] W. Woods, R. Kaplan, and B. Webber, *The Lunar Sciences Natural Language Information System: Final Report*. Cambridge, MA, USA: Bolt Beranek and Newman Inc., 1972.

[4] T. Winograd, *Understanding Natural Language*. New York, NY, USA: Academic Press, 1972.

[5] J. Clarke, D. Goldwasser, M. W. Chang, and D. Roth, Driving semantic parsing from the world's response, in *Proc. 14th Conf. Computational Natural Language Learning*, Uppsala, Sweden, 2010, pp. 18–27.

[6] P. Liang, M. I. Jordan, and D. Klein, Learning dependency-based compositional semantics, in *Proc. 49th Ann. Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, Portland, OR, USA, 2011, pp. 590–599.

[7] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, Freebase: A collaboratively created graph database for structuring human knowledge, in *Proc. 2008 ACM SIGMOD Int. Conf. Management of Data*, Vancouver, Canada, 2008, pp. 1247–1250.

[8] D. Vrandecic and M. Krötzsch, Wikidata: A free collaborative knowledgebase, *Commun. ACM*, vol. 57, no. 10, pp. 78–85, 2014.

[9] S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. Teller, and N. Roy, Understanding natural language commands for robotic navigation and mobile manipulation, in *Proc. 25th AAAI Conf. Artificial Intelligence*, San Francisco, CA, USA, 2011.

[10] Y. Artzi and L. Zettlemoyer, Weakly supervised learning of semantic parsers for mapping instructions to actions, *Trans. Assoc. Comput. Linguist.*, vol. 1, pp. 49–62, 2013.

[11] C. Matuszek, N. FitzGerald, L. Zettlemoyer, L. Bo, and D. Fox, A joint model of language and perception for grounded attribute learning, in *Proc. 29th Int. Conf. Machine Learning*, Edinburgh, UK, 2012.

[12] J. Krishnamurthy and T. Kollar, Jointly learning to parse and perceive: Connecting natural language to the physical world, *Trans. Assoc. Comput. Linguist.*, vol. 1, pp. 193–206, 2013.

[13] N. Kushman and R. Barzilay, Using semantic unification

to generate regular expressions from natural language, in *Proc. 2013 Conf. North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Atlanta, GA, USA, 2013.

[14] Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[15] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.R. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, et al., Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups, *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 82–97, 2012.

[16] K. M. He, X. Y. Zhang, S. Q. Ren, and J. Sun, Deep residual learning for image recognition, in *Proc. 2016 IEEE Conf. Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, 2016, pp. 770–778.

[17] Y. Li, H. Z. Qi, J. F. Dai, X. Y. Ji, and Y. C. Wei, Fully convolutional instance-aware semantic segmentation, in *Proc. 2017 IEEE Conf. Computer Vision and Pattern Recognition*, Honolulu, HI, USA, 2017.

[18] X. Z. Ma and E. Hovy, End-to-end sequence labeling via Bi-directional LSTM-CNNs-CRF, arXiv preprint arXiv:1603.01354, 2016.

[19] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, Neural architectures for named entity recognition, in *Proc. 2016 Conf. North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, San Diego, CA, USA, 2016.

[20] Q. L. Zhu, X. L. Li, A. Conesa, and P. Cecile, GRAM-CNN: A deep learning approach with local context for named entity recognition in biomedical text, *Bioinformatics*, vol. 34, no. 9, pp. 1547–1554, 2018.

[21] R. M. Sun, X. Y. Yuan, P. He, Q. L. Zhu, A. K. Chen, A. Gregio, D. Oliveira, and X. L. Li, Learning fast and slow: Propaedeutica for real-time malware detection, arXiv preprint arXiv:1712.01145, 2017.

[22] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, Natural language processing (almost) from scratch, *J. Mach. Learn. Res.*, vol. 12, pp. 2493–2537, 2011.

[23] E. F. T. K. Sang and F. De Meulder, Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition, in *Proc. 7^{th} Conf. Natural Language Learning at HLT-NAACL 2003-Volume 4*, Edmonton, Canada, 2003, pp. 142–147.

[24] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini, Building a large annotated corpus of English: The Penn Treebank, *Comput. Linguist.*, vol. 19, no. 2, pp. 313–330, 1993.

[25] D. K. Misra and Y. Artzi, Neural shift-reduce CCG semantic parsing, in *Proc. 2016 Conf. Empirical Methods in Natural Language Processing*, Austin, TX, USA, 2016, pp. 1775–1786.

[26] J. M. Zelle and R. J. Mooney, Learning to parse database queries using inductive logic programming, in *Proc. 13^{th} National Conf. Artificial Intelligence*, Portland, OR, USA, 1996, pp. 1050–1055.

[27] Y. W. Wong and R. J. Mooney, Learning for semantic parsing with statistical machine translation, in *Proc. Main Conf. Human Language Technology Conf. North American Chapter of the Association of Computational Linguistics*, New York, NY, USA, 2006, pp. 439–446.

[28] L. S. Zettlemoyer and M. Collins, Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars, in *Proc. 21^{st} Conf. Uncertainty in Artificial Intelligence*, Edinburgh, UK, 2005.

[29] M. Steedman, *The Syntactic Process*. Cambridge, MA, USA: MIT Press, 2000.

[30] Y. Artzi, K. Lee, and L. Zettlemoyer, Broad-coverage CCG semantic parsing with AMR, in *Proc. 2015 Conf. Empirical Methods in Natural Language Processing*, Lisbon, Portugal, 2015, pp. 1699–1710.

[31] L. S. Zettlemoyer and M. Collins, Online learning of relaxed CCG grammars for parsing to logical form, in *Proc. Joint Conf. Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, Prague, Czech Republic, 2007, pp. 678–687.

[32] T. Kwiatkowski, L. Zettlemoyer, S. Goldwater, and M. Steedman, Inducing probabilistic CCG grammars from logical form with higher-order unification, in *Proc. 2010 Conf. Empirical Methods in Natural Language Processing*, Cambridge, MA, USA, 2010, pp. 1223–1233.

[33] P. Liang, Lambda dependency-based compositional semantics, arXiv preprint arXiv:1309.4408, 2013.

[34] Z. C. Zheng, F. T. Li, M. L. Huang, and X. Y. Zhu, Learning to link entities with knowledge base, in *Human Language Technologies: The 2010 Ann. Conf. North American Chapter of the Association for Computational Linguistics*, Los Angeles, CA, USA, 2010, pp. 483–491.

[35] Y. Yang and M. W. Chang, S-mart: Novel tree-based structured learning algorithms applied to tweet entity linking, in *Proc. Association for Computational Linguistics*, Beijing, China, 2015.

[36] R. C. Schank and L. Tesler, A conceptual dependency parser for natural language, in *Proc. 1969 Conf. Computational Linguistics*, Sång-Säby, Sweden, 1969, pp. 1–3.

[37] K. Zhao and L. Huang, Type-driven incremental semantic parsing with polymorphism, in *Proc. Human Language Technologies: The Ann. Conf. North American Chapter of the ACL*, Denver, CO, USA, 2015.

[38] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[39] M. Collins, Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms, in *Proc. ACL-02 Conf. Empirical Methods in Natural Language Processing-Volume 10*, Stroudsburg, PA, USA, 2002, pp. 1–8.

[40] F. Rosenblatt, *The Perceptron: A Perceiving and Recognizing Automation*. Buffalo, NY, USA: Cornell Aeronautical Laboratory, 1957.

[41] Y. Freund and R. E. Schapire, Large margin classification using the perceptron algorithm, *Mach. Learn.*, vol. 37, no. 3, pp. 277–296, 1999.

[42] K. Zhao, *Structured Prediction with Perceptron*: *Theory and Algorithms*, New York, NY, USA: The City University of New York, 2014.

[43] P. Liang, A. Bouchard-Côté, D. Klein, and B. Taskar, An end-to-end discriminative approach to machine translation, in *Proc. 21st Int. Conf. Computational Linguistics and the 44th Ann. Meeting of the Association for Computational Linguistics*, Sydney, Australia, 2006, pp. 761–768.

[44] N. Singh-Miller and M. Collins, Trigger-based language modeling using a loss-sensitive perceptron algorithm, in *Proc. 2007 IEEE Int. Conf. Acoustics, Speech and Signal Processing*, Honolulu, HI, USA, 2007, pp. 25–28.

[45] Y. Artzi and L. Zettlemoyer, Bootstrapping semantic parsers from conversations, in *Proc. 2011 Conf. Empirical Methods in Natural Language Processing*, Edinburgh, UK, 2011, pp. 421–432.

[46] D. L. Chen and R. J. Mooney, Learning to interpret natural language navigation instructions from observations, in *Proc. 25th AAAI Conf. Artificial Intelligence*, San Francisco, CA, USA, 2011, pp. 859–865.

[47] Q. Q. Cai and A. Yates, Large-scale semantic parsing via schema matching and lexicon extension, in *Proc. 51st Ann. Meeting of the Association for Computational Linguistics*, Sofia, Bulgaria, 2013, pp. 423–433.

[48] T. Kwiatkowski, E. Choi, Y. Artzi, and L. Zettlemoyer, Scaling semantic parsers with on-the-fly ontology matching, in *Proc. 2013 Conf. Empirical Methods in Natural Language Processing*, Seattle, WA, USA, 2013.

[49] J. Berant, A. Chou, R. Frostig, and P. Liang, Semantic parsing on freebase from question-answer pairs, in *Proc. 2013 Conf. Empirical Methods in Natural Language Processing*, Seattle, WA, USA, 2013.

[50] J. Berant and P. Liang, Semantic parsing via paraphrasing, in *Proc. 52nd Ann. Meeting of the Association for Computational Linguistics*, Baltimore, MA, USA, 2014, pp. 1415–1425.

[51] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, Distributed representations of words and phrases and their compositionality, in *Proc. 26th Int. Conf. Neural Information Processing Systems*, Lake Tahoe, NV, USA, 2013, pp. 3111–3119.

[52] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, Backpropagation applied to handwritten zip code recognition, *Neural Comput.*, vol. 1, no. 4, pp. 541–551, 1989.

[53] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[54] Y. T. Zhou, R. Chellappa, A. Vaid, and B. K. Jenkins, Image restoration using a neural network, *IEEE Trans. Acoust. Speech Signal Process.*, vol. 36, no. 7, pp. 1141–1151, 1988.

[55] M. Schuster and K. K. Paliwal, Bidirectional recurrent neural networks, *IEEE Trans. Signal Process.*, vol. 45, no. 11, pp. 2673–2681, 1997.

[56] S. Hochreiter, Untersuchungen zu dynamischen neuronalen netzen, Master dissertation, Technische Universitaet München, Munich, Germany, 1991.

[57] Y. Bengio, P. Simard, and P. Frasconi, Learning long-term dependencies with gradient descent is difficult, *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, 1994.

[58] S. Hochreiter and J. Schmidhuber, Long short-term memory, *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[59] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, Learning phrase representations using RNN encoder-decoder for statistical machine translation, in *Proc. 2014 Conf. Empirical Methods in Natural Language Processing*, Doha, Qatar, 2014.

[60] S. W. T. Yih, M. W. Chang, X. D. He, and J. F. Gao, Semantic parsing via staged query graph generation: Question answering with knowledge base, in *Proc. 53rd Ann. Meeting of the Association for Computational Linguistics and the 7th Int. Joint Conf. Natural Language Processing*, Beijing, China, 2015, pp. 1321–1331.

[61] A. Bordes, S. Chopra, and J. Weston, Question answering with subgraph embeddings, in *Proc. 2014 Conf. Empirical Methods in Natural Language Processing*, Doha, Qatar, 2014.

[62] J. Bao, N. Duan, M. Zhou, and T. Zhao, Knowledge-based question answering as machine translation, in *Proc. 52nd Ann. Meeting of the Association for Computational Linguistics*, Baltimore, MD, USA, 2014.

[63] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, Signature verification using a "siamese" time delay neural network, in *Proc. 6th Int. Conf. Neural Information Processing Systems*, Denver, CO, USA, 1993, pp. 737–744.

[64] R. Jia and P. Liang, Data recombination for neural semantic parsing, in *Proc. 54th Ann. Meeting of the Association for Computational Linguistics*, Berlin, Germany, 2016, pp. 12–22.

[65] D. Bahdanau, K. Cho, and Y. Bengio, Neural machine translation by jointly learning to align and translate, in *Proc. Int. Conf. Learning Representations*, San Diego, CA, USA, 2015.

[66] M. T. Luong, H. Pham, and C. D. Manning, Effective approaches to attention-based neural machine translation, in *Proc. 2015 Conf. Empirical Methods in Natural Language Processing*, Lisbon, Portugal, 2015.

[67] O. Vinyals, M. Fortunato, and N. Jaitly, Pointer networks, in *Proc. 28th Int. Conf. Neural Information Processing Systems*, Montreal, Canada, 2015, pp. 2692–2700.

[68] A. Krizhevsky, I. Sutskever, and G. E. Hinton, Imagenet classification with deep convolutional neural networks, in *Proc. 25th Int. Conf. Neural Information Processing Systems*, Lake Tahoe, NV, USA, 2012, pp. 1097–1105.

[69] N. Jaitly and G. E. Hinton, Vocal Tract Length Perturbation (VTLP) improves speech recognition, in *Proc. 30th Int. Conf. Machine Learning*, Atlanta, GA, USA, 2013.

[70] L. Dong and M. Lapata, Language to logical form with neural attention, in *Proc. 54th Ann. Meeting of the Association for Computational Linguistics*, Berlin, Germany, 2016.

[71] C. Liang, J. Berant, Q. Le, K. D. Forbus, and N. Lao,

Neural symbolic machines: Learning semantic parsers on freebase with weak supervision, in *Proc. 55th Ann. Meeting of the Association for Computational Linguistics*, Vancouver, Canada, 2017, pp. 23–33.

[72] R. J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning, *Mach. Learn.*, vol. 8, nos. 3&4, pp. 229–256, 1992.

[73] J. P. Cheng, S. Reddy, V. Saraswat, and M. Lapata, Learning structured natural language representations for semantic parsing, in *Proc. 55th Ann. Meeting of the Association for Computational Linguistics*, Vancouver, Canada, 2017, pp. 44–55.

[74] S. Reddy, M. Lapata, and M. Steedman, Large-scale semantic parsing without question-answer pairs, *Trans. Assoc. Comput. Linguist.*, vol. 2, pp. 377–392, 2014.

[75] S. Reddy, O. Täckström, M. Collins, T. Kwiatkowski, D. Das, M. Steedman, and M. Lapata, Transforming dependency structures to logical forms for semantic parsing, *Trans. Assoc. Comput. Linguist.*, vol. 4, pp. 127–140, 2016.

[76] R. J. Kate, Y. W. Wong, and R. J. Mooney, Learning to transform natural to formal languages, in *Proc. 20th National Conf. Artificial Intelligence*, Pittsburgh, PA, USA, 2005, pp. 1062–1068.

[77] C. Dyer, A. Kuncoro, M. Ballesteros, and N. A. Smith, Recurrent neural network grammars, in *Proc. NAACL-HLT*, San Diego, CA, USA, 2016.

[78] C. Dyer, M. Ballesteros, W. Ling, A. Matthews, and N. A. Smith, Transition-based dependency parsing with stack long short-term memory, in *Proc. 53rd Ann. Meeting of the Association for Computational Linguistics and the 7th Int. Joint Conf. Natural Language Processing*, Beijing, China, 2015.

[79] Y. Su and X. F. Yan, Cross-domain semantic parsing via paraphrasing, in *Proc. 2017 Conf. Empirical Methods in Natural Language Processing*, Copenhagen, Denmark, 2017.

[80] J. Herzig and J. Berant, Neural semantic parsing over multiple knowledge-bases, in *Proc. 55th Ann. Meeting of the Association for Computational Linguistics*, Vancouver, Canada, 2017.

[81] M. Johnson, M. Schuster, Q. V. Le, M. Krikun, Y. H. Wu, Z. F. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado, et al., Google's multilingual neural machine translation system: Enabling zero-shot translation, *Trans. Assoc. Comput. Linguist.*, vol. 5, pp. 339–351, 2017.

[82] M. Yu, W. P. Yin, K. S. Hasan, C. dos Santos, B. Xiang, and B. W. Zhou, Improved neural relation detection for knowledge base question answering, in *Proc. 55th Ann. Meeting of the Association for Computational Linguistics*, Vancouver, Canada, 2017, pp. 571–581.

[83] R. Das, M. Zaheer, S. Reddy, and A. McCallum, Question answering on knowledge bases and text using universal schema and memory networks, in *Proc. 55th Ann. Meeting of the Association for Computational Linguistics*, Vancouver, Canada, 2017.

[84] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein, Learning to compose neural networks for question answering, in *Proc. NAACL-HLT*, San Diego, CA, USA, 2016.

[85] L. R. Tang and R. J. Mooney, Using multiple clause constructors in inductive logic programming for semantic parsing, in *Proc. 12th European Conf. Machine Learning*, Freiburg, Germany, 2001, pp. 466–477.

[86] W. T. Yih, M. Richardson, C. Meek, M. W. Chang, and J. Suh, The value of semantic parse labeling for knowledge base question answering, in *Proc. 54th Ann. Meeting of the Association for Computational Linguistics*, 2016, pp. 201–206.

[87] Y. Bisk, S. Reddy, J. Blitzer, J. Hockenmaier, and M. Steedman, Evaluating induced CCG parsers on grounded semantic parsing, in *Proc. 2016 Conf. Empirical Methods in Natural Language Processing*, Austin, TX, USA, 2016.

[88] E. Gabrilovich, M. Ringgaard, and A. Subramanya, *Facc1*: *Freebase Annotation of ClueWeb Corpora*. Google Inc., 2013.

[89] A. Bordes, N. Usunier, S. Chopra, and J. Weston, Large-scale simple question answering with memory networks, arXiv preprint arXiv:1506.02075, 2015.

[90] Y. S. Wang, J. Berant, and P. Liang, Building a semantic parser overnight, in *Proc. 53rd Ann. Meeting of the Association for Computational Linguistics and the 7th Int. Joint Conf. Natural Language Processing*, Beijing, China, 2015, pp. 1332–1342.

[91] C. Quirk, R. Mooney, and M. Galley, Language to code: Learning semantic parsers for if-this-then-that recipes, in *Proc. 53rd Ann. Meeting of the Association for Computational Linguistics and the 7th Int. Joint Conf. Natural Language Processing*, Beijing, China, 2015, pp. 878–888.

[92] A. Bapna, G. Tür, D. Hakkani-Tür, and L. Heck, Towards zero-shot frame semantic parsing for domain scaling, in *Proc. INTERSPEECH*, Stockholm, Sweden, 2017.

[93] D. Hakkani-Tür, G. Tür, A. Celikyilmaz, Y. N. V. Chen, J. F. Gao, L. Deng, and Y. Y. Wang, Multi-domain joint semantic frame parsing using bi-directional RNN-LSTM, in *Proc. 17th Ann. Meeting of the Int. Speech Communication Association*, San Francisco, CA, USA, 2016, pp. 715–719.

[94] A. Jaech, L. Heck, and M. Ostendorf, Domain adaptation of recurrent neural networks for natural language understanding, in *Proc. 17th Ann. Meeting of the Int. Speech Communication Association*, San Francisco, CA, USA, 2016.

[95] X. Fan, E. Monti, L. Mathias, and M. Dreyer, Transfer learning for neural semantic parsing, in *Proc. 2nd Workshop on Representation Learning for NLP*, Vancouver, Canada, 2017.

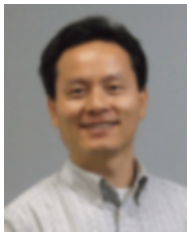
[96] D. L. Silver, Q. Yang, and L. H. Li, Lifelong machine learning systems: Beyond learning algorithms, in *AAAI Spring Symp.*: *Lifelong Machine Learning*, Palo Alto, CA, USA, 2013, p. 5.

[97] Y. N. Dauphin, G. Tür, D. Hakkani-Tür, and L. P.

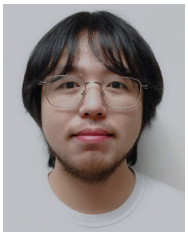
Heck, Zero-shot learning and clustering for semantic utterance classification, in *Proc. 2nd Int. Conf. Learning Representations*, Banff, Canada, 2014.

[98] C. Finn, P. Abbeel, and S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in *Proc. 34th Int. Conf. Machine Learning*, Sydney, Australia, 2017, pp. 1126–1135.

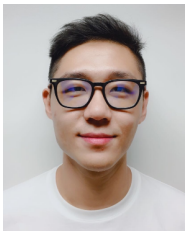
[99] P. S. Huang, C. L. Wang, R. Singh, W. T. Yih, and X. D. He, Natural language to structured query generation via meta-learning, in *Proc. NAACL-HLT 2018*, New Orleans, LA, USA, 2018.



Xiaolin Li is a professor and university term professor in Department of Electrical and Computer Engineering (ECE) and Department of Computer at University of Florida. He is the founding director of National Science Foundation Center for Big Learning (CBL). His research interests include big data, machine learning, deep learning, cloud computing, intelligent platforms, HPC, security & privacy for health, precision medicine, IoT, CV, NLP, robotics, genomics, and science, engineering, and business. He received the PhD degree from Rutgers University. He received the National Science Foundation CAREER Award in 2010, the Internet2 Innovative Application Award in 2013, NSF I-Corps Top Team Award in 2015, Top Team (DeepBipolar) in the CAGI Challenge on detecting bipolar disorder in 2016, and best paper awards (IEEE ICMLA 2016, IEEE SECON 2016, ACM CAC 2013, and IEEE UbiSafe 2007).



Qile Zhu is a PhD student in National Science Foundation (NSF) Center for Big Learning, Department of Computer & Information Science & Engineering (CISE), Univerisity of Florida. His research interests include natural language processing and understanding, graph analysis, and Bayesian methods.



Xiyao Ma is a PhD student in NSF Center for Big Learning, Department of Electrical and Computer Engineering, Univerisity of Florida. He is interested in natural language processing, generative adversarial network, and adversarial examples.