A Split-and-Recombine Approach for Follow-up Query Analysis

Qian Liu[†*], **Bei Chen**[§], **Haoyan Liu**[◇*], **Jian-Guang Lou**[§], **Lei Fang**[§],
Bin Zhou[†], **Dongmei Zhang**[§]
[†]State Key Laboratory of Virtual Reality Technology and Systems,
School of Computer Science and Engineering, Beihang University, China
[◇]State Key Lab of Software Development Environment, Beihang University, China
[§]Microsoft Research, Beijing, China
[†◇]{qian.liu, haoyan.liu, zhoubin}@buaa.edu.cn;
[§]{beichen, jlou, leifa, dongmeiz}@microsoft.com

Abstract

Context-dependent semantic parsing has proven to be an important yet challenging task. To leverage the advances in context-independent semantic parsing, we propose to perform follow-up query analysis, aiming to restate context-dependent natural language queries with contextual information. To accomplish the task, we propose STAR, a novel approach with a well-designed two-phase process. It is parser-independent and able to handle multifarious follow-up scenarios in different domains. Experiments on the FollowUp dataset show that STAR outperforms the state-of-the-art baseline by a large margin of nearly 8%. The superiority on parsing results verifies the feasibility of follow-up query analysis. We also explore the extensibility of STAR on the SQA dataset, which is very promising.

1 Introduction

Recently, Natural Language Interfaces to Databases (NLIDB) has received considerable attention, as they allow users to query databases by directly using natural language. Current studies mainly focus on context-independent semantic parsing, which translates a single natural language sentence into its corresponding executable form (e.g. Structured Query Language) and retrieves the answer from databases regardless of its context. However, context does matter in real world applications. Users tend to issue queries in a coherent way when communicating with NLIDB. For example, after the query "How much money has Smith earned?" (**Precedent Query**), users may pose another query by simply asking "How about Bill Collins?" (**Follow-up Query**) instead of the complete "How much money has Bill Collins earned?" (**Restated Query**). Therefore, contextual information is essential for more accurate and robust semantic parsing, namely context-dependent semantic parsing.

Compared with context-independent semantic parsing, context-dependent semantic parsing has received less attention. Several attempts include a statistical model with parser trees (Miller et al., 1996), a linear model with context-dependent logical forms (Zettlemoyer and Collins, 2009) and a sequence-to-sequence model (Suhr et al., 2018). However, all these methods cannot apply to different domains, since the ATIS dataset (Dahl et al., 1994) they rely on is domain-specific. A search-based neural method DynSP* arises along with the SequentialQA (SQA) dataset (Iyyer et al., 2017), which takes the first step towards cross-domain context-dependent semantic parsing. Nevertheless, DynSP* focuses on dealing with relatively simple scenarios. All the aforementioned methods design context-dependent semantic parser from scratch. Instead, inspired by Liu et al. (2019), we propose to directly leverage the technical advances in context-independent semantic parsing. We define *follow-up query analysis* as restating the follow-up queries using contextual information in natural language, then the restated queries can be translated to the corresponding executable forms by existing context-independent parsers. In this way, we boost the performance of context-dependent semantic parsing.

In this paper, we focus on follow-up query analysis and present a novel approach. The main idea is to decompose the task into two phases by introducing a learnable intermediate structure *span*: two queries first get split into several spans, and then undergo the recombination process. As no intermediate annotation is involved, we design re-

*Work done during an internship at Microsoft Research.

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 5316–5326,
Hong Kong, China, November 3–7, 2019.

wards to jointly train the two phases by applying reinforcement learning (RL) (Sutton and Barto, 1998). Our major contributions are as follows:

- We propose a novel approach, named **S**pli**T**-**A**nd-**R**ecombine (STAR)[1], to restate follow-up queries via two phases. It is parser-independent and can be seamlessly integrated with existing context-independent semantic parsers.
- We conduct experiments on the FollowUp dataset (Liu et al., 2019), which covers multifarious cross-domain follow-up scenarios. The results demonstrate that our approach significantly outperforms the state-of-the-art baseline.
- We redesign the recombination process and extend STAR to the SQA dataset, where the annotations are answers. Experiments show promising results, that demonstrates the extensibility of our approach.

2 Methodology

In this section, we first give an overview of our proposed method with the idea of two-phase process, then introduce the two phases in turn.

2.1 Overview of Split-And-Recombine

Let $\mathbf{x} = (x_1, \ldots, x_n)$, $\mathbf{y} = (y_1, \ldots, y_m)$ and $\mathbf{z} = (z_1, \ldots, z_l)$ denote the precedent query, follow-up query and restated query respectively, each of which is a natural language sentence. Our goal is to interpret the follow-up query $\mathbf{y}$ with its precedent query $\mathbf{x}$ as context, and generate the corresponding restated query $\mathbf{z}$. The restated query has the same meaning with the follow-up query, but it is complete and unambiguous to facilitate better downstream parsing. Formally, given the pair $(\mathbf{x}, \mathbf{y})$, we aim to learn a model $P_{\text{model}}(\mathbf{z}|\mathbf{x}, \mathbf{y})$ and maximize the objective:

$$\mathcal{L} = \mathbb{E}_{(\mathbf{x},\mathbf{y},\mathbf{z})\sim\mathcal{D}}[\log P_{\text{model}}(\mathbf{z}|\mathbf{x}, \mathbf{y})], \quad (1)$$

where $\mathcal{D}$ represents the set of training data. As to $P_{\text{model}}(\mathbf{z}|\mathbf{x}, \mathbf{y})$, since $\mathbf{z}$ always overlaps a great with $\mathbf{x}$ and $\mathbf{y}$, it is intuitively more straightforward to find a way to merge $\mathbf{x}$ and $\mathbf{y}$. To this end, we design a two-phase process and present a novel approach STAR to perform follow-up query analysis with reinforcement learning.

A concrete example of the two-phase process is shown in Figure 1. Phase I is to **Split** input queries

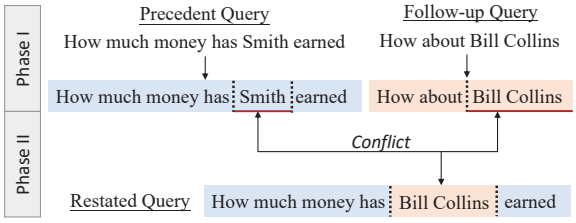

Figure 1: The two-phase process of an example from the FollowUp dataset (More real cases of diverse follow-up scenarios can be found in Table 3).

into several spans. For example, the precedent query is split into 3 spans: "How much money has", "Smith" and "earned". Let q denote a kind of way to split $(\mathbf{x}, \mathbf{y})$, then Phase I can be formulated as $P_{\text{split}}(q|\mathbf{x}, \mathbf{y})$. Phase II is to **Recombine** the spans by finding out the most probable *conflicting way*, and generating the final output by restatement, denoted as $P_{\text{rec}}(\mathbf{z}|q)$. Two spans being conflicting means they are semantically similar. For example, "Smith" conflicts with "Bill Collins". A conflicting way contains all conflicts between the precedent and follow-up spans. Backed by the two-phase idea of splitting and recombination, the overall likelihood of generating $\mathbf{z}$ given $\mathbf{x}, \mathbf{y}$ is:

$$P_{\text{model}}(\mathbf{z}|\mathbf{x}, \mathbf{y}) = \sum_{q\in\mathcal{Q}} P_{\text{split}}(q|\mathbf{x}, \mathbf{y}) P_{\text{rec}}(\mathbf{z}|q), \quad (2)$$

where $\mathcal{Q}$ represents the set of all possible ways to split $(\mathbf{x}, \mathbf{y})$. Due to the lack of annotations for splitting and recombination, it is hard to directly perform supervised learning. Inspired by Liang et al. (2017), we employ RL to optimize P_{model}. Denoting the predicted restated query by $\tilde{\mathbf{z}}$, simplifying $\mathbb{E}_{(\mathbf{x},\mathbf{y},\mathbf{z})\sim\mathcal{D}}$ as $\mathbb{E}$, the goal of the RL training is to maximize following objective:

$$\mathcal{L}_{\text{rl}} = \mathbb{E}[\sum_{\tilde{\mathbf{z}}\in\mathcal{Z}}\sum_{q\in\mathcal{Q}} P_{\text{split}}(q|\mathbf{x}, \mathbf{y}) P_{\text{rec}}(\tilde{\mathbf{z}}|q) r(\mathbf{z}, \tilde{\mathbf{z}})], \quad (3)$$

where $\mathcal{Z}$ is the space of all restated query candidates and r represents the reward defined by comparing $\tilde{\mathbf{z}}$ and the annotation $\mathbf{z}$. However, the overall candidate space $\mathcal{Q}\times\mathcal{Z}$ is vast, making it impossible to exactly maximize $\mathcal{L}_{\text{rl}}$. The most straightforward usage of the REINFORCE algorithm (Williams, 1992), sampling both q and $\tilde{\mathbf{z}}$, also poses challenges for learning. To alleviate the problem, we propose to sample q and enumerate all candidate $\tilde{\mathbf{z}}$ after q is determined. It could shrink the sampling space with an acceptable computational cost, which will be discussed in Sec-

[1]Code is available at http://github.com/microsoft/EMNLP2019-Split-And-Recombine.

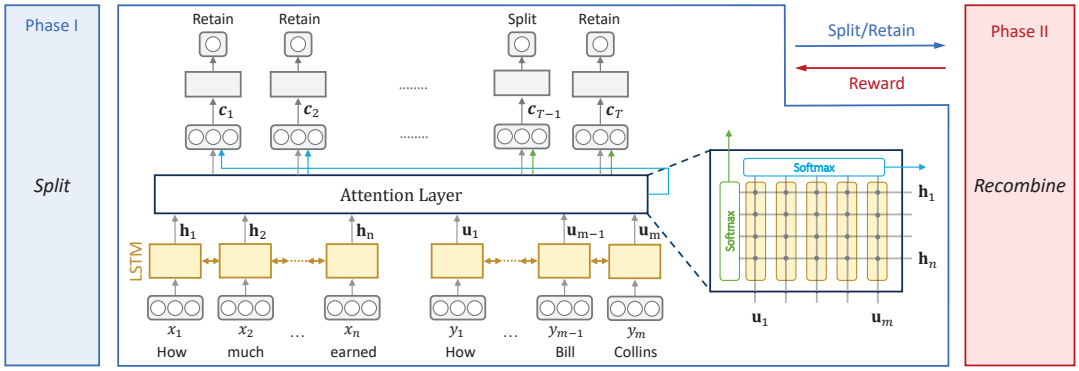

Figure 2: The overview of STAR with two phases.

tion 3.2.2. Thus the problem turns to design a reward function $R(q, \mathbf{z})$ to evaluate q and guide the learning. To achieve it, we reformulate Equation 3 as:

$$\mathcal{L}_{\text{rl}} = \mathbb{E}[\sum_{q \in \mathcal{Q}} P_{\text{split}}(q|\mathbf{x}, \mathbf{y}) \sum_{\tilde{\mathbf{z}} \in \mathcal{Z}} P_{\text{rec}}(\tilde{\mathbf{z}}|q) r(\mathbf{z}, \tilde{\mathbf{z}})], \quad (4)$$

and set the $R(q, \mathbf{z})$ as:

$$\sum_{\tilde{\mathbf{z}} \in \mathcal{Z}} P_{\text{rec}}(\tilde{\mathbf{z}}|q) r(\mathbf{z}, \tilde{\mathbf{z}}). \quad (5)$$

The overview of STAR is summarized in Figure 2. Given $\mathbf{x}, \mathbf{y}$, during training of Phase I (in blue), we fix P_{rec} to provide the reward $R(q, \mathbf{z})$, then P_{split} can be learnt by the REINFORCE algorithm. During training of Phase II (in red), we fix P_{split} and utilize it to generate q, P_{rec} is trained to maximize Equation 5. In this way, P_{split} and P_{rec} can be jointly trained. The details are introduced below.

2.2 Phase I: Split

As mentioned above, fixed P_{rec}, Phase I updates P_{split}, the Split Neural Network (**SplitNet**). Taking the precedent query and follow-up query as input, as shown in Figure 2, splitting spans can be viewed as a sequence labeling problem over input. For each word, SplitNet outputs a label *Split* or *Retain*, indicating whether a split operation will be performed after the corresponding word. A label sequence uniquely identifies a way of splitting $(\mathbf{x}, \mathbf{y})$, mentioned as q in Section 2.1. Figure 3 gives an example on the bottom. In the precedent query, two split operations are performed after "has" and "Smith" , since their labels are *Split*.

2.2.1 Split Neural Network

Intuitively, only after obtaining information from both the precedent query and follow-up query can SplitNet get to know the reasonable way to split spans. Inspired by BiDAF (Seo et al., 2017), we apply a bidirectional attention mechanism to capture the interrelations between the two queries.

Embedding Layer We consider embedding in three levels: character, word and sentence, respectively denoted as ϕ_c, ϕ_w and ϕ_s. Character-level embedding maps each word to a vector in a high-dimensional space using Convolutional Neural Networks (Kim, 2014). Word-level embedding is initialized using GloVe (Pennington et al., 2014), and then it is updated along with other parameters. Sentence-level embedding is a one-hot vector designed to distinguish between precedent and follow-up queries. Then, the overall embedding function is $\phi = [\phi_c; \phi_w; \phi_s]$.

Context Layer On top of the embedding layer, Bidirectional Long Short-Term Memory Network (BiLSTM) (Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997) is applied to capture contextual information within one query. For word $x_i (i = 1, \ldots, n)$ in the precedent query $\mathbf{x}$, the hidden state $\mathbf{h}_i = [\overrightarrow{\mathbf{h}}_i; \overleftarrow{\mathbf{h}}_i]$ is computed, where the forward hidden state is:

$$\overrightarrow{\mathbf{h}}_i = \overrightarrow{\text{LSTM}}(\phi(x_i); \overrightarrow{\mathbf{h}}_{i-1}). \quad (6)$$

Similarly, a hidden state $\mathbf{u}_j$ is computed for word $y_j (j = 1, \ldots, m)$. The BiLSTMs for $\mathbf{x}$ and $\mathbf{y}$ share the same parameters.

Attention Layer The interrelations between the precedent and follow-up queries are captured via attention layer. Let $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n]$ and $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_m]$ denote the hidden states of two queries respectively, the similarity matrix is:

$$\mathbf{A} = \cos(\mathbf{H}^\top \mathbf{U}), \quad (7)$$

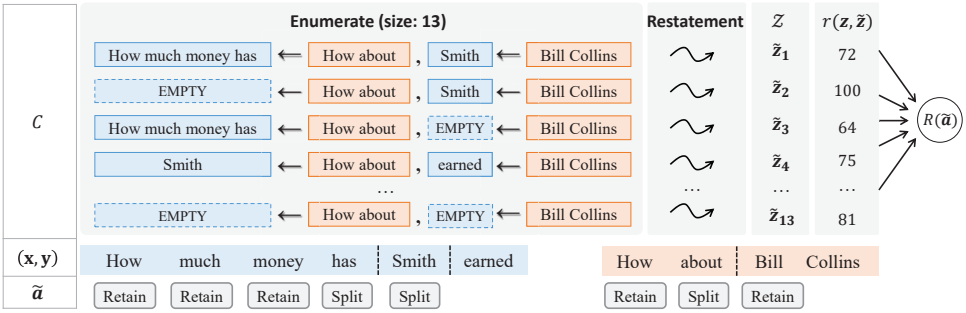

Figure 3: Illustration of reward computation in Phase II.

where $\mathbf{A} \in \mathbb{R}^{n \times m}$ and the entry $A_{i,j}$ represents the similarity between words x_i and y_j. Then the softmax function is used to obtain the precedent-to-follow **(P2F)** attention and the follow-to-precedent **(F2P)** attention. P2F attention represents y_j using the similarities between y_j and every word in $\mathbf{x}$. Specifically, let $\mathbf{f}_j = \mathrm{softmax}(\mathbf{A}_{:,j})$, where $\mathbf{f}_j \in \mathbb{R}^n$ denotes the attention weights on $\mathbf{x}$ according to y_j. Then y_j can be represented by a precedent-aware vector $\tilde{\mathbf{u}}_j = \sum_{k=1}^{n} \mathbf{f}_j[k] \cdot \mathbf{h}_k$. Similarly, F2P attention computes the attention weights on $\mathbf{y}$ according to x_i, and represents x_i as $\tilde{\mathbf{h}}_i$.

Output Layer Combining the outputs of the context layer and the attention layer, we design the final hidden state as follows:

$$\mathbf{c}^{x_i} = [\mathbf{h}_i;\ \mathbf{h}_i \circ \tilde{\mathbf{h}}_i;\ \mathbf{h}_{i+1} \circ \tilde{\mathbf{h}}_{i+1}], \quad (8)$$

$$\mathbf{c}^{y_j} = [\mathbf{u}_j;\ \mathbf{u}_j \circ \tilde{\mathbf{u}}_j;\ \mathbf{u}_{j+1} \circ \tilde{\mathbf{u}}_{j+1}], \quad (9)$$

where $i \in \{1, \ldots, n-1\}, j \in \{1, \ldots, m-1\}$ and $\circ$ denotes element-wise multiplication (Lee et al., 2017). Let $\mathbf{c} = (\mathbf{c}_t)_{t=1}^{T} = (\mathbf{c}^{x_1}, ..., \mathbf{c}^{x_{n-1}}, \mathbf{c}^{y_1}, ..., \mathbf{c}^{y_{m-1}})$ denote the final hidden state sequence. At each position t, the probability of *Split* is $\sigma(\mathbf{W} * \mathbf{c}_t + b)$, where σ denotes the *sigmoid* function and $\{\mathbf{W}, b\}$ denotes the parameters.

2.2.2 Training

It is difficult to train RL model from scratch. Therefore, we propose to initialize SplitNet via pre-training, and then use reward to optimize it.

Pre-training We obtain the pre-training annotation $\mathbf{a}$ by finding the common substrings between $(\mathbf{x}, \mathbf{y})$ and $\mathbf{z}$. One $\mathbf{a}$ is a label sequence, each of which is *Split* or *Retain*. Given the pre-training data set $\mathcal{D}_{\mathrm{pre}}$ whose training instance is as $(\mathbf{x}, \mathbf{y}, \mathbf{a})$, the objective function of pre-training is:

$$\mathcal{L}_{\mathrm{pre}}(\theta) = \mathbb{E}_{(\mathbf{x},\mathbf{y},\mathbf{a}) \sim \mathcal{D}_{\mathrm{pre}}} [\log p_\theta(\mathbf{a} | \mathbf{x}, \mathbf{y})], \quad (10)$$

where θ is the parameter of SplitNet.

Policy Gradient After pre-training, we treat the label sequence as a variable $\tilde{\mathbf{a}}$. The reward $R(\tilde{\mathbf{a}}, \mathbf{z})$ (details in Section 2.3) is used to optimize the parameter θ with policy gradient methods (Sutton et al., 1999). SplitNet is trained to maximize the following objective function:

$$\mathcal{L}_{\mathrm{rl}}(\theta) = \mathbb{E}_{(\mathbf{x},\mathbf{y},\mathbf{z}) \sim \mathcal{D}} [\mathbb{E}_{\tilde{\mathbf{a}} \sim p_\theta(\tilde{\mathbf{a}} | \mathbf{x}, \mathbf{y})} R(\tilde{\mathbf{a}}, \mathbf{z})]. \quad (11)$$

In practice, REINFORCE algorithm (Williams, 1992) is applied to approximate Equation 11 via sampling $\tilde{\mathbf{a}}$ from $p_\theta(\tilde{\mathbf{a}} | \mathbf{x}, \mathbf{y})$ for M times, where M is a hyper-parameter representing the sample size. Furthermore, subtracting a baseline (Weaver and Tao, 2001) on $R(\tilde{\mathbf{a}}, \mathbf{z})$ is also applied to reduce variance. The final objective function is as follows:

$$\mathcal{L}_{\mathrm{rl}}(\theta) = \mathbb{E}_{(\mathbf{x},\mathbf{y},\mathbf{z}) \sim \mathcal{D}} \Big[\sum_{i=1}^{M} p_\theta(\tilde{\mathbf{a}}_i | \mathbf{x}, \mathbf{y}) \big(R(\tilde{\mathbf{a}}_i, \mathbf{z}) - \bar{R} \big) \Big],$$

$$\text{where } \bar{R} = \frac{1}{M} \sum_{i=1}^{M} R(\tilde{\mathbf{a}}_i, \mathbf{z}). \quad (12)$$

2.3 Phase II: Recombine

Here we present Phase II with two questions: (1) Receiving the sampled label sequence $\tilde{\mathbf{a}}$, how to compute its reward $R(\tilde{\mathbf{a}}, \mathbf{z})$; (2) How to do training and inference for P_{rec}.

2.3.1 Reward Computation

Receiving the label sequence $\tilde{\mathbf{a}}$, we first enumerate all conflicting way candidates. Following the example in Figure 3, once we get a deterministic $\tilde{\mathbf{a}}$, the split of $(\mathbf{x}, \mathbf{y})$ is uniquely determined. Here $\mathbf{x}$ and $\mathbf{y}$ are split into 3 and 2 spans respectively. Treating spans as units, we enumerate all conflicting way candidates methodically. We act up to the one-to-one conflicting principle, which means a

span either has no conflict (denoted as *EMPTY*) or has only one conflict with a span in another query. Let $\mathcal{C}$ denote the set of all conflicting way candidates, the size of which is 13 in Figure 3.

For each conflicting way, we deterministically generate a restated query via the process named **Restatement**. In general, we simply replace spans in the precedent query with their conflicting spans to generate the restated query. For example, in Figure 3, the first one in $\mathcal{C}$ is restated as "How about Bill Collins earned". For spans in the follow-up query, if they contain column names or cell values and do not have any conflict, they are appended to the tail of the precedent query. It is designed to remedy the sub-query situation where there is no conflict (e.g. "Which opponent received over 537 attendance" and "And which got the result won 5-4"). Specially, if a span in the follow-up query contains a pronoun, we will in reverse replace it with its conflicting span to obtain the restated query.

Finally, the reward can be computed. Here we use BLEU and SymAcc[2] to build the reward function, expanding $r(\mathbf{z}, \tilde{\mathbf{z}})$ in Equation 5 as:

$$r(\mathbf{z}, \tilde{\mathbf{z}}) = \alpha \cdot \text{BLEU}(\mathbf{z}, \tilde{\mathbf{z}}) + \beta \cdot \text{SymAcc}(\mathbf{z}, \tilde{\mathbf{z}}), \quad (13)$$

where $\alpha, \beta > 0$ and $\alpha + \beta = 1$. The reward for $\tilde{\mathbf{a}}$ can be obtained using Equation 5.

2.3.2 Training and Inference

Besides the reward computation, the recombination model P_{rec} needs to be trained to maximize Equation 5. To achieve this, we define a conflicting probability matrix $\mathbf{F} \in \mathbb{R}^{N_x \times N_y}$, where N_x and N_y denote the number of spans in $\mathbf{x}$ and $\mathbf{y}$ respectively. The entry $\mathbf{F}_{u,v}$, the conflicting probability between the u-th span in $\mathbf{x}$ and the v-th span in $\mathbf{y}$, is obtained by normalizing the cosine similarity between their representations. Here the span representation is the subtraction representation (Wang and Chang, 2016; Cross and Huang, 2016), which means that span $(x_i, \ldots, x_k)$ is represented by $[\overrightarrow{\mathbf{h}_k} - \overrightarrow{\mathbf{h}_i}; \overleftarrow{\mathbf{h}_i} - \overleftarrow{\mathbf{h}_k}]$ from the same BiLSTM in the context layer in Section 2.2.1. Given a conflicting way denoted as $\tilde{c} \in \mathcal{C}$, the probability of generating its corresponding $\tilde{\mathbf{z}}$ can be written as the multiplication over $g(u, v)$:

$$P_{\text{rec}}(\tilde{\mathbf{z}}|\tilde{\mathbf{a}}) = P(\tilde{c}|\mathbf{F}) = \prod_{u=1}^{N_x} \prod_{v=1}^{N_y} g(u, v), \quad (14)$$

where $g(u, v) = \mathbf{F}_{u,v}$ if the u-th span in $\mathbf{x}$ conflicts with the v-th span in $\mathbf{y}$; otherwise, $g(u, v) = 1 - \mathbf{F}_{u,v}$. With the above formulation, we can maximize Equation 5 through automatic differentiation. To reduce the computation, we only maximize $P_{\text{rec}}(\tilde{\mathbf{z}}^*|\tilde{\mathbf{a}})$, the near-optimal solution to Equation 5, where $\tilde{\mathbf{z}}^* = \arg\max_{\tilde{\mathbf{z}} \in \mathcal{Z}}(r(\mathbf{z}, \tilde{\mathbf{z}}))$ denotes the best predicted restated query so far.

Guided by the golden restated query $\mathbf{z}$, in training, we find out $\tilde{\mathbf{z}}^*$ by computing the reward of each candidate. However in inference, where there is no golden restate query, we can only obtain $\tilde{\mathbf{z}}^*$ from $\mathbf{F}$. Specially, for the v-th span in the follow-up query, we find $u^* = \arg\max_u \mathbf{F}_{u,v}$. That means, compared to other spans in the precedent query, the u^*-th span has the highest probability to conflict with the v-th span in the follow-up query. Moreover, similar to Lee et al. (2017), if $\mathbf{F}_{u^*,v} < \lambda$, then the v-th span in the follow-up query has no conflict. The hyper-parameter $\lambda > 0$ denotes the threshold.

2.4 Extension

So far, we have introduced the whole process of STAR. Next we explore its extensibility. As observed, when the annotations are restated queries, STAR is parser-independent and can be incorporated into any context-independent semantic parser. But what if the annotations are answers to follow-up queries? Assuming we have an ideal semantic parser, a predicted restated query $\tilde{\mathbf{z}}$ can be converted into its corresponding answer $\tilde{w}$. For example, given $\tilde{\mathbf{z}}$ as "where are the players from", $\tilde{w}$ could be "Las Vegas". Therefore, revisiting Equation 3, in theory STAR is able to be extended by redesigning r as $r(w, \tilde{w})$, where w denotes the answer annotation. We conduct an extension experiment to verify it, as discussed in Section 3.3.

3 Experiments

In this section, we demonstrate the effectiveness of STAR on the FollowUp dataset[3] with restated query annotations, and its promising extensibility on the SQA dataset[4] with answer annotations.

3.1 Implementation details

We utilize PyTorch (Paszke et al., 2017) and AllenNLP (Gardner et al., 2018) for implementation, and adopt Adam (Kingma and Ba, 2015) as

[2]Their definitions along with the motivations of using them will be explained in Section 3.2.

[3]http://github.com/SivilTaram/FollowUp

[4]http://aka.ms/sqa

Model	Dev		Test		
	SymAcc (%)	BLEU (%)	SymAcc (%)	BLEU (%)	AnsAcc (%)
SEQ2SEQ† (Bahdanau et al., 2015)	0.63 ± 0.00	21.34 ± 1.14	0.50 ± 0.22	20.72 ± 1.31	–
COPYNET† (Gu et al., 2016)	17.50 ± 0.87	43.36 ± 0.54	19.30 ± 0.93	43.34 ± 0.45	–
COPY+BERT (Devlin et al., 2019)	18.63 ± 0.61	45.14 ± 0.68	22.00 ± 0.45	44.87 ± 0.52	–
CONCAT†	–	–	22.00 ± –	52.02 ± –	25.24
E2ECR† (Lee et al., 2017)	–	–	27.00 ± –	52.47 ± –	27.18
FANDA† (Liu et al., 2019)	49.00 ± 1.28	60.14 ± 0.98	47.80 ± 1.14	59.02 ± 0.54	60.19
STAR	**55.38** ± 1.21	**67.62** ± 0.65	**54.00** ± 1.09	**67.05** ± 1.05	**65.05**

Table 1: SymAcc, BLEU and AnsAcc on the FollowUp dataset. Results marked † are from Liu et al. (2019).

the optimizer. The dimensions of word embedding and hidden state are both 100. Variational dropout (Blum et al., 2015) is employed at embedding layer for better generalization ability (with probability 0.5). The learning rate is set to be 0.001 for pre-training, 0.0001 for RL training on FollowUp, and 0.0002 for SQA. In the implementation of the REINFORCE algorithm, we set M to be 20. Finally, for hyper-parameters, we set $\alpha = 0.5$, $\beta = 0.5$ and $\lambda = 0.6$. All the results are averaged over 5 runs with random initialization.

3.2 Results on FollowUp dataset

The FollowUp dataset contains 1000 natural language query triples $(\mathbf{x}, \mathbf{y}, \mathbf{z})$. Each triple belongs to a single database table, and there are 120 tables in several different domains. Following the previous work, we split them into the sets of size 640/160/200 for train/dev/test. We evaluate the methods using both answer level and query level metrics. *AnsAcc* is to check the answer accuracy of predicted queries manually. Concretely, 103 golden restated queries can be successfully parsed by COARSE2FINE (Dong and Lapata, 2018). We parse their corresponding predicted queries into SQL using COARSE2FINE and manually check the answers. Although AnsAcc is most convincing, it cannot cover the entire test set. Therefore, we apply two query level metrics: *SymAcc* detects whether all the SQL-related words are correctly involved in the predicted queries, for example column names, cell values and so on. It reflects the approximate upper bound of AnsAcc, as the correctness of SQL-related words is a prerequisite of correct execution in most cases; *BLEU*, referring to the cumulative 4-gram BLEU score, evaluates how similar the predicted queries are to the golden ones (Papineni et al., 2002). SymAcc focuses on limited keywords, so we introduce BLEU to evaluate quality of the entire predicted query.

3.2.1 Model Comparison

Our baselines fall into two categories. **Generation-based methods** conform to the architecture of sequence-to-sequence (Sutskever et al., 2014) and generate restated queries by decoding each word from scratch. SEQ2SEQ (Bahdanau et al., 2015) is the sequence-to-sequence model with attention, and COPYNET further incorporates a copy mechanism. COPY+BERT incorporates the latest pre-trained BERT model (Devlin et al., 2019) as the encoder of COPYNET. **Rewriting-based methods** obtain restated queries by rewriting precedent and follow-up queries. CONCAT directly concatenates the two queries. E2ECR (Lee et al., 2017) obtain restated queries by performing coreference resolution in follow-up queries. FANDA (Liu et al., 2019) utilizes a structure-aware model to merge the two queries. Our method STAR also belongs to this category.

Answer Level Table 1 shows AnsAcc results of competitive baselines on the test set. Compared with them, STAR achieves the highest, 65.05%, which demonstrates its superiority. Meanwhile, it verifies the feasibility of follow-up query analysis in cooperating with context-independent semantic parsing. Compared with CONCAT, our approach boosts over 39.81% on COARSE2FINE for the capability of context-dependent semantic parsing.

Query Level Table 1 also shows SymAcc and BLEU of different methods on the dev and test sets. As observed, STAR significantly outperforms all baselines, demonstrating its effectiveness. For example, STAR achieves an absolute improvement of 8.03% BLEU over the state-of-the-art baseline FANDA on testing. Moreover, the rewriting-based baselines, even the simplest CONCAT, perform better than the generation-based

Variant	SymAcc (%)	BLEU (%)
STAR	55.38	67.62
– Phase I	40.63	61.82
– Phase II	23.12	48.65
– RL	41.25	60.19
+ Basic Reward	43.13	58.48
+ Oracle Reward	45.20	63.04
+ Uniform Reward	53.40	66.93

Table 2: Variant results on FollowUp dev set.

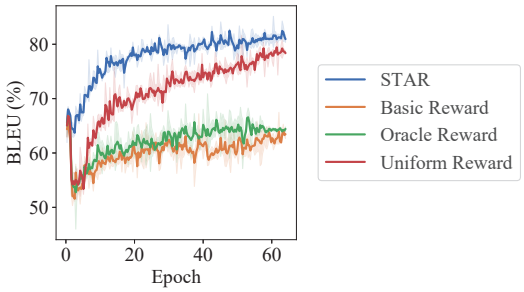

Figure 4: Learning curve on FollowUp train set.

ones. It suggests that the idea of rewriting is more reasonable for the task, where precedent and follow-up queries are of full utilization.

3.2.2 Variant Analysis

Besides baselines, we also conduct experiments with several variants of STAR to further validate the design of our model. As shown in Table 2, there are three variants with ablation: "– Phase I" takes out SplitNet and performs Phase II on word level; "– Phase II" performs random guess in the recombination process for testing; and "– RL" only contains pre-training. The SymAcc drops from about 55% to 40% by ablating Phase I, and to 23% by ablating Phase II. Their poor performances indicate both of the two phases are indispensable. "– RL" also performs worse, which again demonstrates the rationality of applying RL.

Three more variants are presented with different designs of $R(q, \mathbf{z})$ to prove the efficiency and effectiveness of Equation 5 as a reward. "+ Basic Reward" represents the most straightforward REINFORCE algorithm, which samples both $q \in \mathcal{Q}$ and $\tilde{\mathbf{z}} \in \mathcal{Z}$, then takes $r(\mathbf{z}, \tilde{\mathbf{z}})$ as $R(q, \mathbf{z})$. "+ Oracle Reward" assumes the conflicts are always correct and rewrites $R(q, \mathbf{z})$ as $\max_{\tilde{\mathbf{z}} \in \mathcal{Z}}(r(\mathbf{z}, \tilde{\mathbf{z}}))$. "+ Uniform Reward" assigns the same probability to all $\tilde{\mathbf{z}}$ and obtains $R(q, \mathbf{z})$ as $\text{mean}(\sum_{\tilde{\mathbf{z}} \in \mathcal{Z}} r(\mathbf{z}, \tilde{\mathbf{z}}))$. As shown in Table 2 and Figure 4, STAR learns better

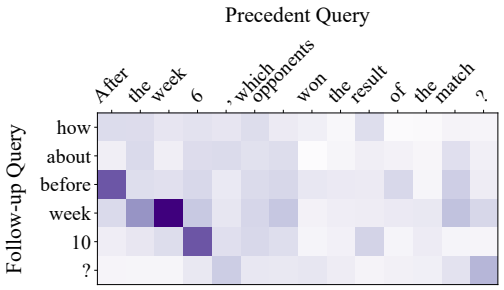

Figure 5: An example of similarity matrix in SplitNet.

and faster than the variants due to the reasonable reward design. In fact, as mentioned in Section 2.1, the vast action space of the most straightforward REINFORCE algorithm leads to poor learning. STAR shrinks the space from $|\mathcal{Q}| \cdot |\mathcal{Z}|$ down to $|\mathcal{Q}|$ by enumerating $\tilde{\mathbf{z}}$. Meanwhile, statistics show that STAR obtains a $15\times$ speedup over "+ Basic Reward" on the convergence time.

3.2.3 Case Study

Figure 5 shows a concrete example of the similarity matrix $\mathbf{A}$ on attention layer of SplitNet. The span "before week 10" is evidently more similar to "After the week 6" than to others, which meets our expectations. Moreover, the results of three real cases are shown in Table 3. The spans in blue are those have conflicts, and the histograms represent the conflict probabilities to all the spans in precedent queries. In Case 1, "glebe park", "hampden park" and "balmoor" are all cell values in the database table with similar meanings. STAR correctly finds out the conflict between "compared to glebe park" and "compared to balmoor" with the highest probability. Case 2 shows STAR can discover the interrelation of words, where "the writer Nancy miller" is learnt as a whole span to replace "Nancy miller" in the precedent query. As for Case 3, STAR successfully performs coreference resolution and interprets "those two films" as "greatest love and promised land". Benefiting from two phases, STAR is able to deal with diverse follow-up scenarios in different domains.

3.2.4 Error Analysis

Our approach works well in most cases except for few ones, where SplitNet fails. For example, given the precedent query "what's the biggest zone?" and the follow-up query "the smallest one", STAR prefers to recognize "the biggest zone" and "the smallest one" as two spans, rather

No	Case Analysis	
1	*Precedent*	: [compared to glebe park] [, does] [hampden park] [holds more attendances at capacity ?]
	Follow-up	: [how about] [compared to balmoor]
	STAR	: compared to balmoor , does hampden park holds more attendances at capacity ?
2	*Precedent*	: [Is there any book which belongs to] [Nancy miller]
	Follow-up	: [I mean] [the writer Nancy miller]
	STAR	: Is there any book which belongs to the writer Nancy miller
3	*Precedent*	: [show directors of] [greatest love and promised land]
	Follow-up	: [show air date of] [those two films]
	STAR	: show air date of greatest love and promised land

Table 3: Case analysis of STAR on FollowUp dataset. Square brackets denote different spans.

Model	Precedent	Follow-up
DynSP (Iyyer et al., 2017)	70.9	35.8
NP (Neelakantan et al., 2016)	58.9	35.9
NP + STAR	58.9	38.1
DynSP + STAR	70.9	39.5
DynSP* (Iyyer et al., 2017)	70.4	41.1

Table 4: Answer accuracy on SQA test set.

than perform split operations inside them. The SplitNet fails probably because the conflicting spans, "the biggest" ↔ "the smallest" and "zone" ↔ "one", are adjacent, which makes it difficult to identify span boundaries well.

3.3 Extension on SQA dataset

Finally, we demonstrate STAR's extensibility in working with different annotations. As mentioned in Section 2.4, by designing $r(w, \tilde{w})$, STAR can cooperate with the answer annotations. We conduct experiments on the SQA dataset, which consists of 6066 query sequences (5042/1024 for train/test). Each sequence contains multiple natural language queries and their answers, where we are only interested in the first query and the immediate follow-up one. As discussed in (Iyyer et al., 2017), every answer can be represented as a set of cells in the tables, each of which is a multi-word value, and the intentions of the follow-up queries mainly fall into three categories. *Column selection* means the follow-up answer is an entire column; *Subset selection* means the follow-up answer is a subset of the precedent answer; and *Row selection* means the follow-up answer has the same rows with the precedent answer.

We employ two context-independent parsers, DynSP (Iyyer et al., 2017) and NP (Neelakantan et al., 2016), which are trained on the SQA dataset to provide relatively reliable answers for reward computing. Unfortunately, they both perform poorly for the restated queries, as the restated queries are quite different from the original queries in SQA. To address the problem, we redesign the recombination process. Instead of generating the restated query, we recombine the predicted precedent answer $\tilde{w}_x$ and the predicted follow-up answer $\tilde{w}_y$ to produce the restated answer $\tilde{w}$. Therefore, the objective of Phase II is to assign an appropriate intention to each follow-up span via an additional classifier. The goal of Phase I turns to split out spans having obvious intentions such as "of those". The way of recombining answer is determined by the voting from intentions on all spans. If the intention column selection wins, then $\tilde{w} = \tilde{w}_y$; for subset selection, we obtain the subset $\tilde{w}$ by taking the rows of $\tilde{w}_y$ as the constraint and applying it to $\tilde{w}_x$; and for row selection, we take the rows of $\tilde{w}_x$ and the columns of $\tilde{w}_y$ as the constraints, then apply them to the whole database table to obtain the answer $\tilde{w}$ retrieved by the predicted SQL. The reward $r(w, \tilde{w})$ is computed based on Jaccard similarity between the gold answer w and $\tilde{w}$ as in (Iyyer et al., 2017), and the overall training process remains unchanged.

Table 4 shows the answer accuracy of precedent and follow-up queries on test set. DynSP* (Iyyer et al., 2017) is designed for SQA by introducing a special action *Subsequent* to handle follow-up queries based on DynSP. DynSP* is incapable of being extended to work with the annotation of the restated queries. We attempt to apply DynSP* (trained on SQA) directly on FollowUp test set, which results in an extremely low AnsAcc. On the contrary, STAR is extensible. "+STAR" means our method STAR is incorporated into the context-independent parser and empowers them with the ability to perform follow-up query analysis. As observed, integrating STAR consistently improves

performance for follow-up queries, which demonstrates the effectiveness of STAR in collaborating with different semantic parsers. The comparable results of DynSP+STAR to DynSP* further verifies the promising extensibility of STAR.

4 Related Work

Our work is closely related to two lines of work: context-dependent sentence analysis and reinforcement learning. From the perspective of context-dependent sentence analysis, our work is related to researches like reading comprehension in dialogue (Reddy et al., 2019; Choi et al., 2018), dialogue state tracking (Williams et al., 2013), conversational question answering in knowledge base (Saha et al., 2018; Guo et al., 2018), context-dependent logic forms (Long et al., 2016), and non-sentential utterance resolution in open-domain question answering (Raghu et al., 2015; Kumar and Joshi, 2017). The main difference is that we focus on the context-dependent queries in NLIDB which contain complex scenarios. As for the most related context-dependent semantic parsing, Zettlemoyer and Collins (2009) proposes a context-independent CCG parser and then conduct context-dependent substitution, Iyyer et al. (2017) presents a search-based method for sequential questions, and Suhr et al. (2018) presents a sequence-to-sequence model to solve the problem. Compared to their methods, our work achieves context-dependent semantic parsing via learnable restated queries and existing context-independent semantic parsers.

Moreover, the technique of reinforcement learning has also been successfully applied to natural language tasks in dialogue, such as hyper-parameters tuning for coreference resolution (Clark and Manning, 2016), sequential question answering (Iyyer et al., 2017) and coherent dialogue responses generation (Li et al., 2016). In this paper, we employ reinforcement learning to capture the structures of queries, which is similar to Zhang et al. (2018) for text classification.

5 Conclusion and Future Work

We present a novel method, named Split-And-Recombine (STAR), to perform follow-up query analysis. A two-phase process has been designed: one for splitting precedent and follow-up queries into spans, and the other for recombining them. Experiments on two different datasets demonstrate the effectiveness and extensibility of our method. For future work, we may extend our method to other natural language tasks.

Acknowledgments

We thank all the anonymous reviewers for their valuable comments. This work was supported by the National Natural Science Foundation of China (Grant Nos. U1736217 and 61932003).

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015*.

Avrim Blum, Nika Haghtalab, and Ariel D. Procaccia. 2015. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems, NIPS 2015, Montreal, Quebec, Canada, December 7-12, 2015*, pages 2575–2583.

Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wentau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. 2018. QuAC: Question answering in context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018, Brussels, Belgium, October 31 - November 4, 2018*, pages 2174–2184.

Kevin Clark and Christopher D. Manning. 2016. Deep reinforcement learning for mention-ranking coreference models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2256–2262.

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1–11.

Deborah A. Dahl, Madeleine Bates, Michael Brown, William M. Fisher, Kate Hunicke-Smith, David S. Pallett, Christine Pao, Alexander I. Rudnicky, and Elizabeth Shriberg. 1994. Expanding the scope of the ATIS task: The ATIS-3 corpus. In *Proceedings of the 1994 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 1994, ,Plainsboro, New Jerey, USA, March 8-11, 1994*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1:Long and Short Papers*, pages 4171–4186.

Li Dong and Mirella Lapata. 2018. Coarse-to-Fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 731–742.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software, NLP-OSS, of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018*, pages 1–6.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, Berlin, Germany, August 7-12, 2016, Volume 1: Long Papers*.

Daya Guo, Duyu Tang, Nan Duan, Ming Zhou, and Jian Yin. 2018. Dialog-to-action: Conversational question answering over a large-scale knowledge base. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems, NeurIPS 2018, Montréal, Canada, December 3-8, 2018*, pages 2946–2955.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation, Volume 9*, pages 1735–1780.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1821–1831.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, Doha, Qatar, October 25-29, 2014*, pages 1746–1751.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015*.

Vineet Kumar and Sachindra Joshi. 2017. Incomplete follow-up question resolution using retrieval based sequence to sequence learning. In *Proceedings of the 40th International ACM Conference on Research and Development in Information Retrieval, SIGIR 2017, Shinjuku, Tokyo, Japan, August 7-11, 2017*, pages 705–714.

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, EMNLP 2017, Copenhagen, Denmark, September 9-11, 2017*, pages 188–197.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 1192–1202.

Chen Liang, Jonathan Berant, Quoc V. Le, Kenneth D. Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30-August 4, 2017, Volume 1: Long Papers*, pages 23–33.

Qian Liu, Bei Chen, Jian-Guang Lou, Ge Jin, and Dongmei Zhang. 2019. FANDA: A novel approach to perform follow-up query analysis. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6770–6777.

Reginald Long, Panupong Pasupat, and Percy Liang. 2016. Simpler context-dependent logical forms via model projections. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, Berlin, Germany, August 7-12, 2016, Volume 1: Long Papers*.

Scott Miller, David Stallard, Robert J. Bobrow, and Richard M. Schwartz. 1996. A fully statistical approach to natural language interfaces. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics, ACL 1996, University of California, Santa Cruz, California, June 24-27, 1996*, pages 55–61.

Arvind Neelakantan, Quoc V. Le, and Ilya Sutskever. 2016. Neural programmer: Inducing latent programs with gradient descent. In *Proceedings of the 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of*

the 40th Annual Meeting of the Association for Computational Linguistics, ACL 2002, Philadelphia, PA, USA., July 6-12, 2002, pages 311–318.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems, NIPS 2017, Long Beach, CA, USA, December 4-9, 2017*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, Doha, Qatar, October 25-29, 2014*, pages 1532–1543.

Dinesh Raghu, Sathish Indurthi, Jitendra Ajmera, and Sachindra Joshi. 2015. A statistical approach for non-sentential utterance resolution for interactive QA system. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue, SIGDIAL 2015, Prague, Czech Republic, September 2-4, 2015*, pages 335–343.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics, Volume 7*, pages 249–266.

Amrita Saha, Vardaan Pahuja, Mitesh M. Khapra, Karthik Sankaranarayanan, and Sarath Chandar. 2018. Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence, AAAI 2018, New Orleans, Louisiana, USA, February 2-7, 2018*, pages 705–713.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Signal Processing, Volume 45*, pages 2673–2681.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017*.

Alane Suhr, Srinivasan Iyer, and Yoav Artzi. 2018. Learning to map context-dependent sentences to executable formal queries. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1:Long Papers*, pages 2238–2249.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems, NIPS 2014, Montreal, Quebec, Canada, December 8-13, 2014*, pages 3104–3112.

Richard S. Sutton and Andrew G. Barto. 1998. Reinforcement learning: An introduction. *IEEE Trans. Neural Networks, Volume 9*, pages 1054–1054.

Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12: Annual Conference on Neural Information Processing Systems, NIPS 1999, Denver Colorado, USA, November 29 - December 4, 1999*, pages 1057–1063.

Wenhui Wang and Baobao Chang. 2016. Graph-based dependency parsing with bidirectional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, Berlin, Germany, August 7-12, 2016, Volume 1: Long Papers*.

Lex Weaver and Nigel Tao. 2001. The optimal reward baseline for gradient-based reinforcement learning. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence, UAI 2001, Seattle, Washington, USA, August 2-5, 2001*, pages 538–545.

Jason D. Williams, Antoine Raux, Deepak Ramachandran, and Alan W. Black. 2013. The dialog state tracking challenge. In *Proceedings of the 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue, SIGDIAL 2013, Metz, France, August 22-24, 2013*, pages 404–413.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning, Volume 8*, pages 229–256.

Luke S. Zettlemoyer and Michael Collins. 2009. Learning context-dependent mappings from sentences to logical form. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics, ACL 2009, and the 4th International Joint Conference on Natural Language, IJCNLP 2009, Singapore, August 2-7, 2009*, pages 976–984.

Tianyang Zhang, Minlie Huang, and Li Zhao. 2018. Learning structured representation for text classification via reinforcement learning. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence, AAAI-18, New Orleans, Louisiana, USA, February 2-7, 2018*, pages 6053–6060.