Effective Search of Logical Forms for Weakly Supervised Knowledge-Based Question Answering

Tao Shen[1*], **Xiubo Geng**[2†], **Guodong Long**[1], **Jing Jiang**[1], **Chengqi Zhang**[1] and **Daxin Jiang**[2]
[1] Centre for Artificial Intelligence, School of Computer Science, University of Technology Sydney
[2] STCA NLP Group, Microsoft
tao.shen@student.uts.edu.au, {xigeng, djiang}@microsoft.com
{guodong.long, jing.jiang, chengqi.zhang}@uts.edu.au

Abstract

Many algorithms for Knowledge-Based Question Answering (KBQA) depend on semantic parsing, which translates a question to its logical form. When only weak supervision is provided, it is usually necessary to search valid logical forms for model training. However, a complex question typically involves a huge search space, which creates two main problems: 1) the solutions limited by computation time and memory usually reduce the success rate of the search, and 2) spurious logical forms in the search results degrade the quality of training data. These two problems lead to a poorly-trained semantic parsing model. In this work, we propose an effective search method for weakly supervised KBQA based on operator prediction for questions. With search space constrained by predicted operators, sufficient search paths can be explored, more valid logical forms can be derived, and operators possibly causing spurious logical forms can be avoided. As a result, a larger proportion of questions in a weakly supervised training set are equipped with logical forms, and fewer spurious logical forms are generated. Such high-quality training data directly contributes to a better semantic parsing model. Experimental results on one of the largest KBQA datasets (i.e., CSQA) verify the effectiveness of our approach and deliver a new state-of-the-art performance.

1 Introduction

Knowledge-based question answering (KBQA) interacts with a knowledge base (KB) and draws a correct answer for a factoid question. Many top-performing approaches to KBQA are based on a semantic parsing framework, that is, translating a natural language question into corresponding logical form in the light of pre-defined grammars [Artzi and Zettlemoyer, 2013; Vlachos and Clark, 2014; Suhr *et al.*, 2018; Agarwal *et al.*, 2019]. For example, "*how many people have birthplace at Provence*" has a corresponding logical form COUNT(FIND(*Provence*, *place-of-birth*)). The logical form is then executed by a KB system to retrieve an answer.

To train a semantic parser, ideal training example is in the format of ⟨*question, logical form*⟩. Nevertheless, it usually requires some expertise to compose logical forms, especially for complex questions [Berant *et al.*, 2013; Pasupat and Liang, 2015; Saha *et al.*, 2018], so it is not realistic to employ crowdsourcing to scale up the size of such training data. To circumvent this challenge, a weakly supervised training setting was proposed. The idea is to create training examples in the format ⟨*question, answer*⟩ instead of ⟨*question, logical form*⟩, since it is easier to get the answer for a factoid question than writing the corresponding logical form. However, answers cannot be directly used to train a semantic parser effectively. Hence, given a factoid question, a crucial step in weak supervision is to automatically search for valid logical forms over a knowledge base, which must lead to the given ground-truth answer after its execution. Logical forms derived from the searching process will then be considered as fully-supervised training targets for a semantic parser.

In this case, the quality of a semantic parser depends heavily on the effectiveness of upstream searching process for logical forms. However, the search space for eligible logical forms can be very large [Iyyer *et al.*, 2017]. For example, a complex question frequently involves 7 to 8 steps, and in each step an operator is chosen from up to 20 candidates. The size of search space is then about $20^7 \sim 20^8$. Although we may leverage the constraint of grammars to prune the search space, it can still be at the magnitude of $10^4 \sim 10^5$. The large search space results in two challenges as follows.

First, it may not be practical to exhaustively search the whole hypothesis space, since this takes huge cost in computation time and memory to verify each candidate by executing the logical form in a large-scale KB. Therefore, an usual practice is to search a randomly-selected, middle-sized subspace. However, such incomplete search possibly misses valid logical forms. In our empirical study, we define *search success ratio* as the number of questions for which the subspace search can find valid logical forms, divided by the total number of questions. We applied some traditional search algorithms, such as naive BFS [Guo *et al.*, 2018], on a public dataset CSQA [Saha *et al.*, 2018], and found the success ratio is very low. For example, the search success ratio for *comparative* and *quantitative* questions are barely 25% and 43%

[*] Work done during internship at STCA NLP group, Microsoft
[†] Corresponding author

respectively. In other words, for a large percentage of these questions, there are no corresponding logical forms generated as training data. As shown in our experiment part (§3.2), the insufficient training data can negatively impact the performance of a semantic parsing model.

Second, even if we overcome the practical resource limitations and search the entire space, we are likely to find spurious logical forms rather than correct ones. A spurious logical form does not match the semantic meaning of the original question, but coincidentally results in ground-truth answer after execution. For example, given a question "*Who wrote the screenplay for Inherent Vice*", the following two logical forms, i.e., 1) DIFF(FIND(SET(*Inherent Vice*), *screenwriter*), SET(*Inherent Vice*)), and 2) FIND(SET(*Inherent Vice*, *screenwriter*)), are valid and lead to the ground-truth answer, but only the latter is correct. To measure the severity of spurious logical forms, we conducted a quantitative analysis over randomly held-out examples by human evaluation on CSQA, and found that up to 54.5% of the search results are spurious. A large percentage of spurious forms in the training set can introduce high noise and thus also diminish the performance of a semantic parsing model.

Several prior works have been proposed to reduce search space or decrease spurious logical forms, which can be categorized into two ways. First, some methods use techniques to reduce search space but still suffer from spurious logical forms, such as macro grammars [Zhang *et al.*, 2017] and logical form sketch [Dong and Lapata, 2018]. Second, other works try to gradually reduce spurious logical forms while their models are iteratively trained on weakly supervised data, such as iterative search [Dasigi *et al.*, 2019] and discrete hard EM [Min *et al.*, 2019], but these methods still suffer from high failure ratio due to exponentially-growing search space.

In this work, we propose a novel approach to effectively search for logical forms over a large-scale knowledge base by introducing an operator predictor. Intuitively, we can estimate operator candidates for a given question based on its semantics, e.g., interrogative, keywords and contextual embeddings. For example, the phrase "the most" may suggest ARGMAX and "less than" may suggest LESS. With the constraint of the predicted operator set, searching for valid logical forms will result in a lower percentage of spurious logical forms and a higher ratio of the search success. In turn, high-quality training data will improve the accuracy of downstream question-to-logical-form translation model. Additionally, the predicted small set of operators can also be easily integrated into translation model's decoder to improve performance by constraining the decoding vocabulary.

Experiments on CSQA dataset [Saha *et al.*, 2018], one of the largest weakly supervised KBQA datasets over a large-scale KB with complex questions, verify the effectiveness of this approach. In particular, by searching for logical forms with our approach, the percentage of spurious logical form is reduced from 55% to 27% by human evaluation, and the search success ratio increases from 72% to 76%; for KBQA task, the overall score is significantly improved compared to baselines, and achieves a new state-of-the-art performance.

Alias	Operator
A1/A2/A3	$start \rightarrow set/num/bool$
A4	$set \rightarrow \text{FIND}(set, p)$
A5	$num \rightarrow \text{COUNT}(set)$
A6	$bool \rightarrow \text{IN}(e, set)$
A7/A8/A9	$set \rightarrow \text{UNION} / \text{INTER} / \text{DIFF}(set_1, set_2)$
A10/A11	$set \rightarrow \text{GREATER} / \text{LESS}(set, p, num)$
A12	$set \rightarrow \text{EQUAL}(set, p, num)$
A13/A14	$set \rightarrow \text{ARGMAX} / \text{ARGMIN}(set, p)$
A15	$set \rightarrow \text{SET}(e)$
A16/A17/A18	$e/p/num \rightarrow constant$ *

Table 1: Grammars to compose logical form. *Instantiation for entity e, predicate p or number num from an input question.

2 Our Approach

This section starts with an introduction to grammars and logical form. Then, an outline of proposed approach and the details of its components are elaborated.

2.1 Grammar and Logical Form

We leverage similar formats of grammar and logical form as in [Guo *et al.*, 2018]. Here we give a brief introduction and refer readers to [Guo *et al.*, 2018] for more details.

Grammar. The grammar definitions are shown in Table 1, where each operator is composed of three parts, i.e., a semantic category, a function symbol and a list of arguments. An argument can be a semantic category or a constant instantiated from a question.

Logical Form. A KB-executable logical form is usually formatted as a tree structure, where the root is the $start$ operator and each child node is a legitimate operator constrained by a semantic category in its parent's argument list. To take advantage of sophisticated sequence-to-sequence models [Bahdanau *et al.*, 2015; Vaswani *et al.*, 2017] for question-to-logical-form translation, we re-format tree structure into a sequence by applying depth-first traversal over the tree. Reversely, once a sequence-formatted logical form is generated during decoding phase, it can be easily recovered into tree structure under grammars' guidance.

2.2 Overview of Our Approach

As shown in Figure 1, our approach mainly consists of 6 steps: searching, cleaning, training operator predictor, operator prediction, re-searching, and training semantic parser.

Searching and Cleaning. Step 1 and 2 create training examples for proposed operator predictor based on a sampled small subset (e.g., $1/10$ of the total in this work) of the entire training data $\mathcal{D}$, which results in a new training set $\mathcal{D}'$.

In Step 1, for each question Q^i in the sampled small dataset, we naively search and get valid[1] logical forms $LF^i = \{lf^{i1}, lf^{i2}, \dots\}$. As stated in §1, this step could generate spurious logical forms and lead to bad operator predictor if we directly use these data for the model training. Hence, we further clean the searched results in Step 2.

[1]Multiple valid logical forms would be found for one question, but only a part of them are correct, otherwise spurious.

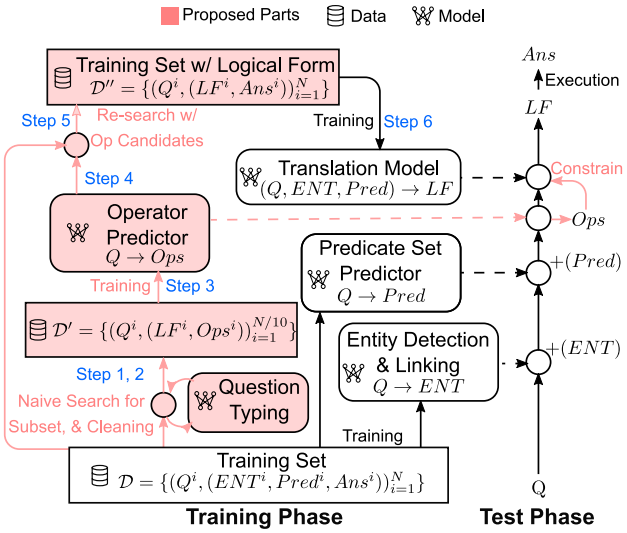

Figure 1: An overview of the proposed approach.

In Step 2, we clean searched logical forms according to taxonomy of questions, which is inspired by an observation that questions belong to the same type require similar operators. For example, these three quantitative questions, "*How many cities are sister town of ...*", "*How many rivers flow through ...*", and "*How many countries have ...*" require the COUNT operator, rather than ARGMAX. Specifically, we first create a legitimate operator set for each question type. The criterion is that, for a question type, an operator is legitimate if removing the operator from the candidates leads to a notable (e.g., 1% in our setup) search success ratio drop for the questions in that type. Then for each LF^i, we remove illegal logical forms that contain any operators not belonging to the legitimate set of the corresponding question type. And, Ops^i is defined as a set of unique operators appearing in the cleaned LF^i. To obtain a type of each question, we use *unsupervised clustering* to train a question typing model (detailed in §2.3).

Model Training for Operator Predictor. In Step 3, an operator predictor model $\mathcal{M}$, which maps a question Q^i into its most likely operators Ops^i to compose a correct logical form, is trained based on the cleaned training data $\mathcal{D}'$. More details about operator predictor are presented in §2.3.

Training Data Generation for KBQA. In Step 4 and 5, we apply $\mathcal{M}$ to each question in all training data $\mathcal{D}$ for predicting its most likely operators Ops^i, and then re-search for valid logical forms LF^i based on the reduced operator candidates $Ops^i \subseteq \mathcal{A}$. $\mathcal{A}$ is a set of all operators defined in Table 1.

Model Training for KBQA. Lastly, in Step 6, we train a semantic parser based on the re-searched results in Step 5.

Notably, this approach involves two rounds of searches, but it is still faster than previous works (e.g., naive BFS by Guo *et al.* [2018]). The reason is that, in Step 1 only a small subset (e.g., 10%) needs to be fully searched as previous works do; and with reduced search space, the re-search in Step 5 is much faster than previous works. For example, on CSQA benchmark, our algorithm is ∼20× faster than its baseline.

There are two main benefits to incorporating an operator predictor into a standard "*searching* and *training*" scheme. First, it helps to provide high-quality data for downstream model training by improving search success ratio and reducing the number of spurious logical forms. Second, it makes training and inference more efficient by providing the constraint from legal operators.

2.3 Model Details

We detail the implementation of our models in this section, including question typing in Step 2 and operator predictor in Step 3. Then, we summarize a neural symbolic model [Guo *et al.*, 2018] as semantic parsing baseline to complete our framework, which consists of three sub-tasks: entity detection & linking, predicate prediction, and sequence-to-sequence translation (refer to [Guo *et al.*, 2018] for details).

In formal terms, a question Q is first tokenized as a list of words, i.e., $Q = [w_1, \ldots, w_n]$, and then, a word embedding approach [Mikolov *et al.*, 2013] is invoked to transform the discrete words into low-dimensional vector representations, i.e., $\boldsymbol{X} = [\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}] \in \mathbb{R}^{d_e \times n}$, where d_e denotes embedding size and n stands for question sequence length.

Unsupervised Question Typing. Off-the-shelf question type information is not usually provided in a dataset, so an unsupervised question typing approach is proposed to generalize our pipeline. This can be regarded as a text clustering task that targets assigning same cluster label to the questions with similar purposes. LDA topic modeling or k-means clustering, which operates on texts (or their embeddings), are typical ways to fulfill text clustering. However, direct operating on texts in an unsupervised manner empirically results in an inferior performance because the pivotal words (e.g., *how many*, *large than*) of this problem are usually identified as stop words by TF-IDF. To handle this, we make the clustering algorithm operate on the logical forms. We first derive a binary vector $\boldsymbol{b}^i \in \{0, 1\}^{|\mathcal{A}|}$ from LF^i in naive search (Step 1). Each element in $\boldsymbol{b}^i$ denotes whether the corresponding operator appears in LF^i. Then we use the binary vectors of the small dataset in Step 1 as input features to train a k-means model, and assign each question with a cluster label.

Operator Predictor. We define this sub-task as a multi-label classification problem, whose input is a natural language question and output is a set of operators possibly composing correct logical form for the question. In particular, a bi-directional LSTM (Bi-LSTM) [Hochreiter and Schmidhuber, 1997] performs over input word embeddings as an encoder to capture contextual information, which is denoted as

$$\overrightarrow{\boldsymbol{h}}_{\boldsymbol{i}}^{(\boldsymbol{s})} = \overrightarrow{\text{LSTM}}(\boldsymbol{x_i}, \overrightarrow{\boldsymbol{h}}_{\boldsymbol{i-1}}^{(\boldsymbol{s})}; \theta^{(s1)}),\ \forall i = 1, \ldots, n, \quad (1)$$

$$\overleftarrow{\boldsymbol{h}}_{\boldsymbol{i}}^{(\boldsymbol{s})} = \overleftarrow{\text{LSTM}}(\boldsymbol{x_i}, \overleftarrow{\boldsymbol{h}}_{\boldsymbol{i+1}}^{(\boldsymbol{s})}; \theta^{(s2)}),\ \forall i = n, \ldots, 1, \quad (2)$$

$$\boldsymbol{u}^{(\boldsymbol{s})} = [\overrightarrow{\boldsymbol{h}}_{\boldsymbol{n}}^{(\boldsymbol{s})}; \overleftarrow{\boldsymbol{h}}_{\boldsymbol{1}}^{(\boldsymbol{s})}] \in \mathbb{R}^{2d_h}, \quad (3)$$

where $\theta^{(*)}$ are learnable parameters for Bi-LSTM, $[;]$ denotes a concatenation operation, and $\boldsymbol{u}^{(\boldsymbol{s})}$ is the resulting vector representation for the whole question. Then, the probability of generating each operator $\boldsymbol{p}^{(\boldsymbol{s})}$ is defined as

$$\boldsymbol{p}^{(\boldsymbol{s})} = \text{sigmoid}(\text{MLP}(\boldsymbol{u}^{(\boldsymbol{s})}; \theta^{(s3)})) \in \mathbb{R}^{|\mathcal{A}|}, \quad (4)$$

where $\mathrm{MLP}(\cdot)$ is a multi-layer perceptron.

Entity Detection & Linking and Predicate Prediction. Entity detection aims to locate named entity mention in the input question, which is usually formulated as a sequence labeling problem. To solve this, we use a Bi-LSTM to predict entity mention tag for each input word. Then, given the detected mentions, we use entity linking system from [Guo *et al.*, 2018] for linking them back to entities in a KB. In addition, identifying predicates in a question is also essential to compose an executable logical form. To this end, we simply formulate this predicate prediction sub-task as a multi-class classification problem. In brief, a Bi-LSTM is used to embed the sentence as a vector representation, which is followed by an MLP classifier to predict the correct predicate.

Question-to-Logical-Form Translation. Given predicted entity and predicate candidates from the upstream, a semantic parsing model aims to translate an input natural language question into KB-executable logical form. Since the gold logical forms have been formatted as sequences, we employ a sequence-to-sequence encoder-decoder structure with attention mechanism [Bahdanau *et al.*, 2015] to fulfill this task.

3 Experiment

This section begins with experimental setups. Then, the evaluations includes KBQA, logical form searching, and two subtasks. Lastly, case study and error analysis are presented for qualitative understanding of this work.

3.1 Experimental Settings

Dataset. We used one of the largest weakly-supervised KBQA datasets over a large-scale knowledge base, Complex Sequential Question Answering (CSQA) [Saha *et al.*, 2018]. There are 1.6M turns in 200K dialogues without logical form. Its KB is built on WIKIDATA in the form of (*subject*, *predicate*, *object*), including 21.2M triplets over 12.8M entities.

Evaluation Metrics. Following Saha *et al.* [2018], we used *Precision/Recall* for questions whose answer is entity(s), and *Accuracy* for questions whose answer is boolean/numeric.

Model Setup. For each neural model, the word embedding weight matrix was independent of each other and the embedding size d_e was 300D; the hidden state size d_h was also set to 300D and activation function was ReLU for the middle layer of each MLP. For the optimization, we used Adam optimizer with learning rate of 10^{-3}, the batch size was set to 64 for 6 epochs, and early stop strategy was applied when there was no longer a significant improvement over the development set during the training. Moreover, for the question typing and operator predictor we first used naive BFS method to search only 10% training data from CSQA and applied the data preprocessing steps outlined in §2.2. And the number of clusters in question typing was 10.

Baselines. *HRED+KVmem* [Saha *et al.*, 2018] and *D2A* [Guo *et al.*, 2018] are two typical paradigms in regard to information retrieval and neural symbolic machine respectively. In particular, HRED+KVmem involves a seq-to-seq based HRED model [Serban *et al.*, 2016] and a key-value memory network [Miller *et al.*, 2016] to retrieve answer from KB. In contrast, D2A[2] defines a set of semantic parsing grammars and translates natural language questions into corresponding logical forms to query KB via a memory-augmented neural symbolic model. Further, Guo *et al.* [2019] extended D2A to *D2A-MAML* by coupling retrieval and meta-learning for semantic parsing, and Shen *et al.* [2019] proposed a multi-task learning framework, *MaSP*, for neural semantic parsing, which are also considered as competitors. In addition, we also adapted and implemented D2A with other three methods that also aim to mitigate large search space or spurious logical form problems: 1) *rule-based filtering* method filters spurious logical forms according to handcraft lexicon rules adapted from Dasigi *et al.* [2019]; 2) *iterative search* [Dasigi *et al.*, 2019] assigns a loss weight to each logical form based on current model for subsequent training; 3) *hard EM* [Min *et al.*, 2019] aims at training on the most likely logical form from its candidates associated to a question at each update.

Methods	Overall Score		
	Recall	Precision	F1
HRED+KVmem [Saha *et al.*, 2018]	18.40%	6.30%	9.39%
D2A [Guo *et al.*, 2018]	66.83%	66.57%	66.70%
D2A+MAML [Guo *et al.*, 2019]	65.23%	63.02%	64.11%
MaSP [Shen *et al.*, 2019]	78.07%	80.48%	79.26%
D2A+Rule-based filtering†	64.54%	63.88%	64.21%
D2A+Iterative† [Dasigi *et al.*, 2019]	67.86%	69.40%	68.62%
D2A+Hard EM† [Min *et al.*, 2019]	69.03%	70.90%	69.95%
D2A+ours	71.14%	71.33%	71.23%
MaSP+ours	80.10%	82.49%	81.28%

Table 2: Comparisons on CSQA dataset. Note, 1) "+ours" means that D2A or MaSP model is integrated with the proposed pipeline as shown in Figure 1; and 2) although the oracle type label is provided in CSQA, question type for our framework is still obtained in an unsupervised manner. †adapted and implemented by us.

3.2 Question Answering Performance

As listed in Table 2, our proposed effective search approach coupled with D2A or MaSP model improves previous baselines by a significant margin, setting a new state-of-the-art performance on CSQA dataset. Specifically, compared with D2A and MaSP baseline, the proposed framework is able to improve by 4.53% and 2.02% F1 score respectively. Our proposed method also outperforms the other three with similar purpose (i.e., rule, iterative search and hard EM), which further verifies the effectiveness of the proposed method.

Additionally, in Table 3, we compared D2A baseline to our frameworks equipped with different ways to obtain the question type for logical form cleaning. In particular, although obtaining question types via oracle labels slightly outperforms that via unsupervised clustering, the former depends on labor-intensive crowdsourcing – with the cost of generalization ability. And, it is also observed that the improvements are more notable with more complex question types. For example, 8.89%/12.07% improvement of recall/precision over

[2]The re-implemented D2A in this work outperforms the one originally proposed by Guo *et al.* [2018], and one possible reason is that our re-implemented grammars reach a better performance balance between simple and non-simple questions. For a fair comparison, we report the re-produced results for D2A in this paper.

Methods	D2A		**+ours(oracle)**		**+ours(cluster)**	
Q Type in CSQA	R	P	R	P	R	P
Overall	66.83	66.57	**71.63**	**72.42**	71.14	71.33
Simple (Direct)	79.50	77.37	82.80	**83.20**	**83.23**	81.71
Simple (Co-ref)	58.47	56.94	**64.67**	**64.58**	63.64	63.21
Simple (Ellipsis)	84.67	77.90	84.88	**83.02**	**86.05**	82.64
Logical (All)	65.82	68.86	73.88	**72.00**	**77.12**	70.78
Quantitative (All)	52.74	60.63	**60.30**	**68.06**	41.42	57.87
Comparative (All)	44.14	54.68	50.42	60.62	**53.03**	**66.75**
Clarification	37.24	33.97	**38.74**	34.80	35.44	**35.70**
Q Type in CSQA	Accu		Accu		Accu	
Verification (Boolean)	37.07		45.80		**46.97**	
Quantitative (Count)	38.42		**41.35**		36.90	
Comparative (Count)	16.62		20.93		**21.17**	

Table 3: Detailed comparisons on CSQA. Note, "oracle" and "cluster" in parenthesis denote the way to obtain question type for logical form cleaning (Step 2 in §2.2) – using oracle question type labels from the dataset and unsupervised clustering respectively.

Methods	**Overall Score**		
	Recall	Precision	F1
D2A+ours (cluster)	71.14%	71.33%	71.23%
· w/o cleaning in Step 2	68.46%	69.27%	68.86%
· w/o constraints in decoding	68.78%	69.45%	69.11%
· w/o proposed pipeline†	66.83%	66.57%	66.70%

Table 4: Ablation study on CSQA. †degraded to D2A model.

Comparative Reasoning is much greater than 3.73%/4.34% improvement over *Simple*. A possible reason is that the logical forms for complex questions usually require more operators to compose, and thus they are more vulnerable to large search space problem. We attribute this to more operators required to answer more complex questions, which exacerbates the problems associated with large search space. Besides, our framework with cluster achieves an inferior performance over *Quantitative* and a reason is that the errors from clustering algorithm are propagated to logical form cleaning, where the correct logical forms were mistakenly filtered out.

Lastly, we conducted an ablation study to verify the effectiveness of each part in proposed pipeline. As shown in Table 4, removing logical form cleaning for operator predictor training, and discarding operator constraints during decoding lead to 2.37% and 2.12% F1 score decreases respectively.

3.3 Searching Effectiveness and Efficiency

In this section, we analyzed our proposed algorithm in terms of alleviating the two problems raised by large search space. Lastly, we also compared search efficiency with the baseline.

Increasing Search Success Ratio. The search success ratio is defined as the number of questions, each with at least one valid logical form found by a search method, over the total number of questions. We compared our method with traditional BFS one [Guo *et al.*, 2018], and reported the results in Figure 2. From the figure we found our method increases the search success ratio significantly, especially almost 2× increase for *logical reasoning* questions. And we also found, the improvement of KBQA is roughly proportional to the increase of search success ratio w.r.t. question types.

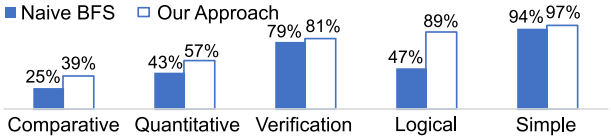

Figure 2: Search success ratio comparison w.r.t. question type.

Method	#Correct	#Spurious	%Spurious
Naive BFS	114	136	54.5%
Ours	**115**	**40**	**26.7%**

Table 5: Statistics of spurious logical forms. Note that a question may be assigned with multiple valid logical forms.

Reducing Spurious Logical Forms. To determine whether our approach reduces the number of spurious logical forms, we randomly sampled 40 questions, each with at least one valid logical form found through both naive BFS search method and our approach. Human evaluators manually inspected the results and made a judgment as to whether the logical form is correct or spurious. The results listed in Table 5 demonstrate that our approach considerably reduces the incidence of spurious logical forms from 54.5% to 26.7% compared to the baseline. This is a substantial reduction and provides a clear evidence that this approach can improve the quality of training data and thus benefit any downstream semantic parsing model.

Search Efficiency. One notable merit of the proposed searching method is that, with the constrained operator candidates, the searching procedure is significantly sped up. Statistically, our searching method takes only 0.06s per dialog turn for valid logical form(s) on average, which is 20× faster than our baseline, naive BFS, that needs 1.19s.

3.4 Sub-Task Evaluation

We evaluated question typing and operator predictor here.

Unsupervised Question Typing. we evaluated this task by checking the consistency between predicted cluster label and oracle question type from dataset. In brief, we calculated averaged proportion of the questions belonging to most frequent oracle question type in each cluster. The resulting 85.10% demonstrates the effectiveness of our proposed clustering for question typing. In contrast, direct applying k-means to raw text only results in ∼20% consistency.

Operator Predictor. To assess the quality of the operator predictor, we took ⟨*question, valid operators*⟩ pairs found by naive BFS as evaluation set, and evaluated the performance according to a metric, *Question Coverage*. *Question coverage* is defined as the ratio of questions, with predicted operators able to compose at least one valid logical form. In result, our operator predictor can achieve 98.67% of question coverage, which means when re-searching logical forms in Step 5, our approach will locate correct searching subspace for at least 98.67% questions. In addition, we calculated mean proportion of the size of predicted operator candidates over the number of all operators defined in Table 1, and found that merely 30.83% operators are retained for logical form searching and thus lead to a reduced search space and improved speed.

Question	Logical Form from Naive Approach	Logical Form from Ours	Ops Prediction
Where is Zinc finger protein 775 found?	DIFF(FIND(SET(*Zinc...775*), *found-in-taxon*), SET(*Zinc...775*))	FIND(SET(*Zinc...775*), *found-in-taxon*)*	[*start* → *set*, FIND, SET]
Is Sumy Oblast adjacent to Poltava Oblast?	IN(*Poltava Oblast*, UNION(FIND(SET(*Sumy Oblast*), *shares-border*), SET(*Italy*)))	IN(*Sumy Oblast*, FIND(SET(*Poltava Oblast*), *shares-border*))*	[*start* → *bool*, IN, FIND, SET]
Which administrative territories holds diplomatic relationship with max number of administrative territories?	DIFF(ARGMAX(COUNT(FIND(FIND(SET(*administrative territorial*), *is-a*)), *diplomatic-relation*)), SET(*Quebec*))	ARGMAX(COUNT(FIND(FIND(SET(*administrative territorial*), *is-a*)), *diplomatic-relation*))*	[*start* → *set*, FIND, COUNT, UNION, DIFF, ARGMAX, SET]
Which administrative territories are Yale University present in and are the origins of Anna Karenina?	INTER(FIND(SET(*Yale University*), *country*), FIND(SET(*Anna Karenina*), *country-of-origin*))*	FIND(SET(*Yale University*), *country*)	[*start* → *set*, FIND, COUNT, UNION, INTER, DIFF, SET]

Table 6: Case study of valid logical forms. And a logical form ending with * means it is correct by human judgement, otherwise spurious.

3.5 Case Study

In this section, we leveraged some cases to demonstrate the effectiveness of our proposed algorithm in searching for logical forms on weakly supervised KBQA. As shown in Table 6, for each question, we listed the logical forms searched by naive BFS approach and our proposed one respectively, as well as the predicted operators from *operator predictor*.

According to first three cases, due to the constraints posted by *operator predictor* (last column), our approach could avoid some spurious results. Meanwhile, as shown in fourth case, although predicted operator candidates substantially reduce the search space, it is still possible to include spurious logical forms in searching results.

3.6 Error Analysis

To conduct an error analysis and provide an insight into the causes of the prediction errors, we randomly sampled 50 wrongly-predicted examples for KBQA, and found the errors could be coarsely categorized as follows. **1) Entity Ambiguity** This is the most serious problem leading to wrong predictions during question-to-logical-form translation since many entities with identical text however express totally different meanings. For example, an entity *The Avengers* could be a movie, a soundtrack album or a punk rock band; even for a movie whose title is *The Avengers*, it also could be *2012 superhero film produced by Marvel* or *1998 film by Jeremiah S. Chechik*. **2) Error Propagation** Because a pipeline approach is employed to solve KBQA problem, it is inevitable that the prediction errors occurring at early stage will be propagated into downstream models. An apparent case is that the errors in unsupervised question typing directly lead to a slight performance decrease for KBQA. **3) Translation Error** Due to translation model's limitation on representative expression, a wrong operator or entity could be chosen to compose a logical form during decoding, which results in an incorrect answer.

4 Related Work

This work is in line with semantic parsing based approach for KBQA task. Given a natural question, based on a set of well-defined grammars for specific task, typical semantic parsing approaches learn a model to transform the question to a KB-executable logical form for answer retrieval [Wong and Mooney, 2007; Zettlemoyer and Collins, 2009; Kwiatkowski *et al.*, 2011; Andreas *et al.*, 2013; Artzi and Zettlemoyer, 2013; Zhao and Huang, 2014; Long *et al.*, 2016; Jia and Liang, 2016; Ling *et al.*, 2016; Xiao *et al.*, 2016].

Due to limited crowdsourcing, only final answers instead of full logical forms are provided to learn a semantic parsing model, i.e., in a weakly supervised learning scheme [Berant *et al.*, 2013; Iyyer *et al.*, 2017; Saha *et al.*, 2018]. Hence, "*searching* and *training*" is a conventional stepwise approach to handle such weakly supervised setting by searching logical form for semantic parser learning [Bao *et al.*, 2014; Yih *et al.*, 2015; Zhang *et al.*, 2017; Guu *et al.*, 2017; Liang *et al.*, 2018; Guo *et al.*, 2018; Dasigi *et al.*, 2019].

However, *searching* over structured KBs inevitably leads to spurious logical form problem, which introduces wrongly labeled data and thus poses negative effect on KBQA [Pasupat and Liang, 2016; Guo *et al.*, 2018]. To alleviate spurious logical forms' effect, for example, Liang *et al.* [2018] separately estimated expectations over the trajectories inside and outside high-rewarded memory buffer, rather than maximum likelihood training. Guu *et al.* [2017] reduced the impact of spurious logical forms by using randomized beam search and more balanced optimization. And, Dasigi *et al.* [2019] alternated between searching for consistent logical forms and maximizing the marginal likelihood of the retrieved ones while iterative training, which increases logical forms' complexity for subsequent ones, thus dealing with the problem of spuriousness. Min *et al.* [2019] proposed a simple but effective method that computes gradients relative to the most likely solution at each update. Some other works are proposed to handle large search space, e.g., macro grammars [Zhang *et al.*, 2017] and logical form sketch [Dong and Lapata, 2018].

5 Conclusion

We proposed a novel approach for effective search of logical forms by operator prediction for weakly supervised KBQA task. It provides sufficient and superior data for downstream question-to-logical-form translation model training, and makes training and inference more effective under the constraints of the possible operators. The proposed approach is simple and effective, which makes it of great practical use. Experimental results verify the effectiveness of our approach in terms of reducing spurious logical forms, increasing search success ratio, improving search efficiency, and boosting the final accuracy for question answering.

Acknowledgments

This research was also funded by the Australian Government through the Australian Research Council (ARC) under grants LP180100654 partnership with KS computer.

References

[Agarwal *et al.*, 2019] Rishabh Agarwal, Chen Liang, Dale Schuurmans, and Mohammad Norouzi. Learning to generalize from sparse and underspecified rewards. In *ICML*, 2019.

[Andreas *et al.*, 2013] Jacob Andreas, Andreas Vlachos, and Stephen Clark. Semantic parsing as machine translation. In *ACL*, volume 2, pages 47–52, 2013.

[Artzi and Zettlemoyer, 2013] Yoav Artzi and Luke Zettlemoyer. Weakly supervised learning of semantic parsers for mapping instructions to actions. *TACL*, 1:49–62, 2013.

[Bahdanau *et al.*, 2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

[Bao *et al.*, 2014] Junwei Bao, Nan Duan, Ming Zhou, and Tiejun Zhao. Knowledge-based question answering as machine translation. In *ACL*, volume 1, pages 967–976, 2014.

[Berant *et al.*, 2013] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *ACL*, pages 1533–1544, 2013.

[Dasigi *et al.*, 2019] Pradeep Dasigi, Matt Gardner, Shikhar Murty, Luke Zettlemoyer, and Eduard Hovy. Iterative search for weakly supervised semantic parsing. In *NAACL*, pages 2669–2680, 2019.

[Dong and Lapata, 2018] Li Dong and Mirella Lapata. Coarse-to-fine decoding for neural semantic parsing. *arXiv preprint arXiv:1805.04793*, 2018.

[Guo *et al.*, 2018] Daya Guo, Duyu Tang, Nan Duan, Ming Zhou, and Jian Yin. Dialog-to-action: Conversational question answering over a large-scale knowledge base. In *NurIPS*, 2018.

[Guo *et al.*, 2019] Daya Guo, Duyu Tang, Nan Duan, Ming Zhou, and Jian Yin. Coupling retrieval and meta-learning for context-dependent semantic parsing. In *ACL*, pages 855–866, 2019.

[Guu *et al.*, 2017] Kelvin Guu, Panupong Pasupat, Evan Zheran Liu, and Percy Liang. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. *arXiv preprint arXiv:1704.07926*, 2017.

[Hochreiter and Schmidhuber, 1997] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[Iyyer *et al.*, 2017] Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. Search-based neural structured learning for sequential question answering. In *ACL*, volume 1, pages 1821–1831, 2017.

[Jia and Liang, 2016] R. Jia and P. Liang. Data recombination for neural semantic parsing. In *ACL*, 2016.

[Kwiatkowski *et al.*, 2011] Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. Lexical generalization in ccg grammar induction for semantic parsing. In *EMNLP*, 2011.

[Liang *et al.*, 2018] Chen Liang, Mohammad Norouzi, Jonathan Berant, Quoc Le, and Ni Lao. Memory augmented policy optimization for program synthesis with generalization. *arXiv preprint arXiv:1807.02322*, 2018.

[Ling *et al.*, 2016] Wang Ling, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, Andrew Senior, Fumin Wang, and Phil Blunsom. Latent predictor networks for code generation. *arXiv preprint arXiv:1603.06744*, 2016.

[Long *et al.*, 2016] Reginald Long, Panupong Pasupat, and Percy Liang. Simpler context-dependent logical forms via model projections. *arXiv preprint arXiv:1606.05378*, 2016.

[Mikolov *et al.*, 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, 2013.

[Miller *et al.*, 2016] Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*, 2016.

[Min *et al.*, 2019] Sewon Min, Danqi Chen, Hannaneh Hajishirzi, and Luke Zettlemoyer. A discrete hard EM approach for weakly supervised question answering. In *EMNLP*, 2019.

[Pasupat and Liang, 2015] Panupong Pasupat and Percy Liang. Compositional semantic parsing on semi-structured tables. *arXiv preprint arXiv:1508.00305*, 2015.

[Pasupat and Liang, 2016] Panupong Pasupat and Percy Liang. Inferring logical forms from denotations. *arXiv preprint arXiv:1606.06900*, 2016.

[Saha *et al.*, 2018] Amrita Saha, Vardaan Pahuja, Mitesh M Khapra, Karthik Sankaranarayanan, and Sarath Chandar. Complex sequential question answering: Towards learning to converse over linked question answer pairs with a knowledge graph. In *AAAI*, 2018.

[Serban *et al.*, 2016] Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, 2016.

[Shen *et al.*, 2019] Tao Shen, Xiubo Geng, Tao Qin, Daya Guo, Duyu Tang, Nan Duan, Guodong Long, and Daxin Jiang. Multi-task learning for conversational question answering over a large-scale knowledge base. In *EMNLP*, pages 2442–2451, 2019.

[Suhr *et al.*, 2018] Alane Suhr, Srinivasan Iyer, and Yoav Artzi. Learning to map context-dependent sentences to executable formal queries. *arXiv preprint arXiv:1804.06868*, 2018.

[Vaswani *et al.*, 2017] Ashish Vaswani, Shazeer, Noam, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

[Vlachos and Clark, 2014] Andreas Vlachos and Stephen Clark. A new corpus and imitation learning framework for context-dependent semantic parsing. *TACL*, 2:547–560, 2014.

[Wong and Mooney, 2007] Yuk Wah Wong and Raymond Mooney. Learning synchronous grammars for semantic parsing with lambda calculus. In *ACL*, pages 960–967, 2007.

[Xiao *et al.*, 2016] Chunyang Xiao, Marc Dymetman, and Claire Gardent. Sequence-based structured prediction for semantic parsing. In *ACL*, volume 1, pages 1341–1350, 2016.

[Yih *et al.*, 2015] Scott Wen-tau Yih, Ming-Wei Chang, Xiaodong He, and Jianfeng Gao. semantic parsing via staged query graph generation: question answering with knowledge base. In *ACL*, 2015.

[Zettlemoyer and Collins, 2009] Luke S Zettlemoyer and Michael Collins. Learning context-dependent mappings from sentences to logical form. In *ACL*, pages 976–984, 2009.

[Zhang *et al.*, 2017] Yuchen Zhang, Panupong Pasupat, and Percy Liang. Macro grammars and holistic triggering for efficient semantic parsing. *arXiv preprint arXiv:1707.07806*, 2017.

[Zhao and Huang, 2014] Kai Zhao and Liang Huang. Type-driven incremental semantic parsing with polymorphism. *arXiv preprint arXiv:1411.5379*, 2014.