Coupling Retrieval and Meta-Learning for Context-Dependent Semantic Parsing

Daya Guo[1]*, **Duyu Tang**[2], **Nan Duan**[2], **Ming Zhou**[2], **and Jian Yin**[1]
[1] The School of Data and Computer Science, Sun Yat-sen University.
Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou, P.R.China
[2] Microsoft Research Asia, Beijing, China
{guody5@mail2,issjyin@mail}.sysu.edu.cn
{dutang,nanduan,mingzhou}@microsoft.com

Abstract

In this paper, we present an approach to incorporate retrieved datapoints as supporting evidence for context-dependent semantic parsing, such as generating source code conditioned on the class environment. Our approach naturally combines a retrieval model and a meta-learner, where the former learns to find similar datapoints from the training data, and the latter considers retrieved datapoints as a pseudo task for fast adaptation. Specifically, our retriever is a context-aware encoder-decoder model with a latent variable which takes context environment into consideration, and our meta-learner learns to utilize retrieved datapoints in a model-agnostic meta-learning paradigm for fast adaptation. We conduct experiments on CONCODE and CSQA datasets, where the context refers to class environment in JAVA codes and conversational history, respectively. We use sequence-to-action model as the base semantic parser, which performs the state-of-the-art accuracy on both datasets. Results show that both the context-aware retriever and the meta-learning strategy improve accuracy, and our approach performs better than retrieve-and-edit baselines.

1 Introduction

Context-dependent semantic parsing aims to map a natural language utterance to a structural logical form (e.g. source code) conditioned on a given context (e.g. class environment) (Ling et al., 2016; Long et al., 2016; Iyyer et al., 2017; Iyer et al., 2018; Suhr et al., 2018; Suhr and Artzi, 2018). Standard approaches typically learn a one-size-fits-all model on the entire training dataset, which is fed with each example individually in the training phase and makes predictions for each test example in the inference phase. However, taking code generation as an example, programmers usually do not write codes from scratch in the real world. When they write a piece of code in a particular environment, they typically leverage past experience on writing or reading codes in the similar situation as a guidance. Meanwhile, datapoints for a task may vary widely (Huang et al., 2018a), thus it is desirable to learn a "personalized" model for the target datapoint. In this work, we study how to automatically retrieve similar datapoints in a context-dependent scenario and use them as the supporting evidence to facilitate semantic parsing.

There are recent attempts at exploiting retrieved examples to improve the generation of logical form and text. Retrieve-and-edit approaches (Hashimoto et al., 2018; Huang et al., 2018b; Wu et al., 2018; Gu et al., 2017) typically first use a context-independent retriever to find the most relevant datapoint, and then use it as an additional input of the editing model. However, a context-aware retriever is very important for the task of context-dependent semantic parsing. For examples, as shown in Figure 1, class environment can help the retriever decide whether the desired code of *"Increment this vector"* is generated by directly calling $add()$ or iterating the $vecElements$ array to increment each element. Furthermore, retrieve-and-edit approaches typically consider only one similar example to edit. In semantic parsing, the pattern of a structural output may come from different retrieved examples. There also exist works to utilize multiple examples to guide the semantic parser (Hayati et al., 2018; Huang et al., 2018a), however, these approaches either use a heuristic way to exploit the retrieved logical form such as increasing the probability of actions (Hayati et al., 2018) or use a relevance function designed and learned based on expertise about the target logical form (Huang et al., 2018a). When we consider the context environment, it's nontrivial to design

* Work done while this author was an intern at Microsoft Research.

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 855–866
Florence, Italy, July 28 - August 2, 2019.

a context-aware relevance function since the form of the context environment varies widely.

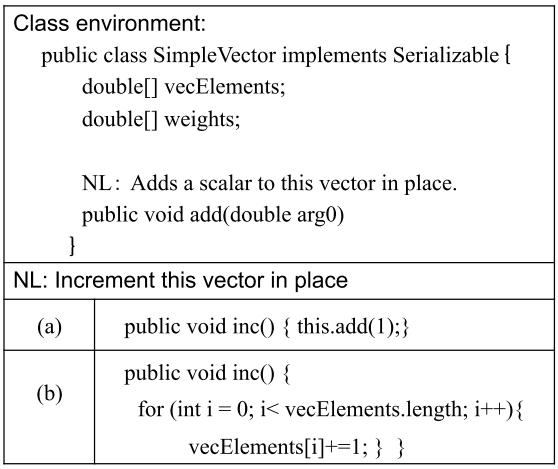

Figure 1: Code generation based on the class environment and a natural language documentation (NL). (a) shows a example of code generation by applying the class function $add()$, while (b) iterates the $vecElements$ array to increment each element.

In this work, we propose to retrieve similar examples by taking into account the context environment, and use meta-learning to utilize retrieved examples to guide the generation of a logical form. Our retriever is a context-aware encoder-decoder model that takes context environment into consideration. Specially, the model is based on the variational auto-encoder framework (Kingma and Welling, 2013; Rezende et al., 2014), which encodes a natural language utterance with the context environment into a latent variable that can produce the correct logical form. We adopt meta-learning framework (Finn et al., 2017) to train a general semantic parser that can quickly adapt to a new (pseudo) task via few-shot learning, where multiple retrieved examples are viewed as a support set of a pseudo task. Our approach naturally make use of multiple similar examples to guide the semantic parser in the current task.

We evaluate our approach on CONCODE (Iyer et al., 2018) and CSQA (Saha et al., 2018) datasets, where tasks are generating source code conditioned on the class environment in JAVA codes and answering conversational question over a knowledge graph conditioned on conversational history. Results show that our approach achieves the state-of-the-art performances on both datasets. We show that coupling retrieval and meta-learning performs better than two retrieve-and-edit baselines. Further analysis show that both the context-aware retriever and the meta-learning strategy improve the performance.

2 Task Definition and Datasets

Context-dependent semantic parsing aims to map a natural language to a structural logical form conditioned on the context environment. In this section, we introduce two tasks we study, namely code generation and conversational question answering, and the datasets we use.

2.1 Context-dependent Code Generation

Figure 1 shows a example of code generation. Given a natural language (NL) description x, the goal aims to generate a source code y conditioned on the class environment c. Formally, the class environment comprises two kinds of context: (1) class variables v composed of variable names and their data type (e.g. $double[\ ]\ vecElements$), and (2) class methods m, including method names with their return type (e.g. $void\ add()$). We conduct experiments on the CONCODE[1] dataset (Iyer et al., 2018). The dataset is built from about 33,000 public Java projects on Github that contains NL and codes together with class environment information.

2.2 Conversational Question Answering

This task aims to answer questions in conversations based on a knowledge base (KB). We tackle the problem in a context-dependent semantic parsing manner. Specially, the task aims to map the question x conditioned on conversational history c into a logical form y, which will be executed on the KB to produce the answer. The conversational history refers to preceding questions $\{q_1, q_2, .., q_{i-1}\}$. In particular, we use the CSQA[2] dataset (Saha et al., 2018) to develop our model and to conduct the experiments. The dataset is created based on Wikidata with 12.8M entities, including 152K/16K/28K dialogs for training/development/testing.

3 Overview of the Approach

We present our approach in this section, which first retrieves supporting datapoints from the training dataset using a context-aware retriever, and then considers retrieved datapoints as a pseudo

[1] https://github.com/sriniiyer/concode
[2] https://amritasaha1812.github.io/CSQA

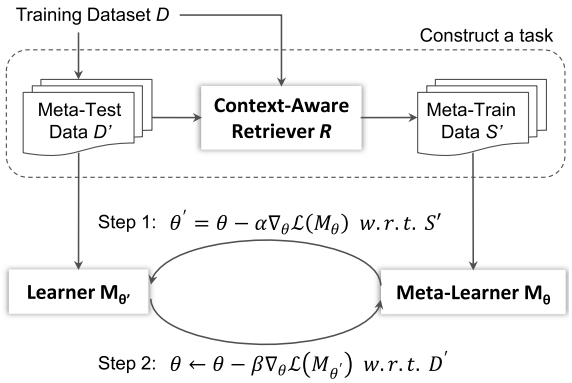

Figure 2: An overview of our approach that couples context-aware retriever and meta-learning.

task for fast adaptation in a model-agnostic meta-learning paradigm (Finn et al., 2017). Figure 2 gives an overview of our approach. First, we sample a batch of examples D' from the training dataset D. In meta-learning, there are two optimizing steps, namely the meta-train step (Step 1 in Figure 2) that learns a task-specific learner $M_{\theta'}$ based on the current parameter θ, and the meta-test step (Step 2 in Figure 2) that updates the parameter θ based on the evaluation of $M_{\theta'}$. In this work, D' is used for meta-test process, and retrieved examples S' from the context-aware retriever are used for meta-training. In the inference phase, We consider the prediction of each test example as a new task, given retrieved examples from the training data as the supporting evidence. Instead of applying the general model $M_{\theta'}$ directly, retrieved examples are used to update the model, and the updated model will be used to make predictions. The approach is summarized in Algorithm 1.

The details about the context-aware retriever and the semantic parser model will be introduced in Sections 4 and Section 5, respectively.

4 Context-Aware Retriever

In this section, we present the model architecture of our context-aware retriever, the way to use the model to retrieve similar examples using a distance metric in the latent space, and how to effectively train the model.

4.1 Model Architecture

Figure 3 illustrates an overview of the retrieval model in the task of generating source code. Following Hashimoto et al. (2018), our retriever is a encoder-decoder model based on the variational autoencoder framework, which encodes a natural

Algorithm 1 Retrieval-MAML

Input: Training dataset $D = (x^{(j)}, c^{(j)}, y^{(j)})$, step size α and β
Output: Meta-learner M

1: Training a context-aware retriever R using D.
2: For each example d, we obtain a support set S^d retrieved by R
3: Randomly initialize θ for M
4: **while** not done **do**
5: Sample a batch of examples D' from D as test examples, and $S' = \bigcup_{d \in D'} S^d$ are viewed as training examples
6: Evaluate $\nabla_\theta\mathcal{L}(M_\theta)$ using S', and compute adapted parameters with gradient descent: $\theta' = \theta - \alpha\nabla_\theta\mathcal{L}(M_\theta)$
7: Update $\theta \leftarrow \theta - \beta\nabla_\theta\mathcal{L}(M_{\theta'})$ using D' for meta-update
8: **end while**

language x with the context environment c into a latent variable z that can predict the output y.

Encoder We use bidirectional RNNs with LSTM (Hochreiter and Schmidhuber, 1997) as encoders to compute the representation h_x of the natural language x and the representation h_c of the context environment c, where h_x is the hidden states of the NL encoder at the last token and details about h_c for CONCODE and CSQA datasets are provided in the appendix A.

Latent variable We have two latent variables, one (z_x) is for the current utterance and another (z_c) is for the context. We use the concatenation of z_x and z_c as the embedding of the natural language with the context, namely $z = [z_x; z_c]$.

We describe how to map the the natural language x into a latent variable z_x here. The calculation of z_c is analogous to z_x. Following (Hashimoto et al., 2018), we choose z_x to be a von Mises-Fisher (vMF) distribution over unit vectors centered on μ_x, where both z_x and μ_x are unit vectors, and Z_κ is a normalization constant depending only on constant κ and the dimension d of z_x. The μ_x is calculated by a linear layer followed by a activation function, and the input is h_x.

$$p(z_x|x) = vMF(z_x; \mu_x, \kappa) = Z_\kappa^{-1} e^{(\kappa {\mu_x}^T z_x)} \quad (1)$$

Other distributions such as the Gaussian distribution can also be used to represent latent variables, but we choose the vMF in this paper since

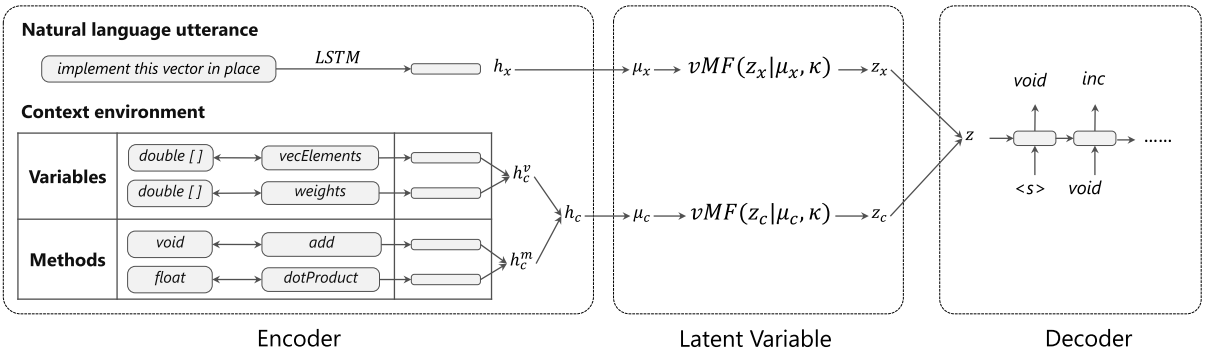

Figure 3: An overview of our context-dependent retriever.

the KV-divergence is proportional to the squared Euclidean distance between their respective direction vectors μ with the same κ and d. The property will be used in the next section.

Decoder At the decoding, we first sample z from $p(z|x,c)$ using the re-parametrization trick (Kingma and Welling, 2014), and then use an additional linear layer over z to obtain the initial hidden state of the decoder. We use LSTM as the decoder. At each time-step t, the current hidden state s_t of the decoder is used to predict a word from the vocabulary. In order to ensure that the target y is only inferred by the latent variable z, we don't incorporate attention or copying mechanism. The strategy is also used in Hashimoto et al. (2018).

4.2 Retrieve Examples

We use KL-divergence as the distance metric to retrieve similar examples in the latent space. In particular, the KL divergence between two vMF distributions with the same concentration parameter κ is calculated as follows, where $\mu_1, \mu_2 \in \mathbb{R}^{d-1}$ and C_κ is calculated as Equation 3.

$$KL(vMF(\mu_1,\kappa)||vMF(\mu_2,\kappa)) = C_\kappa||\mu_1-\mu_2||_2^2 \quad (2)$$

I_d stands for the modified Bessel function of the first kind at order d. Since C_κ only depends on κ and d, the KL divergence is proportional to the squared Euclidean distance between their respective direction vectors μ with the same κ and d. More details about the proof of this proposition can be found in (Hashimoto et al., 2018).

$$C_\kappa = \kappa \frac{I_{d/2}(\kappa)}{2I_{d/2-1}(\kappa)} \quad (3)$$

Given two examples (x,c) and (x',c'), the KL divergence between their distributions of latent variables (i.e. $p(z|x,c)$ and $p(z|x',c')$) is equivalent to the distance calculated as given in Equation 4. The retriever will find top-K nearest examples according to the distance.

$$\begin{aligned} distance &= KL(p(z|x,c)||p(z|x',c')) \\ &= KL(p(z_x|x)||p(z_x|x')) \\ &\quad + KL(p(z_c|c)||p(z_c|c')) \\ &= C_\kappa(||\mu_x-\mu_x'||_2^2 + ||\mu_c-\mu_c'||_2^2) \end{aligned} \quad (4)$$

4.3 Training

Our entire approach corresponds to the following generative process. Given a example (x,c), we first use the retriever $p_{ret}(S|x,c)$ to find similar examples S as a support set and then generate an output y by a meta-learner model $p_m(y|x,c,S)$ based on S. Therefore, the probability distribution over targets y is formulated in Equation 5.

$$p(y|x,c) = \sum_{S\subset D} p_m(y|x,c,S)p_{ret}(S|x,c) \quad (5)$$

A basic idea for learning the retriever might be maximizing the marginal likelihood by jointly learning, but it is computationally intractable. Instead, we train the retriever in isolation, assuming that semantic parser provides the true conditional distribution over the target y given context c and retrieved examples S under the joint distribution $p_{ret}(S|x,c)p_{data}(x,c,y)$. Then, we optimize a lower bound for the marginal likelihood under this semantic parser (Hashimoto et al., 2018), which decomposes the reconstruction term and the KL divergence as follows.

$$\begin{aligned} &logp(y|x,c) \geq E_{z\sim p(z|x,c)} logp(y|z) \\ &-E_{\{(x',c')\}\sim p_{ret}} KL(p(z|x,c)||p(z|x',c')) \end{aligned} \quad (6)$$

According to Equation 4, the upper bound of $KL(p(z|x,c)||p(z|x',c'))$ is $8C_\kappa$. Therefore, we

can maximize this worst-case lower bound, where C_κ is constant in our case. This lower bound objective is analogous to the recently proposed hyperspherical variational autoencoder (Davidson et al., 2018; Xu and Durrett, 2018).

$$logp(y|x,c) \geq E_{z\sim p(z|x,c)} logp(y|z) - 8C_\kappa \quad (7)$$

Thus, we optimize the context-aware retriever by maximizing $E_{z\sim p(z|x,c)} logp(y|z)$.

5 Semantic Parser

Recently, sequence-to-action models (Yin and Neubig, 2017; Chen et al., 2018; Iyer et al., 2018; Guo et al., 2018) have achieved strong performance in semantic parsing, which consider the generation of a logical form as the prediction of a sequence of actions (e.g. derivation rules in a defined grammar). We use two context-dependent sequence-to-action models (Iyer et al., 2018; Guo et al., 2018) as the base semantic parsers, both of which take a natural language with the context environment as the input and outputs an action sequence. Both models achieve state-of-the-art on CONCODE and CSQA datasets.

In the task of code generation, the JAVA abstract grammar contains a set of production rules composed of an non-terminal and multiple symbols (e.g. $Statement \rightarrow return\ Expression$). We represent a source code as an Abstract Syntax Tree (AST) by applying several production rules (Aho et al., 2007). The sequence of production rules applied to generate an AST is viewed as an action sequence $a_1, ..., a_n$, where an action a refers to a production rule. To access to the context environment, we introduce several special actions. For examples, two actions $IdentifierNT \rightarrow ClassMethod$ and $ClassMethod \rightarrow constant$ are used to invoke class methods. The former action means that the identifier comes from class methods, and the latter is an action used for instantiating $ClassMethod$ by a copying mechanism (Gu et al., 2016). In the task of conversational question answering, we convert the logical form into a action sequence using the similar grammar defined in (Guo et al., 2018).

Specially, a bidirectional LSTM takes a source sentence as the input, and feeds the concatenation of both ends as the initial state of the decoder. The decoder has another LSTM to generate an action sequence in a sequential way. At each time-step t, the decoder calculates the current hidden state s_t^{dec} as Equation 8, where s_{t-1}^{dec} is the last hidden state, n_t is the current non-terminal to be expanded, and y_{t-1} is the previously predicted action. p_{n_t} and s_{n_t} are the parent action and the hidden state of the decoder respectively, which produce the current non-terminal. If the previously predicted action is an instantiated action, the embedding y_{t-1} is the representation of the selected constant.

$$s_t^{dec} = LSTM(s_{t-1}^{dec}, [n_t; y_{t-1}; p_{n_t}; s_{n_t}]) \quad (8)$$

In order to generate a valid logical form, the model incorporates an action-constrained grammar to filter illegitimate actions. An action is legitimate if its left-hand non-terminal is the same as the current non-terminal to be expanded. Let us denote the set of legitimate actions at the time step t as $A_t = \{a_1, ..., a_N\}$. The probability distribution over the set is calculated as Equation 9, where v_i is the one-hot indicator vector for a_i, W_a is model parameter, and $a_{<t}$ stands for the preceding actions of the t-th time step.

$$p(a_i|a_{<t}, x) = \frac{exp(v_i^T W_a s_t^{dec})}{\sum_{a_j \in A_t} exp(v_j^T W_a s_t^{dec})} \quad (9)$$

For instantiated actions (e.g. $ClassMethod \rightarrow constant$), the probability of a constant m being instantiated at time-step t is calculated as Equation 10, where W is model parameter, v_m is the embedding of the constant.

$$p(m|a_{<t}, x) = \frac{exp(v_m^T tanh(W s_t^{dec}))}{\sum_{m'} exp(v_{m'}^T tanh(W s_t^{dec}))} \quad (10)$$

Please see more details about the model hyperparameters in the Appendix B.

6 Experiment

6.1 Model Comparisons on CONCODE

Table 1 reports results of different approaches on the CONCODE dataset. We use Exact match accuracy as the major evaluation metric, which measure whether the generated program is exactly correct. Following Iyer et al. (2018), we also report BLEU-4 score (Papineni et al., 2002) between the reference and generated code as a reference. These approaches are divided into three groups. The first group is retrieval ONLY, which directly returns the top-ranked retrieved example. The second group report numbers of existing systems and our base

Methods	Dev		Test	
	Exact	BLEU	Exact	BLEU
Retrieval ONLY				
TFIDF	1.25	17.78	1.50	19.73
Context-independent Retrieval	0.85	19.63	0.80	21.98
Context-dependent Retrieval	1.30	21.21	1.00	24.94
Parsing-based methods without retrieved examples				
Seq2Seq	2.90	21.00	3.20	23.51
Seq2Prod (Yin and Neubig, 2017)	5.55	21.00	6.65	21.29
Iyer et al. (2018)	7.05	21.28	8.60	22.11
Seq2Action	**7.75**	**22.47**	**9.15**	**23.34**
Parsing-based methods with retrieved examples				
Seq2Action+Edit vector (Context-independent Retrieval)	6.6	21.27	7.90	22.51
Seq2Action+Edit vector (Context-aware Retrieval)	7.75	20.69	9.20	22.68
Seq2Action+Retrieve-and-edit (Context-independent Retrieval)	5.55	21.27	7.05	22.74
Seq2Action+Retrieve-and-edit (Context-aware Retrieval)	7.55	22.20	9.30	23.95
Seq2Action+MAML (Context-independent Retrieval)	9.15	21.48	9.85	23.22
Seq2Action+MAML (Context-aware Retrieval, w/o finetune)	8.30	21.27	10.30	24.12
Seq2Action+MAML (Context-aware Retrieval)	**8.45**	**21.32**	**10.50**	**24.40**

Table 1: Performance of different approaches on the CONCODE dataset.

model Seq2Action, all of which do not use retrieved examples. Models in the last group utilize retrieved examples.

From the first group, we can see that directly using the retrieved output has extremely low Exact score since of mismatching environment variables and methods, which means that its meaning is incorrect. Yet, the BLEU score is acceptable, which means that some constituents might be useful. In the second group, we compare parsing based methods without retrieved examples. As we can see, our **Seq2Action** model outperforms others models, resulting in the state-of-the-art accuracy without using retrieved examples. In the third group, we implement two retrieval-augmented methods for comparison. **Retrieve-and-edit** uses a copying mechanism to copy tokens from the retrieved example (Hashimoto et al., 2018). **Edit vector** calculates an edit vector by considering lexical differences between a prototype context and current context, and uses the edit vector as an extra feature (Wu et al., 2018). We can see that applying the MAML framework to the Seq2Action model achieves a gain of 1.35% exact match accuracy. Results also show that our context-aware retriever performs better than the context-independent retriever in various settings.

6.2 Model Comparisons on CSQA

We follow the experiment protocol of Guo et al. (2018). To make the comparison clearer, we use F1 score as evaluation metrics for questions whose answers are sets of entities. Accuracy is used to measure the performance for questions which produce boolean and numerical answers. Table 2 shows the results of different methods on the CSQA dataset. More detailed numbers are provided in the appendix C. **HRED+KVmem** (Saha et al., 2018) is a encoder-decoder model with key-value memory network (Miller et al., 2016) to directly produce answers. **D2A** (Guo et al., 2018) is a sequence-to-action model described in Section 5. Since the dataset does not provide annotated action sequence for each question, we follow (Guo et al., 2018) to use a breadth-first-search algorithm to obtain action sequences that lead to correct answers. However, some of action sequences are spurious (Guu et al., 2017), in the sense they do not represent the meaning of questions but get the correct answers. We use retrieved examples by our context-aware model to filter out spurious action sequences. We choose the most similar action sequence to retrieved action sequences, measured by editing distance. We denote the model learned in this way as **S2A**.

Table 2 shows that filtering out spurious action

Methods	HRED +KVmem	D2A	S2A	S2A +EditVec	S2A +RAndE	S2A +MAML
Question Type	F1					
Simple Question (Direct)	13.64%	91.41%	92.01%	91.95%	92.08%	**92.66%**
Simple Question (Co-referenced)	7.26%	69.83%	71.40%	72.94%	**73.19%**	71.18%
Simple Question (Ellipsis)	9.95%	81.98%	81.75%	83.31%	**84.61%**	82.21%
Logical Reasoning (All)	8.33%	43.62%	42.00%	43.85%	41.83%	**44.34%**
Quantitative Reasoning (All)	0.96%	50.25%	45.37%	46.93%	42.64%	**50.30%**
Comparative Reasoning (All)	2.96%	44.20%	41.51%	43.96%	44.46%	**48.13%**
Clarification	16.35%	18.31%	18.9%	18.42%	18.70%	**19.12%**
Question Type	Accuracy					
Verification (Boolean)	21.04%	45.05%	51.17%	47.81%	**55.00%**	50.16%
Quantitative Reasoning (Count)	12.13%	40.94%	46.01%	44.67%	43.07%	**46.43%**
Comparative Reasoning (Count)	5.67%	17.78%	16.52%	17.52%	16.43%	**18.91%**

Table 2: Performance of different approaches on the CSQA dataset.

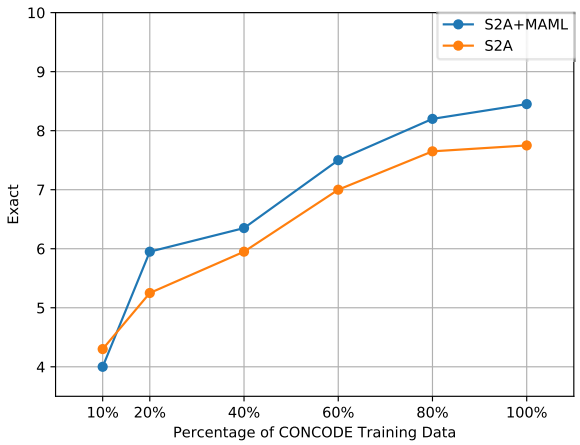

Figure 4: Comparison between S2A and S2A+MAML with different portions of supervised data.

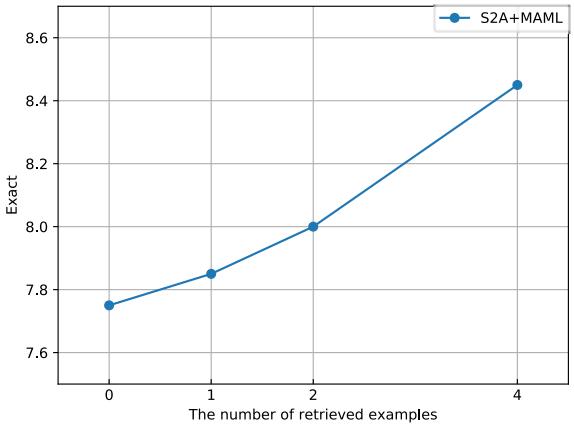

Figure 5: S2A+MAML with different number of retrieved examples on the CONCODE devset.

sequences brings about 5% point improvement on boolean and Quantitative Reasoning (Count) questions. Results also show that applying the MAML framework performs better than both retrieve-and-edit approaches, namely RAndE (Hashimoto et al., 2018) and EditVec (Wu et al., 2018), on the majority of question types, especially on complex questions.

6.3 Model Analysis

We study how the amount of training dataset and retrieved examples impacts the overall performance on the CONCODE. From Figure 4, we can see that S2A+MAML performs better than S2A in when >20% supervised datapoints are available to retrieve from. From Figure 5, we can see that the accuracy increases as the number of retrieved examples expands. This is consistent with our intuition that the performance of the semantic parser is improved by utilizing multiple retrieved examples, since the pattern of a logical form may come from different retrieved examples. We did not try larger number of retrieved examples due to the memory limit of our GPU device. However, excessive retrieved examples may introduce noise, which hurts the performance of the semantic parser. Therefore, we need to choose the appropriate amount of retrieved examples.

6.4 Case Study

We give a case study to illustrate the retrieved results by our context-aware retriever, with a comparison to the context-independent retriever. Results are given in Figure 6. We can see that our retriever can capture semantic content to retrieve. For examples, in the second row, the current question (i.g. *"who is the spouse of that one"*) has the

Input	Context-Aware Retriever	Context-Independent Retriever

Class environment:
HashMap<lalr_item, lalr_item> _all;
NL:
Does the set contain a particular item
Code:
boolean function(lalr_item arg0){
return _all.constainsKey(arg0); }
	Class environment:
	Map<Point, RailwayNode> _nodeMap;
	NL:
	Check if a node at a specific position exits.
	Code:
	boolean function(Point arg0){
	return _nodeMap.constainsKey(arg0);}
		Class environment:
		Node root;
		Node get(Node x, String key, int d);
		NL: Does the set contain the given key
		Code:
		boolean function(String arg0){
		Node loc0==get(root,arg0,0);
		if (loc0==null) return false;
		return loc0.isString; }

Q1: who is the dad of jorgen ottesen brahe? **A1**: otte brahe **Q2**: who is the spouse of that one?	**Q1**: whose child are gio batta bellotti? **A1**: matteo bellotti, paola cresipi guzzo **Q2**: which person is married to that one?	**Q1**: which abstract beings have marge simpson as an offspring? **A1**: clancy bouvier, jacqueline bouvier **Q2**: who is the spouse of that one?

Figure 6: Examples from the CONCODE dataset (first row) and the CSQA dataset (second row). The retrieved examples found by context-aware retriever (center panels) and context-independent retriever (right panels) follow the input (left panels).

same semantic as that one of the context-aware retrieved example (i.e. *"which person is married to that one"*), which demonstrates that our retriever learns the semantic of *"spouse"* and *"married"* in the retrieval process. Comparing with the context-independent retriever, incorporating the context environment can improve the performance of the retrieval. Taking the first row as a example, although the NL of the input (i.e. *"Does the set contain a particular item"*) have similar semantic to that one of the context-independent retriever (i.e. *"Does the set contain the given key"*), source codes differ greatly because the types of the sets are different (i.e. $HashMap$ and $Node$ respectively). Our context-aware retriever can find the example with similar source code by considering their context environment (both $HashMap$ and Map have same $constainsKey$ function).

6.5 Error Analysis

We analyze a randomly selected set of wrongly predicted 100 instances on the CONCODE dataset. We observe that 44% examples do not correctly copy class members, among which the majority of them lack information about class member (e.g. the effect of the class method get). This problem might be mitigated by encoding source codes of class methods or incorporating descriptions of class members. 24% examples fail to invoke functions of the library and member class (e.g. a model is required to know there exit a $size()$ function in List class to invoke $list.size()$). A potential direction to mitigate the problem is incorporate definitions of the external or system classes, which requires an updated version of the dataset. Among the other 32% examples, the major problem is that some of retrieved examples are incorrect. Incorporating more signal to measure the usefulness of retrieved examples might alleviate this problem.

7 Related work

Semantic parsing is a fundamental problem in NLP that maps natural languages to logical forms of their underlying meaning, including variable-free logic (Zelle and Mooney, 1995; Clarke et al., 2010), lambda calculus (Zettlemoyer and Collins, 2005; Kushman and Barzilay, 2013), dependency-based compositional semantics (Liang et al., 2011; Berant et al., 2013), and database queries (Iyer et al., 2017; Zhong et al., 2017; Liang et al., 2017). Recently, context-dependent semantic parsing has drawn plenty of attention (Long et al., 2016; Iyyer et al., 2017; Iyer et al., 2018; Suhr et al., 2018; Suhr and Artzi, 2018), where the generation of logical forms is conditioned on the context environment.

Neural encoder-decoder models have proved effective in semantic parsing (Neelakantan et al., 2015; Dong and Lapata, 2016; Yin and Neubig, 2017; Herzig and Berant, 2018). One direction is to employ sequence-to-sequence model by modeling semantic parsing as a sentence to logical form translation problem (Dong and Lapata, 2016; Jia and Liang, 2016; Ling et al., 2016; Xiao et al., 2016). However, regarding logical form as a se-

quence could not guarantee the grammatical correctness of the generated output. Sequence-to-Action approaches (Yin and Neubig, 2017; Krishnamurthy et al., 2017; Iyyer et al., 2017; Chen et al., 2018) treat semantic parsing as the prediction of a action sequence that can construct logical forms, which not only guarantee the grammatical correctness of outputs, but also leverage the strength of sequence-to-sequence model in learning sequential transformations.

Recently, there are recent attempts at exploiting retrieved examples to improve the generation of logical forms. Hashimoto et al. (2018) propose a retrieve-and-edit approach, including an encoder-decoder based retrieval model learnt in a task-dependent way without relying on a hand-crafted metric, and an editing model with a copying mechanism to replicate tokens from the retrieved example. Hayati et al. (2018) increase the probability of actions that can derive the retrieved subtrees. Huang et al. (2018a) also use MAML and treat each example as a new task. The relevance function for retrieving examples is based on the predicated type of the SQL query and the question length. Different from these three works, we focus on context-dependent semantic parsing, and our context-aware retriever is learned from the dataset without the help of a hand-craft relevant function. Different from (Hashimoto et al., 2018), our approach naturally make use of multiple similar examples to improve the semantic parser.

Retrieval-augmented models have also been studied in text generation (Gu et al., 2017; Huang et al., 2018b; Guu et al., 2018; Wu et al., 2018). Gu et al. (2017) use the retrieved sentence pairs as extra inputs to the NMT model. Wu et al. (2018) calculate an edit vector by considering lexical difference between a prototype context and current context, which is used as extra features.

8 Conclusion

In this paper, we present an approach which combines a context-aware retrieval model and model-agnostic meta-learning (MAML) to utilize multiple retrieved examples for context-dependent semantic parsing. We show that both context-aware retriever and MAML are useful on CONCODE and CSQA datasets. Our approach achieves the state-of-the-art performances and outperforms two retrieve-and-edit baselines.

Acknowledgments

This work is supported by the National Key R&D Program of China (2018YFB1004404), Key R&D Program of Guangdong Province (2018B010107005), National Natural Science Foundation of China (U1711262, U1401256, U1501252, U1611264, U1711261). Jian Yin is the corresponding author.

References

A Aho, M Lam, Ravi Sethi, and J Ullman. 2007. Compilers: Principles, techniques and tools, 2nd editio.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *EMNLP*, 5, page 6.

Bo Chen, Le Sun, and Xianpei Han. 2018. Sequence-to-action: End-to-end semantic graph generation for semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 766–777. Association for Computational Linguistics.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of the fourteenth conference on computational natural language learning*, pages 18–27. Association for Computational Linguistics.

Tim R Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M Tomczak. 2018. Hyperspherical variational auto-encoders. *arXiv preprint arXiv:1804.00891*.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org.

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 (Volume 1: Long Papers)*, pages 1631–1640. Association for Computational Linguistics.

Jiatao Gu, Yong Wang, Kyunghyun Cho, and Victor OK Li. 2017. Search engine guided non-parametric neural machine translation. *arXiv preprint arXiv:1705.07267*.

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*, pages 2946–2955.

Kelvin Guu, Tatsunori B Hashimoto, Yonatan Oren, and Percy Liang. 2018. Generating sentences by editing prototypes. *Transactions of the Association of Computational Linguistics*, 6:437–450.

Kelvin Guu, Panupong Pasupat, Evan Liu, and Percy Liang. 2017. From language to programs: Bridging reinforcement learning and maximum marginal likelihood. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1051–1062. Association for Computational Linguistics.

Tatsunori B Hashimoto, Kelvin Guu, Yonatan Oren, and Percy S Liang. 2018. A retrieve-and-edit framework for predicting structured outputs. In *Advances in Neural Information Processing Systems*, pages 10073–10083.

Shirley Anugrah Hayati, Raphael Olivier, Pravalika Avvaru, Pengcheng Yin, Anthony Tomasic, and Graham Neubig. 2018. Retrieval-based neural code generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 925–930. Association for Computational Linguistics.

Jonathan Herzig and Jonathan Berant. 2018. Decoupling structure and lexicon for zero-shot semantic parsing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1619–1629. Association for Computational Linguistics.

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

Po-Sen Huang, Chenglong Wang, Rishabh Singh, Wen-tau Yih, and Xiaodong He. 2018a. Natural language to structured query generation via meta-learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 732–738. Association for Computational Linguistics.

Shaohan Huang, Yu Wu, Furu Wei, and Ming Zhou. 2018b. Dictionary-guided editing networks for paraphrase generation. *arXiv preprint arXiv:1806.08077*.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a neural semantic parser from user feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 963–973.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2018. Mapping language to code in programmatic context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1643–1652. Association for Computational Linguistics.

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 (Volume 1: Long Papers)*, volume 1, pages 1821–1831.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12–22. Association for Computational Linguistics.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational bayes. *international conference on learning representations*.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1516–1526, Copenhagen, Denmark. Association for Computational Linguistics.

Nate Kushman and Regina Barzilay. 2013. Using semantic unification to generate regular expressions from natural language. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 826–836.

Chen Liang, Jonathan Berant, Quoc 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 (Volume 1: Long Papers)*, pages 23–33. Association for Computational Linguistics.

Percy Liang, Michael I Jordan, and Dan Klein. 2011. Learning dependency-based compositional semantics. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, pages 590–599.

Wang Ling, Phil Blunsom, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiský, Fumin Wang, and Andrew Senior. 2016. Latent predictor networks for code generation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 599–609. Association for Computational Linguistics.

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 (Volume 1: Long Papers)*, pages 1456–1465. Association for Computational Linguistics.

Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. 2016. Key-value memory networks for directly reading documents. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1400–1409. Association for Computational Linguistics.

Arvind Neelakantan, Quoc V Le, and Ilya Sutskever. 2015. Neural programmer: Inducing latent programs with gradient descent. *arXiv preprint arXiv:1511.04834*.

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 on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1278–1286, Bejing, China. PMLR.

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. *arXiv preprint arXiv:1801.10314*.

Alane Suhr and Yoav Artzi. 2018. Situated mapping of sequential instructions to actions with single-step reward observation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pages 2072–2082. Association for Computational Linguistics.

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, Volume 1 (Long Papers)*, pages 2238–2249. Association for Computational Linguistics.

Yu Wu, Furu Wei, Shaohan Huang, Zhoujun Li, and Ming Zhou. 2018. Response generation by context-aware prototype editing. *arXiv preprint arXiv:1806.07042*.

Chunyang Xiao, Marc Dymetman, and Claire Gardent. 2016. Sequence-based structured prediction for semantic parsing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1341–1350.

Jiacheng Xu and Greg Durrett. 2018. Spherical latent spaces for stable variational autoencoders. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4503–4513. Association for Computational Linguistics.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 440–450. Association for Computational Linguistics.

John M Zelle and Raymond J Mooney. 1995. Comparative results on using inductive logic programming for corpus-based parser construction. In *International Joint Conference on Artificial Intelligence*, pages 355–369. Springer.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI '05, Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence*, pages 658–666.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2sql: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

A Encoders for Context-Aware Retriever

To obtain the representation of the context environment h_c, we first use a 2-step Bi-LSTM to encode the variable to get the contextual representation of class variables h_v^i. The representations of class methods h_m^i are also obtained in the same way. Usually, identifiers are composed of multiple words, such as $vecElements$. We split them based on camel-casing and encode them to get corresponding embedding by a Bi-LSTM. Finally, the representation of the context environment h_c is calculated by average pooling over the vectors.

Different from code generation, the context of conversational question answering refers to previous questions $\{q_1, q_2, .., q_{i-1}\}$. Therefore, we use a bidirectional LSTM to encode them to get the representation $\{h_{q_1}, h_{q_2}, .., h_{q_{i-1}}\}$. The representation of the context environment h_c is calculated by average pooling over the vectors.

B Model Training

For the context-aware retriever on both experiment, we set the dimension of the word embed-

Methods	HRED+KVmem		D2A		S2A	
Question Type	Recall	Precision	Recall	Precision	Recall	Precision
Overall	18.40%	6.30%	64.04%	61.76%	64.86%	62.51%
Simple Question (Direct)	33.30%	8.58%	93.67%	89.26%	93.19%	90.86%
Simple Question (Co-referenced)	12.67%	5.09%	71.31%	68.41%	72.96%	69.91%
Simple Question (Ellipsis)	17.30%	6.98%	86.58%	77.85%	85.24%	78.54%
Logical Reasoning (All)	15.11%	5.75%	42.49%	44.82%	44.30%	39.93%
Quantitative Reasoning (All)	0.91%	1.01%	48.59%	52.03%	45.98%	44.77%
Comparative Reasoning (All)	2.11%	4.97%	44.73%	43.69%	43.23%	39.93%
Clarification	25.09%	12.13%	19.36%	17.36%	19.84%	18.04%
Question Type	Accuracy		Accuracy		Accuracy	
Verification (Boolean)	21.04%		45.05%		51.17%	
Quantitative Reasoning (Count)	12.13%		40.94%		46.01%	
Comparative Reasoning (Count)	8.67%		17.78%		16.52%	

Table 3: Results of methods without utilizing retrieved examples on the CSQA dataset.

Methods	S2A+EditVec		S2A+RAndE		S2A+MAML	
Question Type	Recall	Precision	Recall	Precision	Recall	Precision
Overall	65.51%	63.45%	65.54%	63.12%	65.23%	63.02%
Simple Question (Direct)	93.47%	90.48%	93.72%	90.50%	94.43%	90.95%
Simple Question (Co-referenced)	74.11%	71.81%	74.47%	71.96%	72.72%	69.70%
Simple Question (Ellipsis)	87.01%	79.91%	88.06%	81.42%	85.89%	78.84%
Logical Reasoning (All)	42.21%	45.63%	40.55%	43.20%	42.59%	46.23%
Quantitative Reasoning (All)	48.26%	45.67%	45.44%	40.17%	50.77%	49.83%
Comparative Reasoning (All)	46.60%	41.60%	47.08%	42.11%	48.32%	47.95%
Clarification	19.30%	17.61%	19.81%	17.71%	20.01%	18.31%
Question Type	Accuracy		Accuracy		Accuracy	
Verification (Boolean)	47.81%		55.00%		50.16%	
Quantitative Reasoning (Count)	44.67%		43.07%		46.43%	
Comparative Reasoning (Count)	17.52%		16.43%		18.91%	

Table 4: Results of methods with utilizing retrieved examples on the CSQA dataset.

ding as 300. The encoder is a 2-layer bidirectional LSTM with hidden states of size 300, and the decoder is a 4-layer unidirectional LSTM with hidden states of size 300. We use dropout with a rate of 0.5, which is applied to the inputs of RNN. We set the dimension of latent variable and κ as 600 and 500, respectively. Model parameters are initialized with uniform distribution, and updated with the Adam method. We set the learning rate as 0.0001 and the batch size as 20. We tune hyper parameters and perform early stopping on the development set.

The hyperparameters of encoder and decoder are the same as the context-aware retriever. We follow (Huang et al., 2018a) to train the meta-learner without back-propagating to second order gradients. The number of retrieved examples for the CONCODE dataset and CSQA dataset are 4 and 1 respectively, which are tuned on the development sets. We set the step size α of task update as 0.001 for both experiment. On the CONCODE dataset, the learning rate β is 0.0002 and the test-batch size is 10, while the CSQA dataset are 0.001 and 32 respectively.

C Results on CSQA

Here, we provide more detailed numbers about the performance of different approaches on the CSQA dataset. Precision and recall are used as evaluation metrics for questions whose answers are sets of entities. Accuracy is used to measure the performance for questions which produce boolean and numerical answers. Table 3 shows the results of methods without utilizing retrieved examples, and Table 4 shows the results of retrieval-augmented methods.