Iterative Utterance Segmentation for Neural Semantic Parsing

Yinuo Guo[1]*, **Zeqi Lin**[2], **Jian-Guang Lou**[2], **Dongmei Zhang**[2]
[1]Key Laboratory of Computational Linguistics, School of EECS, Peking University
[2]Microsoft Research Asia
gyn0806@pku.edu.cn, {Zeqi.Lin, jlou, dongmeiz}@microsoft.com

Abstract

Neural semantic parsers usually fail to parse long and complex utterances into correct meaning representations, due to the lack of exploiting the principle of compositionality. To address this issue, we present a novel framework for boosting neural semantic parsers via iterative utterance segmentation. Given an input utterance, our framework iterates between two neural modules: a *segmenter* for segmenting a span from the utterance, and a *parser* for mapping the span into a partial meaning representation. Then, these intermediate parsing results are composed into the final meaning representation. One key advantage is that this framework does not require any handcraft templates or additional labeled data for utterance segmentation: we achieve this through proposing a novel training method, in which the parser provides pseudo supervision for the segmenter. Experiments on GEO, COMPLEXWEBQUESTIONS and FORMULAS show that our framework can consistently improve performances of neural semantic parsers in different domains. On data splits that require compositional generalization, our framework brings significant accuracy gains: GEO 63.1 → 81.2, FORMULAS 59.7 → 72.7, COMPLEXWEBQUESTIONS 27.1 → 56.3.

Introduction

Semantic parsing is the task of mapping natural language utterances to machine interpretable meaning representations. Many semantic parsing methods are based on the principle of semantic compositionality (aka, compositional semantics) (Pelletier 1994), of which the main idea is to put together the meanings of utterances by combining the meanings of the parts (Zelle and Mooney 1996; Zettlemoyer and Collins 2005, 2007; Liang, Jordan, and Klein 2011; Pasupat and Liang 2015). However, these methods suffer from heavy dependence on handcrafted grammars, lexicons, and features.

To overcome this problem, many neural semantic parsers have been proposed and achieved promising results (Jia and Liang 2016; Dong and Lapata 2016; Ling et al. 2016; Dong and Lapata 2018; Shaw et al. 2019). However, due to the lack of capturing compositional structures in utterances, neural semantic parsers usually have poor generalization ability to handle unseen compositions of semantics (Finegan-Dollak et al. 2018). For example, a parser trained on "*How many rivers run through oklahoma?*" and "*Show me states bordering colorado?*" may not perform well on "*How many rivers run through the states bordering colorado?*".

Q	How many rivers run through the states bordering colorado
M	$count(river(traverse\_2(state(next\_to\_2(stateid('colorado'))))))$
Q1	the states bordering colorado
M1	$state(next\_to\_2(stateid('colorado')))$
Q2	rivers run through $\$state\$$
M2	$river(traverse\_2(\$state\$))$
Q3	How many $\$river\$$
M3	$count(\$river\$)$

Figure 1: An example of iterative utterance segmentation for semantic parsing. The first cell shows an utterance (Q) and its meaning representation (M). The following three cells show how we iterate between: (1) segmenting the utterance to obtain a simpler span ($Q1$/$Q2$/$Q3$); (2) parsing the span into a partial meaning representation ($M1$/$M2$/$M3$). Finally, we compose $M1$, $M2$ and $M3$ into the final result (M).

In this paper, we propose a novel framework to boost neural semantic parsers with the principle of compositionality (Pelletier 1994). It iterates between segmenting a span from the utterance and parsing it into a partial meaning representation. Figure 1 shows an example. Given an utterance "*How many rivers run through the states bordering colorado?*", we parse it through three iterations: (1) we segment a span "*the states bordering colorado*" from the utterance, and parse it into $state(next\_to\_2(stateid('colorado')))$; (2) as the utterance is reduced to "*How many rivers run through $state$?*", we segment a span "*rivers run through $state$*" from it, and parse it into $river(traverse\_2(\$state\$))$; (3) the utterance is further reduced to "*How many $river$?*", and we parse it into $count(\$river\$)$. We compose these partial meaning representations into the final result.

Our framework consists of two neural modules: an utterance segmentation model (*segmenter* for short) and a base parser (*parser* for short). The former is in charge of segmenting a span from an utterance, and the latter is in charge of parsing the span into its meaning representation. These two

*Work done during an internship at Microsoft Research Asia.

Dataset	Example
GEO	*x: "How many rivers run through the states bordering colorado?"* *y: count(river(traverse_2(state(next_to_2(stateid('colorado'))))))*
COMPLEX WEBQUESTIONS	*x: "What is the mascot of the school where Thomas R. Ford is a grad student?"* *y: SELECT ?x WHERE {* *?c ns:education.educational_institution.students_graduates ?k .* *?k ns:education.education.student ns:m.0_thgpt .* *?c ns:education.educational_institution.mascot ?x }*
FORMULAS	*x: "What is the smaller value between F14 divided by E14 and the largest number in A1:D10."* *y: MIN(F14/E14, MAX(A1:D10))*

Table 1: Examples of natural language utterances (x) paired with their structured meaning representations (y) from our experimental datasets.

modules work together to parse complex input utterances in a divide-and-conquer fashion.

One key advantage of this framework is that it does not require any handcraft templates or additional labeled data for utterance segmentation: we achieve this through proposing a novel training method, in which the base parser provides pseudo supervision to the utterance segmentation model. Specifically: we train a preliminary base parser on the original train data; then, for each train sample (x, y), we leverage the preliminary base parser to collect all reasonable spans (those can be parsed to a part of y by the preliminary base parser). Finally, we use these collected spans as pseudo supervision signals for training the utterance segmentation model, without requiring any handcraft templates or additional labeled data.

In summary, our proposed framework has four advantages: (1) the base parser learns to parse simpler spans instead of whole complex utterances, thus alleviating the training difficulties and improving the compositional generalization ability; (2) our framework is flexible to incorporate various popular encoder-decoder models as the base parser; (3) our framework does not require any handcraft templates or additional labeled data for utterance segmentation; (4) our framework improves the interpretability of neural semantic parsing by providing explicit alignment between spans and partial meaning representations.

We conduct experiments on three datasets: GEO (Zelle and Mooney 1996), COMPLEXWEBQUESTIONS (Talmor and Berant 2018), and FORMULAS (a new dataset introduced in this paper). They use different forms of meaning representations: FunQL, SPARQL, and Spreadsheet Formula. Experimental results show that our framework consistently improves the performances of neural semantic parsers in different domains. On data splits that require compositional generalization, our framework brings significant accuracy gain: GEO $63.1 \rightarrow 81.2$, FORMULAS $59.7 \rightarrow 72.7$, COMPLEXWEBQUESTIONS $27.1 \rightarrow 56.3$.

Related Work

Semantic Parsing

There are two major paradigms of semantic parsing: compositional semantic parsing (Zelle and Mooney 1996; Zettlemoyer and Collins 2005; Liang, Jordan, and Klein 2011; Pasupat and Liang 2015; Berant and Liang 2015), and neural semantic parsing (Jia and Liang 2016; Dong and Lapata 2016; Ling et al. 2016; Dong and Lapata 2018; Shaw et al. 2019). Our work aims to combine their respective advantages.

In neural semantic parsing, various efforts have been made to leverage the syntax of meaning representations (typically, the tree structures) to enhance decoders (Dong and Lapata 2016; Yin and Neubig 2017; Guo et al. 2019). In these works, encoders treat input utterances as sequential tokens, without considering their compositional structures. On the other hand, some researchers focus on exploring data augmentation techniques to provide a compositional inductive bias in models (Jia and Liang 2016; Andreas 2019). However, they rely on exact matching of spans, which work well on word/phrase-level re-combination or simple domains (e.g., GEO and SCAN (Lake and Baroni 2018)), but is not suitable for more complex scenarios (e.g., diverse subsentences in COMPLEXWEBQUESTIONS). Therefore, the lack of compositional generalization ability is still a challenging problem in neural semantic parsing (Finegan-Dollak et al. 2018; Keysers et al. 2020).

Utterance Segmentation

In Question Answering, question segmentation has been successfully applied to help answer questions requiring multi-hop reasoning (Kalyanpur et al. 2012; Talmor and Berant 2018; Qi et al. 2019). A key challenge in these works is to derive supervision for question segmentation. Kalyanpur et al. (2012) segments questions based on predominantly lexico-syntactic features. Talmor and Berant (2018) leverages simple questions to derive distant supervision. Min et al. (2019) uses additional labeled data to fine-tune a BERT-based model; Qi et al. (2019) utilizes the longest common strings/sequences between utterances and their supporting context documents for segmentation.

In Semantic Parsing, Zhang et al. (2019) proposes HSP, a novel hierarchical semantic parsing method, which utilizes the decompositionality of complex utterances for semantic parsing. This method requires *(utterance, sub-utterances, meaning representation)* instances for training. Pasupat et al. (2019) proposes a span-based hierarchical semantic parsing method for task-oriented dialog. This method requires span-based annotations for training.

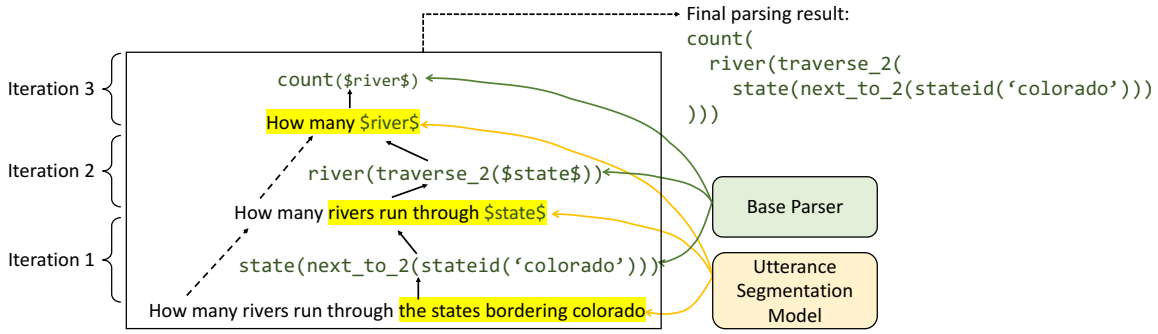

Figure 2: Framework overview. Given a natural language utterance, the framework iteratively segments a span (yellow highlight) from it and parses the span to *a partial meaning representation (green)*. We obtain span meanings via neural semantic parsing, then piece them to assemble whole-utterance meaning.

Framework

Problem Statement

Our goal is to learn semantic parsers from $\mathcal{D}$, a set of instances of natural language utterances paired with their structured meaning representations. We wish to estimate $p(y|x)$, the conditional probability of meaning representation y given utterance x. Table 1 shows examples from different datasets.

Framework Overview

Neural semantic parsers usually have poor generalization ability to handle unseen compositions of semantics. To address this problem, we propose to segment a complex utterance into simpler sub-utterances. We obtain partial meanings via neural semantic parsing, and then compose them together as the entire meaning representation.

Specifically, our framework consists of an utterance segmentation model and a base neural semantic parser. It parses utterances in an iterative "segment-and-parse" fashion. The k-th iteration process is detailed in the following four steps:

1. **Segmentation.** Given an input utterance $x^{(k)}$ ($x^{(1)}$ is the original input utterance), our utterance segmentation model predicts a span $\hat{x} = x^{(k)}_{i:j} (1 \leq i < j \leq |x^{(k)}|)$, representing an independent clause of $x^{(k)}$.

$$\underbrace{\text{How many rivers run through } \overbrace{\text{the states bordering colorado}}^{\hat{x}} \text{ ?}}_{x^{(k)}}$$

2. **Parsing.** The base parser maps span $\hat{x}$ to a partial meaning representation $\hat{y}$.

$$\hat{x} \rightarrow \overbrace{state(next\_to\_2(stateid('colorado')))}^{\hat{y}}$$

3. **Reducing.** Based on z, we reduce $x^{(k)}$ to $x^{(k+1)}$ through substituting $\hat{x}$ with $d(\hat{y})$:

$$x^{(k+1)} = [\hat{x} \mapsto d(\hat{y})]x^{(k)} \quad (1)$$

where the notation $[u \mapsto v]p$ refers to the result of replacing span u in p (an utterance or meaning representation) with v, and $d(\hat{y})$ is the denotation type (i.e., answer type) of $\hat{y}$[1].

$$\underbrace{\text{How many rivers run through } \overbrace{\text{\$state\$}}^{d(\hat{y})} \text{ ?}}_{x^{(k+1)}}$$

4. **Iteration.** If $s \neq x^{(k)}$, start the $(k+1)$ -th iteration. Otherwise, stop the iteration process, and compose the partial meaning representations produced in the iterations into the final result.

$$\underbrace{\text{How many } \overbrace{\text{rivers run through \$state\$}}^{\hat{x}} \text{ ?}}_{x^{(k+1)}}$$

Figure 2 illustrates how our framework parses a natural language utterance to its meaning representation through iterative utterance segmentation.

Probabilistic Reformulation

To be more formal, in this section we rephrase Section using conditional probability.

Firstly, the conditional probability $p(y|x)$ is decomposed into a two-stage generation process:

$$p(y|x) = \sum_{\hat{x} \subset x} p(y|x, \hat{x}) p(\hat{x}|x) \quad (2)$$

where $p(\hat{x}|x)$ represents the conditional probability of segmenting a span $\hat{x}$ from x, and $p(y|x, \hat{x})$ represents the conditional probability of parsing x to meaning representation y after the segmentation.

Then, we further decompose $p(y|x, \hat{x})$ into a two-stage generation process:

$$p(y|x, \hat{x}) = \sum_{\hat{y}} p(y|x, \hat{x}, \hat{y}) p(\hat{y}|x, \hat{x}) \quad (3)$$

As $\hat{x}$ represents an independent clause in x (which means that $\hat{y}$ is independent of x given $\hat{x}$), we have $p(\hat{y}|x, \hat{x}) = p(\hat{y}|\hat{x})$.

[1]Implementation details of $d(\hat{y})$ in different domains are presented in Section .

Now we consider $p(y|x, \hat{x}, \hat{y})$:

$$\begin{aligned} p(y|x, \hat{x}, \hat{y}) &= p(reduced\_y|reduced\_x) \\ reduced\_x &= [\hat{x} \mapsto d(\hat{y})]x \\ reduced\_y &= [\hat{y} \mapsto d(\hat{y})]y \end{aligned} \tag{4}$$

This corresponds to the iteration mechanism in our framework.

According to Equation 2, 3 and 4, the overall conditional probability $p(y|x)$ consists of three components:

- $p(\hat{x}|x)$: an utterance segmentation model that predicts a span $\hat{x}$ from x. Section details this component.
- $p(\hat{y}|\hat{x})$: a base parser that map $\hat{x}$ to a partial meaning representation $\hat{y}$. In this work, we make it a neural semantic parser based on encoder-decoder architecture. A simple implementation is detailed in Section .
- $p(y|x, \hat{x}, \hat{y})$: the iteration mechanism.

Base Parser

Our framework is flexible to incorporate various popular encoder-decoder models as the base parser. Without loss of generality, we use a typical sequence-to-sequence semantic parsing model proposed by Dong and Lapata (2016) as a default base parser.

The encoder is a bi-directional RNN with gated recurrent units (GRU, Cho et al. (2014)):

$$\mathbf{enc} = \text{Bi-GRU}_{enc}(\mathbf{s}) \tag{5}$$

The decoder is another GRU network with attention component (Luong, Pham, and Manning 2015):

$$\mathbf{dec} = \text{Attn-Bi-GRU}_{dec}(\mathbf{enc}) \tag{6}$$

Then, $p(\hat{y}_t|\hat{y}_{<t}, \hat{x})$, the conditional probability for generating the next word $\hat{y}_t$, is estimated via

$$p(\hat{y}_t|\hat{y}_{<t}, \hat{x}) = softmax_{z_t}(Linear(\mathbf{dec}_t)) \tag{7}$$

The base parser will be initially trained on $\mathcal{D}$ (the original train set), and further fine-tuned on pseudo supervision signals (detailed in Section).

Utterance Segmentation

We train an utterance segmentation model Seg that learns to predict a span (i.e., the start position i and end position j of the span) from x.

Span Prediction In Seg, we use a GRU to encode x:

$$\mathbf{U} = \text{GRU}_{seg}(x) \in \mathbb{R}^{m \times u} \tag{8}$$

Then, the span $\hat{x}$ is predicted via:

$$\begin{aligned} p(i|x) &= softmax_i(\mathbf{U}\mathbf{W}_I) \\ p(j|x) &= softmax_j(\mathbf{U}\mathbf{W}_J) \end{aligned} \tag{9}$$

where $\mathbf{W}_I, \mathbf{W}_J \in \mathbb{R}^u$.

Training

Pseudo Supervision Training the utterance segmentation model is non-trivial, because: (1) there is usually no labeled data of how utterances should be segmented; (2) we do not manage to use handcraft utterance segmentation templates aiming to have better generalization ability. To address this problem, we propose to leverage the base parser to derive pseudo supervision for utterance segmentation.

Take the aforementioned utterance "*How many rivers run through the states bordering colorado?*" as an example. As described in , we have a preliminary base parser which is trained on the original train set $\mathcal{D}$. For each span $\hat{x}$ in utterance x, we use this preliminary base parser to check whether this span is a "good" span. Specifically: we use the preliminary base parser to parse $\hat{x}$ into $\hat{y}$; if $\hat{y}$ is a part of the target meaning representation y, we call $\hat{x}$ a good span. For example, "*the states bordering colorado*" is a good span:

$$\text{How many rivers run through } \overbrace{\text{the states bordering colorado}}^{state(next\_to\_2(stateid('CO'))} ?$$

As a comparison, "*run through the states bordering colorado*" is not a good span (because the corresponding $\hat{y}$ is not a part of y):

$$\text{How many rivers } \overbrace{\text{run through the states bordering colorado}}^{state(traverse\_2(stateid('CO'))} ?$$

Each utterance may have several good spans. We define the best span as the shortest good span, with a constraint that the parsing result of the best span should not contain any "ghost" entity. For example, "*the states bordering*" is regarded as a good span, since the preliminary base parser parses it into "$state(traverse\_2(stateid('colorado')))$". We think "$stateid('colorado')$" is a ghost entity, as it is not mentioned in the span. We do not think such spans should be segmented, so we restrict that none of they should be the best span. If a span has no best span, we regard itself as its best span.

For each utterance in the train set $\mathcal{D}$, we use the best span as a pseudo supervision signal for training the utterance segmentation model. We denote all these pseudo supervision signals as $\mathcal{A}$.

Training Objective The overall training objective is:

$$\begin{aligned} \mathcal{J}(\phi, \theta) &= \mathcal{J}_{Seg}(\phi) + \mathcal{J}_{Bp}(\theta) \\ \mathcal{J}_{Seg}(\phi) &= \sum_{(x,\hat{x}) \in \mathcal{A}} \log p(\hat{x}|x) \\ \mathcal{J}_{Bp}(\theta) &= \sum_{(x,y) \in \mathcal{D}} \log p(y|x) + \sum_{(\hat{x},\hat{y}) \in \hat{\mathcal{D}}} \log p(\hat{y}|\hat{x}) \end{aligned} \tag{10}$$

where $\mathcal{J}_{Seg}(\phi)$ is the training objective of the utterance segmentation model, and $\mathcal{J}_{Bp}(\theta)$ is the training objective of the base parser. ϕ and θ refer to learnable parameters in them respectively. $\hat{\mathcal{D}}$ consists of two parts: (1) best spans paired with their partial meaning representations; (2) reduced utterances paired with their partial meaning representations.

Inference

At inference time, we iteratively "segment-and-parse" the utterance. In the k-th iteration, we predict a span $\hat{x}^*$ from $x^{(k)}$ by $\hat{x}^* = \arg\max_{\hat{x}} p(\hat{x}|x^{(k)})$, then parse $\hat{x}^*$ to $\hat{y}^*$ by $\hat{y}^* = \arg\max_{\hat{y}} p(\hat{y}|\hat{x}^*)$. Then, we let $x^{(k+1)} = [\hat{x}^* \mapsto d(\hat{y}^*)]x^{(k)}$ and start the $(k+1)$ -th iteration, until $x^{(k+1)} = x^{(k)}$. Partial meaning representation outputs ($\hat{y}^*$, ...) during these iterations are composed deterministically to form the final meaning representation.

Experiments

Datasets

We conduct experiments on three datasets: GEO (Zelle and Mooney 1996), COMPLEXWEBQUESTIONS (Talmor and Berant 2018) and FORMULAS (a new dataset). Examples of *(utterance, meaning representation)* instances in these three datasets are shown in Table 1.

GEO GEO is a standard semantic parsing benchmark about U.S. geography (Zelle and Mooney 1996), which consists of 880 English questions paired with their meaning representations. These meaning representations can be expressed as four equivalent expressions, namely $\lambda-$calculus, Prolog, SQL, and FunQL (Functional Query Language) (Kate, Wong, and Mooney 2005). Considering the compatibility with the proposed framework, we use FunQL in this paper.

GEO can be splitted into train/test sets in two different ways: (1) *Standard split* (Zettlemoyer and Collins 2005): this split ensures that no natural language question is repeated between the train and test sets; (2) *Compositional split* (Finegan-Dollak et al. 2018): this split ensures that neither questions nor meaning representations(anonymizing named entities) are repeated . We evaluate our framework on both two splits: standard split for comparing with previous systems, and compositional split for measuring the compositional generalization ability. To have a fair comparsion with previous work, we use the preprocessed dataset provided by Dong and Lapata (2016), which does lemmatization for each natural language question and replaces entity mentions by numbered markers. For example, "*How many rivers run through the states bordering colorado*" will be preproccessed to "*How many river run through the state border state_0*".

COMPLEXWEBQUESTIONS This dataset (Talmor and Berant 2018) contains questions paired with SPARQL queries for Freebase (Bollacker et al. 2008). "Complex" means that all questions in this dataset are long and complex, requiring multi-hop reasoning to solve. There are 27,734/3,480/3,475 train/dev/test examples in this dataset. Following the settings in Zhang et al. (2019), we use the *V1.0* version of this dataset, and replace entities in SPARQL queries with placeholders during training and inference. In the test set, 11.9% SPARQL queries (after anonymizing entities, numbers and dates) have never been seen in the train set. We use this subset to measure the compositional generalization ability.

FORMULAS People need to write formulas to manipulate/analyze their tabular spreadsheet data, but it is difficult for them to learn and remember the names and usages of various functions. We want to help people interact with their tabular spreadsheet data using natural language: input natural language commands (e.g., "*show me the largest number in A1:D10*"), and the corresponding formulas (e.g. *MAX(A1:D10)*) will be returned automatically. Therefore, we construct FORMULAS, a semantic parsing dataset containing *(natural language command, spreadsheet formula)* instances. We invited 16 graduate students majoring in computer science (all of them master spreadsheet formulas) as volunteers for manual annotations. They annotate 1,336 formulas (407 single-function formulas and 929 compound formulas) for 30 real-world spreadsheet files. There are total 28 functions used in these formulas, e.g., *SUM*, *MAX*, *FIND*, *CONCAT* and *LOOKUP*. We randomly split this dataset into three parts: 800 instances for training, 268 instances for development, and 268 instances for test. We also replace entity mentions by typed markers, for example, replacing "*A1:D10*" by " *$cellrange$*". In the test set, 29.8% formulas (after anonymizing entities, numbers and dates) have never been seen in the train set. We use this subset to measure the compositional generalization ability.

Implementation Details

Base Parsers Besides the Seq2Seq base parser introduced in Section , we also implement (1) Seq2Tree (Dong and Lapata 2016) for GEO and FORMULAS; (2) Transformer (Vaswani et al. 2017) for COMPLEXWEBQUESTIONS. We do not utilize transformer on GEO/FORMULAS, because they only have hundreds of instances for training. For COMPLEXWEBQUESTION, we switch to transformer in order to have a fair comparison with HSP (state-of-the-art).

Model Configuration We set the dimension of word embedding to 300. In SEQ2SEQ/SEQ2TREE base parsers, we set the dimension of hidden vector to 512. In the utterance segmentation model, the dimension of hidden vector is set to 300. We use the Adam optimizer with default settings (in PYTORCH) and a dropout layer with the rate of 0.5. The training process lasts 100 epochs with batch size 64. For the TRANSFORMER base parser in COMPLEXWEBQUESTIONS, we follow the settings in Vaswani et al. (2017).

Restricted Copy Mechanism HSP (Zhang et al. 2019) incorporates copy mechanism (Gu et al. 2016) to tackle OOV tokens in COMPLEXWEBQUESTIONS and shows its high importance. Observing that most OOV tokens are values (e.g., numbers and dates), we further propose a restricted copy mechanism (denoted as **COPY***): recognize values in utterances via regex-based patterns, and constrain that only these terms can be copied.

Span Substitution In our framework, we need to implement $d(z)$ for each domain, which represents the denotation type of partial meaning representation z (see Equation 4). For GEO and FORMULAS, we directly infer $d(z)$ from z through a syntax-directed translation algorithm (Aho and Ullman 1969): For GEO, $d(z)$ can be "$state$", "$city$", "$river$", "$place$", "$mountain$" or "$lake$"; For FORMULAS, $d(z)$ can be "$number$", "$string$", "$date$", "$bool$", "$cell$" or "$cellrange$". For COMPLEXWEBQUESTIONS, we define

Compositional Semantic Parser	Accuracy
ZC07 (Zettlemoyer and Collins 2007)	86.1%
DCS (Liang, Jordan, and Klein 2011)	87.9%
TISP (Zhao and Huang 2015)	88.9%
Neural Semantic Parser	
SEQ2SEQ (Dong and Lapata 2016)	84.6%
SEQ2TREE (Dong and Lapata 2016)	87.1%
DATARECOMBINE (Jia and Liang 2016)	89.3%
SCANNER* (Cheng et al. 2017)	86.7%
ASN (Rabinovich, Stern, and Klein 2017)	87.1%
COARSE2FINE (Dong and Lapata 2018)	88.2%
GNN* (Shaw et al. 2019)	89.3%
+BERT*	92.5%
SEQ2SEQ*	85.6%
+ PDE*	**90.7%**
SEQ2TREE*	84.2%
+ PDE*	88.9%

Table 2: Accuracies on GEO (standard split). Methods marked with $*$ use FunQL as meaning representations.

$d(z)$ as the first noun phrase in the span, and we compute $[u \mapsto v]y$ (y, u and v are meaning representations) based on SPARQL semantics (rather than the basic string matching-based definition in Section). Moreover, as COMPLEXWEBQUESTIONS contains not only nesting questions but also conjunctive questions, we use a simple but effective heuristic rule to extend our framework: if a span is at the beginning of the utterance, we combine generated partial SPARQL queries using conjunction operation; otherwise we combine them using nesting operation.

Pre-Training In our framework, we need to pre-train the base parser to make it capable to parse simple spans at initial time. For GEO and FORMULAS, we pre-train the base parser using the original training dataset $\mathcal{D}$ which contains many simple instances. In contrast, all questions in COMPLEXWEBQUESTIONS are complex, thus the original training dataset is not suitable for pre-training the base parser. COMPLEXWEBQUESTIONS is constructed through (1) simple instance combination, and (2) crowdsourcing rephrasing. We denote these simple instances as $\mathcal{D}_{seed}$. However, $\mathcal{D}_{seed}$ is also not suitable for pre-training the base parser, since questions in it are lack of diverse linguistic expression (half of them are machine-generated via fixed templates). To tackle this issue, we use a heuristic algorithm to rephrase questions in $\mathcal{D}_{seed}$. Suppose that: $(x, y) \in \mathcal{D}_{seed}$, and there exists $(\tilde{x}, \tilde{y}) \in \mathcal{D}$ and $(x', y') \in \mathcal{D}_{seed}$ such that $\tilde{y}$ is a combination of y and y'. We find a span s in $\tilde{x}$, which maximizes $score(s|x, \tilde{x}, x')$:

$$\begin{aligned} score(s|x, \tilde{x}, x') &= sim(s, x) \\ &\quad + sim([s \mapsto d(y')]\tilde{x}, x') \\ sim(s, s') &= \sum_{w \in s} \max_{w' \in s'} cos(\mathbf{w}, \mathbf{w}') \end{aligned} \tag{11}$$

where $\mathbf{w}$ and $\mathbf{w}'$ represent fasttext word embeddings (Bojanowski et al. 2016) of word w and w', respectively. Then, we treat s as a rephrase of x and add (s, y) to $\mathcal{D}'$. We use $\mathcal{D}'$ to pre-train our base parser.

Method	Accuracy
SCANNER (Cheng et al. 2017)	82.8%
COARSE2FINE (Dong and Lapata 2018)	83.2%
SEQ2SEQ (Dong and Lapata 2016)	80.9%
+ PDE	84.7%
SEQ2TREE (Dong and Lapata 2016)	82.1%
+ PDE	**85.4%**

Table 3: Accuracies on FORMULAS

Method	Accuracy
SEQ2SEQ (Dong and Lapata)	47.3%
SEQ2TREE (Dong and Lapata)	49.7%
POINTERGENERATOR (See, Liu, and Manning)	51.0%
TRANSFORMER (Vaswani et al.)	53.4%
COARSE2FINE (Dong and Lapata)	58.1%
HSP (Zhang et al.)	66.2%
BASEPARSER 1 (SEQ2SEQ+COPY*)	58.4%
+ PDE	64.5%
BASEPARSER 2 (TRANSFORMER+COPY*)	62.8%
+ PDE	**72.2%**

Table 4: Accuracies on COMPLEXWEBQUESTIONS.

Results and Analysis

We denote our framework as PDE (i.e., Parsing via Divide-and-conquEr) and make a comparison against several previously published systems. We use accuracy as the evaluation metric.

GEO and FORMULAS Table 2 presents the results on GEO. Compared with previous work using syntax-aware decoders, PDE performs competitively whereas adopts a relatively simple Seq2Seq model. For the Seq2Seq/Seq2Tree base parser, our segmentation mechanism brings accuracy gains of 5.1% and 4.7%, respectively. Table 3 presents results on FORMULAS, where we observe similar tendencies. For the Seq2Seq/Seq2Tree base parser, our segmentation mechanism brings accuracy gains of 3.8% and 3.3%, respectively.

COMPLEXWEBQUESTIONS Results on COMPLEXWEBQUESTIONS are shown in Table 4. We compare PDE against the state-of-the-art model HSP (Zhang et al. 2019) as well as some other baseline models. The results show that PDE is superior to all baseline models. For BASEPARSER 1 (SEQ2SEQ + COPY*), our segmentation mechanism (+ PDE) achieves an accuracy gain of 6.1% (from 58.4% to 64.5%). For BASEPARSER 2 (TRANSFORMER + COPY*), the accuracy gains brought by PDE is 9.4% (from 62.8% to 72.2%). All these "+ PDE" results in Table 2, 3 and 4 show that: our framework can consistently boost the performance of different neural semantic parsers in different semantic parsing tasks.

Compositional Generalization Ability To demonstrate the compositional generalization ability of PDE, we conduct evaluation on the compositional split for GEO and unseen split for COMPLEXWEBQUESTIONS and FORMULAS (see details in section). In Table 5, we show results on GEO and FORMULAS, together with other two data augmentation baselines. The training data of "+GECA" (Andreas 2019)

Method	GEO	FORMULAS
SEQ2SEQ	63.1%	59.7%
+ GECA	68.2%	64.6%
+ PDE (only DA)	79.1%	66.8%
+ PDE	**81.2%**	**72.7%**
SEQ2TREE	48.7%	63.8%
+ GECA	60.3%	67.5%
+ PDE (only DA)	79.4%	74.2%
+ PDE	**80.5%**	**77.6%**

Table 5: Evaluation of compositional generalization ability on GEO (compositional split) and FORMULAS (unseen subset).

Model	SEEN (88.1%)	UNSEEN (11.9%)
HSP	69.8%	40.8%
BASEPARSER 2	67.7%	27.1%
+ PDE	**74.4%**	**56.3%**

Table 6: Evaluation of compositional generalization ability on COMPLEXWEBQUESTIONS (unseen subset).

is augmentated by the protocol which seeks to provide a compositional inductive bias in sequence models. To further validate the effectiveness of PDE, we conduct another baseline "+only DA", which only trained on the mixture of real data and pseudo supervision (generated by the PDE). Compared to those baselines, PDE provides explicit alignments between spans and meaning representations, thus achieving the best performance. For COMPLEXWEBQUESTIONS, we make a comparison with HSP (Zhang et al. 2019) and show the results in Figure 3. As we can see, the results on all three datasets consistently prove that PDE brings significant accuracy gains (GEO 63.1 $\rightarrow$ 81.2, FORMULAS 59.7 $\rightarrow$ 72.7, COMPLEXWEBQUESTIONS 27.1 $\rightarrow$ 56.3) on splits which require compositional generalization.

Impact of Different Question Types As there are four question types (conjunction, nesting, superlative, and comparative) in COMPLEXWEBQUESTIONS, we further investigate the impact of question types on model performances (Table 7). The segmentation mechanism improves the accuracy by a large margin on CONJ/NEST/COMPAR questions, while it is not good at dealing with SUP questions.

Qualitative Analysis of Segmentation To verify whether PDE can provide meaningful segmentations of complex utterances, we conduct a human evaluation on a sample of 50 test questions: given spans predicted by PDE, three non-native, but fluent-English speakers are asked to label whether the segmentation is meaningful or not. These questions are all from COMPLEXWEBQUESTIONS dataset. PDE performs well on segmenting 41 out of 50 questions, which indicates the "segment-and-parse" mechanism can explicitly capture some meaningful compositional semantics (Figure 3). We observe three types of error cases:

1. *Conjunction Scope.* For example, in E_1 (the first error case in Figure 3), the segmentation model mistakenly regards that the scope of conjunction "*and*" is the whole question, while the real scope is "*cigarettes and chocolate milk*". In our future work, we expect to solve this problem through incorporating syntactic information (e.g., syntax tree) into the utterance segmentation model.
2. *Shared Predicates.* For example, in E_2, the reduced question is "*movie and Erick Estrada?*", which lacks a predicate ("*stars*"). In our future work, we expect to solve this problem through measuring completeness of reduced utterances and accordingly performing span substitution.
3. *Superlative Constraint.* For example, E_3: "*When did the Red Sox win their pennant*" constrained by a superlative "*first*", which cannot be linearly segmented. To address this problem, we need to guide PDE not to segment such utterances (through predicting $s = x$) and leave them to the base parser. This is a general principle for dealing with complex language phenomena that exposes the limitation of linear segmentation in complex utterances. We also leave this to future work.

Model	CONJ (41.5%)	NEST (47.6%)	SUP (5.0%)	COMPAR (6.0%)
HSP	66.2%	68.3%	**68.3%**	46.6%
BASEPARSER 2	64.8%	62.2%	53.2%	62.2%
+ PDE	**71.2%**	**75.7%**	49.1%	**70.3%**

Table 7: Accuracies of different question types on COMPLEXWEBQUESTIONS dataset.

Label	Count	Examples
Correct	41	What other films have **the actor who played Periwinkle** been in?
		What famous facility in Charlotte is home to the Carolina Cougar?
Error	9	**Who is the person nominating for an award for cigarettes** and chocolate milk currently married to?
		Which movie stars both Mario Lopez and Erick Estrada?
		When did the Red Sox win their first pennant?

Figure 3: Human evaluation of utterance segmentation on a random sample of 50 test questions.

Conclusion

In this paper, we propose a novel framework for boosting neural semantic parsers via iterative utterance segmentation. The insight is a bottom-up divide-and-conquer mechanism, which significantly improves the compositional generalization ability and interpretability of neural semantic parsers. Considering the usual absence of labeled data for utterance segmentation, we propose a cooperative training method to tackle this problem. Experimental results show that our framework consistently improves the performance of different neural semantic parsers across tasks.

In the future, we plan to improve the robustness of our framework for various complex language phenomena. We also plan to apply this framework to more semantic parsing tasks such as text-to-SQL and text-to-code.

References

Aho, A. V.; and Ullman, J. D. 1969. Syntax Directed Translations and the Pushdown Assembler. *J. Comput. Syst. Sci.* 3(1): 37–56. ISSN 0022-0000. doi:10.1016/S0022-0000(69)80006-1. URL http://dx.doi.org/10.1016/S0022-0000(69)80006-1.

Andreas, J. 2019. Good-enough compositional data augmentation. *arXiv preprint arXiv:1904.09545* .

Berant, J.; and Liang, P. 2015. Imitation Learning of Agenda-based Semantic Parsers. *Transactions of the Association for Computational Linguistics* 3: 545–558. doi:10.1162/tacl_a_00157. URL https://www.aclweb.org/anthology/Q15-1039.

Bojanowski, P.; Grave, E.; Joulin, A.; and Mikolov, T. 2016. Enriching Word Vectors with Subword Information. *arXiv preprint arXiv:1607.04606* .

Bollacker, K.; Evans, C.; Paritosh, P.; Sturge, T.; and Taylor, J. 2008. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 1247–1250. AcM.

Cheng, J.; Reddy, S.; Saraswat, V.; and Lapata, M. 2017. Learning Structured Natural Language Representations for Semantic Parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 44–55. Vancouver, Canada: Association for Computational Linguistics. doi:10.18653/v1/P17-1005. URL https://www.aclweb.org/anthology/P17-1005.

Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

Dong, L.; and Lapata, M. 2016. Language to Logical Form with Neural Attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 33–43. Berlin, Germany: Association for Computational Linguistics. doi:10.18653/v1/P16-1004. URL https://www.aclweb.org/anthology/P16-1004.

Dong, L.; and Lapata, M. 2018. Coarse-to-Fine Decoding for Neural Semantic Parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 731–742. Melbourne, Australia: Association for Computational Linguistics. doi:10.18653/v1/P18-1068. URL https://www.aclweb.org/anthology/P18-1068.

Finegan-Dollak, C.; Kummerfeld, J. K.; Zhang, L.; Ramanathan, K.; Sadasivam, S.; Zhang, R.; and Radev, D. 2018. Improving Text-to-SQL Evaluation Methodology. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 351–360. Melbourne, Australia: Association for Computational Linguistics. doi:10.18653/v1/P18-1033. URL https://www.aclweb.org/anthology/P18-1033.

Gu, J.; Lu, Z.; Li, H.; and Li, V. O. 2016. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1631–1640. Berlin, Germany: Association for Computational Linguistics. doi:10.18653/v1/P16-1154. URL https://www.aclweb.org/anthology/P16-1154.

Guo, J.; Zhan, Z.; Gao, Y.; Xiao, Y.; Lou, J.-G.; Liu, T.; and Zhang, D. 2019. Towards Complex Text-to-SQL in Cross-Domain Database with Intermediate Representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 4524–4535. Florence, Italy: Association for Computational Linguistics. doi:10.18653/v1/P19-1444. URL https://www.aclweb.org/anthology/P19-1444.

Jia, R.; and Liang, P. 2016. Data Recombination for Neural Semantic Parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 12–22. Berlin, Germany: Association for Computational Linguistics. doi:10.18653/v1/P16-1002. URL https://www.aclweb.org/anthology/P16-1002.

Kalyanpur, A.; Patwardhan, S.; Boguraev, B.; Lally, A.; and Chu-Carroll, J. 2012. Fact-based question decomposition in DeepQA. *IBM Journal of Research and Development* 56(3.4): 13–1.

Kate, R. J.; Wong, Y. W.; and Mooney, R. J. 2005. Learning to transform natural to formal languages. In *AAAI*, 1062–1068.

Keysers, D.; Schärli, N.; Scales, N.; Buisman, H.; Furrer, D.; Kashubin, S.; Momchev, N.; Sinopalnikov, D.; Stafiniak, L.; Tihon, T.; Tsarkov, D.; Wang, X.; van Zee, M.; and Bousquet, O. 2020. Measuring Compositional Generalization: A Comprehensive Method on Realistic Data. In *International Conference on Learning Representations*. URL https://openreview.net/forum?id=SygcCnNKwr.

Lake, B.; and Baroni, M. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *International Conference on Machine Learning*, 2873–2882. PMLR.

Liang, P.; Jordan, M.; and Klein, D. 2011. Learning Dependency-Based Compositional Semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 590–599. Portland, Oregon, USA: Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P11-1060.

Ling, W.; Blunsom, P.; Grefenstette, E.; Hermann, K. M.; Kočiský, T.; Wang, F.; and Senior, A. 2016. Latent Predictor Networks for Code Generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 599–609. Berlin, Germany: Association for Computational Linguistics. doi:10.18653/v1/P16-1057. URL https://www.aclweb.org/anthology/P16-1057.

Luong, T.; Pham, H.; and Manning, C. D. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical*

Methods in Natural Language Processing, 1412–1421. Lisbon, Portugal: Association for Computational Linguistics. doi:10.18653/v1/D15-1166. URL https://www.aclweb.org/anthology/D15-1166.

Min, S.; Zhong, V.; Zettlemoyer, L.; and Hajishirzi, H. 2019. Multi-hop Reading Comprehension through Question Decomposition and Rescoring. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 6097–6109. Florence, Italy: Association for Computational Linguistics. doi:10.18653/v1/P19-1613. URL https://www.aclweb.org/anthology/P19-1613.

Pasupat, P.; Gupta, S.; Mandyam, K.; Shah, R.; Lewis, M.; and Zettlemoyer, L. 2019. Span-based Hierarchical Semantic Parsing for Task-Oriented Dialog. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 1520–1526.

Pasupat, P.; and Liang, P. 2015. Compositional Semantic Parsing on Semi-Structured Tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 1470–1480. Beijing, China: Association for Computational Linguistics. doi:10.3115/v1/P15-1142. URL https://www.aclweb.org/anthology/P15-1142.

Pelletier, F. J. 1994. The Principle of Semantic Compositionality. *Topoi* 13(1): 11–24. ISSN 1572-8749. doi:10.1007/BF00763644. URL https://doi.org/10.1007/BF00763644.

Qi, P.; Lin, X.; Mehr, L.; Wang, Z.; and Manning, C. D. 2019. Answering Complex Open-domain Questions Through Iterative Query Generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2590–2602. Hong Kong, China: Association for Computational Linguistics. doi:10.18653/v1/D19-1261. URL https://www.aclweb.org/anthology/D19-1261.

Rabinovich, M.; Stern, M.; and Klein, D. 2017. Abstract Syntax Networks for Code Generation and Semantic Parsing. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1139–1149. Vancouver, Canada: Association for Computational Linguistics. doi:10.18653/v1/P17-1105. URL https://www.aclweb.org/anthology/P17-1105.

See, A.; Liu, P. J.; and Manning, C. D. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1073–1083. Vancouver, Canada: Association for Computational Linguistics. doi:10.18653/v1/P17-1099. URL https://www.aclweb.org/anthology/P17-1099.

Shaw, P.; Massey, P.; Chen, A.; Piccinno, F.; and Altun, Y. 2019. Generating Logical Forms from Graph Representations of Text and Entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 95–106. Florence, Italy: Association for Computational Linguistics. doi:10.18653/v1/P19-1010. URL https://www.aclweb.org/anthology/P19-1010.

Talmor, A.; and Berant, J. 2018. The Web as a Knowledge-Base for Answering Complex Questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 641–651. New Orleans, Louisiana: Association for Computational Linguistics. doi:10.18653/v1/N18-1059. URL https://www.aclweb.org/anthology/N18-1059.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Yin, P.; and Neubig, G. 2017. A Syntactic Neural Model for General-Purpose Code Generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 440–450. Vancouver, Canada: Association for Computational Linguistics. doi:10.18653/v1/P17-1041. URL https://www.aclweb.org/anthology/P17-1041.

Zelle, J. M.; and Mooney, R. J. 1996. Learning to Parse Database Queries Using Inductive Logic Programming. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence - Volume 2*, AAAI'96, 1050–1055. AAAI Press. ISBN 0-262-51091-X. URL http://dl.acm.org/citation.cfm?id=1864519.1864543.

Zettlemoyer, L.; and Collins, M. 2007. Online Learning of Relaxed CCG Grammars for Parsing to Logical Form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 678–687. Prague, Czech Republic: Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D07-1071.

Zettlemoyer, L. S.; and Collins, M. 2005. Learning to Map Sentences to Logical Form: Structured Classification with Probabilistic Categorial Grammars. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, UAI'05, 658–666. Arlington, Virginia, United States: AUAI Press. ISBN 0-9749039-1-4. URL http://dl.acm.org/citation.cfm?id=3020336.3020416.

Zhang, H.; Cai, J.; Xu, J.; and Wang, J. 2019. Complex Question Decomposition for Semantic Parsing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 4477–4486. Florence, Italy: Association for Computational Linguistics. doi:10.18653/v1/P19-1440. URL https://www.aclweb.org/anthology/P19-1440.

Zhao, K.; and Huang, L. 2015. Type-Driven Incremental Semantic Parsing with Polymorphism. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1416–1421. Denver, Colorado: Association for Computational Linguistics. doi:10.3115/v1/N15-1162. URL https://www.aclweb.org/anthology/N15-1162.