A Mutual Information Maximization Approach for the Spurious Solution Problem in Weakly Supervised Question Answering

Zhihong Shao[1], Lifeng Shang[2], Qun Liu[2], Minlie Huang[1]*
[1]The CoAI group, DCST, Tsinghua University, Institute for Artificial Intelligence;
[1]State Key Lab of Intelligent Technology and Systems;
[1]Beijing National Research Center for Information Science and Technology;
[1]Tsinghua University, Beijing 100084, China
[2]Huawei Noah's Ark Lab
szh19@mails.tsinghua.edu.cn, aihuang@tsinghua.edu.cn
{shang.lifeng, qun.liu}@huawei.com

Abstract

Weakly supervised question answering usually has only the final answers as supervision signals while the correct solutions to derive the answers are not provided. This setting gives rise to the *spurious solution problem*: there may exist many spurious solutions that coincidentally derive the correct answer, but training on such solutions can hurt model performance (e.g., producing wrong solutions or answers). For example, for discrete reasoning tasks as on DROP, there may exist many equations to derive a numeric answer, and typically only one of them is correct. Previous learning methods mostly filter out spurious solutions with heuristics or using model confidence, but do not explicitly exploit the semantic correlations between a question and its solution. In this paper, to alleviate the spurious solution problem, we propose to explicitly exploit such semantic correlations by maximizing the mutual information between question-answer pairs and predicted solutions. Extensive experiments on four question answering datasets show that our method significantly outperforms previous learning methods in terms of task performance and is more effective in training models to produce correct solutions.

1 Introduction

Weakly supervised question answering is a common setting of question answering (QA) where only final answers are provided as supervision signals while the correct solutions to derive them are not. This setting simplifies data collection, but exposes model learning to the *spurious solution problem*: there may exist many spurious ways to derive the correct answer, and training a model with spurious solutions can hurt model performance (e.g., misleading the model to produce unreasonable solutions or wrong answers). As shown in Fig 1, for multi-mention reading comprehension, many mentions of an answer in the document(s) are irrelevant to the question; for discrete reasoning tasks or text2SQL tasks, an answer can be produced by the equations or SQL queries that do not correctly match the question in logic.

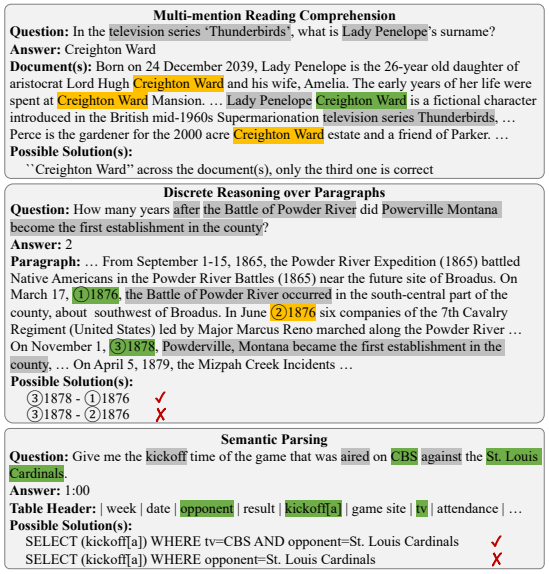

Figure 1: Examples from three weakly supervised QA tasks, i.e., multi-mention reading comprehension, discrete reasoning, and semantic parsing. Spans in dark gray and green denote semantic correlations between a question and its solution, while spans in orange are spurious information and should not be used in a solution.

Some previous works heuristically selected one possible solution per question for training, e.g., the first answer span in the document (Joshi et al., 2017; Tay et al., 2018; Talmor and Berant, 2019); some treated all possible solutions equally and maximized the sum of their likelihood (maximum marginal likelihood, or MML) (Swayamdipta et al., 2018; Clark and Gardner, 2018; Lee et al., 2019); many others selected solutions according to model confidence (Liang et al., 2018; Min et al., 2019),

*Corresponding author: Minlie Huang.

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 4111–4124
August 1–6, 2021.

i.e., the likelihood of the solutions being derived by the model. A drawback of these methods is that they do not explicitly consider the mutual semantic correlations between a question and its solution when selecting solutions for training.

Intuitively speaking, a question often contains vital clues about how to derive the answer, and a wrong solution together with its context often fails to align well with the question. Take the discrete reasoning case in Fig 1 as an example. To answer the question, we need to know the start year of *the Battle of Powder River*, which is answered by the first *1876*; the second *1876* is irrelevant as it is the year of an event that happened during the battle.

To exploit the semantic correlations between a question and its solution, we propose to maximize the mutual information between question-answer pairs and model-predicted solutions. As demonstrated by Min et al. (2019), for many QA tasks, it is feasible to precompute a modestly-sized, task-specific set of possible solutions containing the correct one. Therefore, we focus on handling the spurious solution problem under this circumstance. Specifically, we pair a task-specific model with a question reconstructor and repeat the following training cycle (Fig 2): (1) sample a solution from the solution set according to model confidence, train the question reconstructor to reconstruct the question from that solution, and then (2) train the task-specific model on the most likely solution according to the question reconstructor. During training, the question reconstructor guides the task-specific model to predict those solutions consistent with the questions. For the question reconstructor, we devise an effective and unified way to encode solutions in different tasks, so that solutions with subtle differences (e.g., different spans with the same surface form) can be easily discriminated.

Our contributions are as follows: **(1)** We propose a mutual information maximization approach for the spurious solution problem in weakly supervised QA, which exploits the semantic correlations between a question and its solution; **(2)** We conducted extensive experiments on four QA datasets. Our approach significantly outperforms strong baselines in terms of task performance and is more effective in training models to produce correct solutions.

2 Related Work

Question answering has raised prevalent attention and has achieved great progress these years. A lot of challenging datasets have been constructed to advance models' reasoning abilities, such as (1) reading comprehension datasets with extractive answer spans (Joshi et al., 2017; Dhingra et al., 2017), with free-form answers (Kociský et al., 2018), for multi-hop reasoning (Yang et al., 2018), or for discrete reasoning over paragraphs (Dua et al., 2019), and (2) datasets for semantic parsing (Pasupat and Liang, 2015; Zhong et al., 2017; Yu et al., 2018). Under the weakly supervised setting, the specific solutions to derive the final answers (e.g., the correct location of an answer text, or the correct logic executing an answer) are not provided. This setting is worth exploration as it simplifies annotation and makes it easier to collect large-scale corpora. However, this setting introduces the spurious solution problem, and thus complicates model learning.

Most existing approaches for this learning challenge include heuristically selecting one possible solution per question for training (Joshi et al., 2017; Tay et al., 2018; Talmor and Berant, 2019), training on all possible solutions with MML (Swayamdipta et al., 2018; Clark and Gardner, 2018; Lee et al., 2019; Wang et al., 2019), reinforcement learning (Liang et al., 2017, 2018), and hard EM (Min et al., 2019; Chen et al., 2020). All these approaches either use heuristics to select possibly reasonable solutions, rely on model architectures to bias towards correct solutions, or use model confidence to filter out spurious solutions in a soft or hard way. They do not explicitly exploit the semantic correlations between a question and its solution.

Most relevantly, Cheng and Lapata (2018) focused on text2SQL tasks; they modeled SQL queries as the latent variables for question generation, and maximized the evidence lower bound of log likelihood of questions. A few works treated solution prediction and question generation as dual tasks and introduced dual learning losses to regularize learning under the fully-supervised or the semi-supervised setting (Tang et al., 2017; Cao et al., 2019; Ye et al., 2019). In dual learning, a model generates intermediate outputs (e.g., the task-specific model predicts solutions from a question) while the dual model gives feedback signals (e.g., the question reconstructor computes the likelihood of the question conditioned on predicted solutions). This method is featured in three aspects. First, both models need training on fully-annotated data so that they can produce reasonable intermediate outputs. Second, the intermediate outputs can

introduce noise during learning as they are sampled from models but not restricted to solutions with correct answer or valid questions. Third, this method typically updates both models with reinforcement learning while the rewards provided by a dual model can be unstable or of high variance. By contrast, we focus on the spurious solution problem under the weakly supervised setting and propose a mutual information maximization approach. Solutions used for training are restricted to those with correct answer. What's more, though a task-specific model and a question reconstructor interact with each other, they do not use the likelihood from each other as rewards, which can stabilize learning.

3 Method

3.1 Task Definition

For a QA task, each instance is a tuple $\langle d, q, a\rangle$, where q denotes a question, a is the answer, and d is reference information such as documents for reading comprehension, or table headers for semantic parsing. A solution z is a task-specific derivation of the answer, e.g., a particular span in a document, an equation, or a SQL query (as shown in Fig 1). Let $f(\cdot)$ be the task-specific function that maps a solution to its execution result, e.g., by returning a particular span, solving an equation, or executing a SQL query. Our goal is to train a task-specific model $P_\theta(z|d, q)$ that takes $\langle d, q\rangle$ as input and predicts a solution z satisfying $f(z) = a$.

Under the weakly supervised setting, only the answer a is provided for training while the ground-truth solution $\bar{z}$ is not. We denote the set of possible solutions as $Z = \{z|f(z) = a\}$. In cases where the search space of solution is large, we can usually approximate Z so that it contains the ground-truth solution $\bar{z}$ with a high probability (Min et al., 2019; Wang et al., 2019). Note that Z is task-specific, which will be instantiated in section 4.

During training, we pair the task-specific model $P_\theta(z|d, q)$ with a question reconstructor $P_\phi(q|d, z)$ and maximize the mutual information between $\langle q, a\rangle$ and z. During test, given $\langle d, q\rangle$, we use the task-specific model to predict a solution and return the execution result.

3.2 Learning Method

Given an instance $\langle d, q, a\rangle$, the solution set Z usually contains only one solution that best fits the instance while the rest are spurious. We propose to exploit the semantic correlations between a question and its solution to alleviate the spurious solution problem via mutual information maximization.

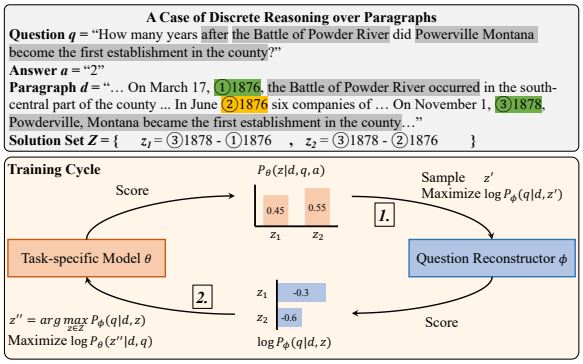

Figure 2: Illustration of the learning method.

Our objective is to obtain the optimal task-specific model θ^* that maximizes the following conditional mutual information:

$$\begin{aligned}\theta^* &= \arg\max_\theta I_\theta(\langle q, a\rangle; z|d)\\ &= \arg\max_\theta H(\langle q, a\rangle|d) - H_\theta(\langle q, a\rangle|d, z)\\ &= \arg\max_\theta -H_\theta(\langle q, a\rangle|d, z)\\ &= \arg\max_\theta E_{P(d,q,a)} E_{P_\theta(z|d,q,a)} \log P_\theta(q, a|d, z)\end{aligned} \tag{1}$$

where $I_\theta(\langle q, a\rangle; z|d)$ denotes conditional mutual information between $\langle q, a\rangle$ and z over $P(d, q, a)P_\theta(z|d, q, a)$. $H(\cdot|\cdot)$ is conditional entropy of random variable(s). $P(d, q, a)$ is the probability of an instance from the training distribution. $P_\theta(z|d, q, a)$ is the *posterior prediction probability* of z ($\in Z$) which is the prediction probability $P_\theta(z|d, q)$ normalized over Z:

$$P_\theta(z|d, q, a) = \begin{cases} \frac{P_\theta(z|d,q)}{\sum_{z' \in Z} P_\theta(z'|d,q)} & z \in Z \\ 0 & z \notin Z \end{cases} \tag{2}$$

Note that computing $P_\theta(q, a|d, z)$ is intractable. We therefore introduce a question reconstructor $P_\phi(q|d, z)$ and approximate $P_\theta(q, a|d, z)$ with $\mathbb{I}(f(z) = a)P_\phi(q|d, z)$ where $\mathbb{I}(\cdot)$ denotes indicator function. Eq. 1 now becomes:

$$\begin{aligned}\theta^* &= \arg\max_\theta \mathcal{L}_1 + \mathcal{L}_2\\ \mathcal{L}_1 &= E_{P(d,q,a)} E_{P_\theta(z|d,q,a)} \log P_\phi(q|d, z)\\ \mathcal{L}_2 &= E_{P(d,q,a)} E_{P_\theta(z|d,q,a)} \log \frac{P_\theta(q, a|d, z)}{P_\phi(q|d, z)}\end{aligned} \tag{3}$$

To optimize Eq. 3 is to repeat the following training cycle, which is analogous to the EM algorithm:

1. Minimize $\mathcal{L}_2$ w.r.t. the question reconstructor ϕ to draw $P_\phi(q|d, z)$ close to $P_\theta(q, a|d, z)$, by sampling a solution $z' \in Z$ according to its posterior prediction probability $P_\theta(z|d, q, a)$ (see Eq. 2) and maximizing $\log P_\phi(q|d, z')$.

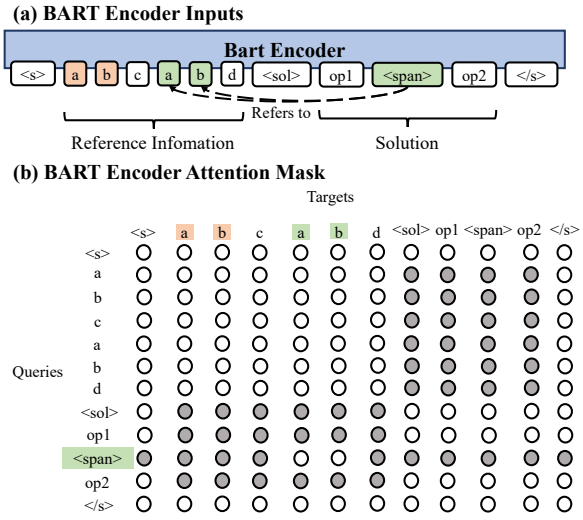

Figure 3: Solution encoding. (a) For BART encoder inputs, $\langle s \rangle$ and $\langle /s \rangle$ denote start and end of input sequence, respectively. $\langle sol \rangle$ denotes start of solution. $\langle span \rangle$ is the placeholder of the referred span in reference information (e.g., the second *ab* in this figure. (b) For attention mask, gray circles block attention. $\langle span \rangle$ retrieves the contextual representation(s) of the referred span by only attending to the referred span. reference information and the solution (except for the token $\langle span \rangle$) are kept from attending to each other.

2. Maximize $\mathcal{L}_1$ w.r.t. the task-specific model θ. $\mathcal{L}_1$ can be seen as a reinforcement learning objective with $\log P_\phi(q|d,z)$ being the reward function. During training, the reward function is dynamically changing and may be of high variance. As we can compute the reward for all $z \in Z$, we therefore adopt a greedy but more stable update method, i.e., to maximize $\log P_\theta(z''|d,q)$ where $z'' = \arg\max_{z \in Z} \log P_\phi(q|d,z)$ is the best solution according to the question reconstructor.

We illustrate the above training cycle in Fig 2.

3.3 Question Reconstructor

The question reconstructor $P_\phi(q|d,z)$ takes reference information d and a solution z as input, and reconstructs the question q. We use $\text{BART}_{\text{base}}$, a pre-trained Seq2Seq model, as the question reconstructor so that semantic correlations between questions and solutions can be better captured.

A solution typically consists of task-specific operation token(s) (e.g., COUNT for discrete reasoning or semantic parsing), literal(s) (e.g., numeric constants for discrete reasoning or semantic parsing), or span(s) from a question or reference information (e.g., for most QA tasks). It is problematic to just feed the concatenation of d and the surface form of z to the BART encoder; otherwise, different spans with the same surface form can no longer be discriminated as their contextual semantics are lost. To effectively encode d and z, we devise a unified solution encoding as in Fig 3 which is applicable to solutions of various types. Specifically, we leave most of the surface form of z unchanged, except that we replace any span from reference information with a placeholder $\langle span \rangle$. The representation of $\langle span \rangle$ is computed by forcing it to only attend to the contextual representation(s) of the referred span. To obtain disentangled and robust representations of reference information and a solution, we keep reference information and the solution (except for the token $\langle span \rangle$) from attending to each other. Intuitively speaking, semantics of reference information should not be affected by a solution, and the representations of a solution should largely determined by its internal logic.

3.4 Solution Set

While our learning method and question reconstructor are task-agnostic, solutions are usually task-specific. Precomputing solution sets needs formal definitions of solutions which define the search space of solutions. A possible search method is to exhaustively enumerate all solutions that produce the correct answer. We will introduce the definitions of solutions for different tasks in section 4.

4 Experiments

Datasets	# Examples			$\lvert Z \rvert$	
	Train	Dev	Test	Avg	Median
Multi-mention Reading Comprehension					
Quasar-T	37,012	3,000	3,000	8.1	4
WebQuestions	3,778	-	2,032	52.1	36
Discrete Reasoning over Paragraphs					
DROP	69,669	7,740	9,535	5.1	2
Semantic Parsing					
WikiSQL	56,355	8,421	15,878	315.4	4

Table 1: Statistics of the datasets we used. Statistics of the size of solution set $|Z|$ are computed on Train sets.

Following Min et al. (2019), we conducted experiments on three QA tasks, namely multi-mention reading comprehension, discrete reasoning over paragraphs, and semantic parsing. This section introduces baselines, the definitions of solutions in different tasks, how the solution set can be precomputed, and our experimental results. Statistics of the datasets we used are presented in Table 1.

For convenience, we denote reference information as $d = [d_1, d_2, ..., d_{|d|}]$ and denote a question as $q = [q_1, q_2, ..., q_{|q|}]$ where d_i and q_j are a token of d and q respectively. A span from reference information and a question span is represented as $(s, e)^d$ and $(s, e)^q$ respectively, where s and e are start and end index of the span respectively.

4.1 Baselines

First Only (Joshi et al., 2017) which trains a reading comprehension model by maximizing $\log P_\theta(z|d, q)$ where z is the first answer span in d.
MML (Min et al., 2019) which maximizes $\log \sum_{z \in Z} P_\theta(z|d, q)$.
HardEM (Min et al., 2019) which maximizes $\log max_{z \in Z} P_\theta(z|d, q)$.
HardEM-thres (Chen et al., 2020): a variant of HardEM that optimizes only on confident solutions, i.e., to maximize $max_{z \in Z} \mathbb{I}(P_\theta(z|d, q) > \gamma) \log P_\theta(z|d, q)$ where γ is an exponentially decaying threshold. γ is initialized such that a model is trained on no less than half of training data at the first epoch. We halve γ after each epoch.
VAE (Cheng and Lapata, 2018): a method that views a solution as the latent variable for question generation and adopts the training objective of Variational Auto-Encoder (VAE) (Kingma and Welling, 2014) to regularize the task-specific model. The overall training objective is given by:

$$\theta^*, \phi^* = \arg\max_{\theta, \phi} \mathcal{L}(\theta, \phi)$$

$$\mathcal{L}(\theta, \phi) = \mathcal{L}^{mle}(\theta) + \lambda \mathcal{L}^{vae}(\theta, \phi)$$

$$= \sum_{z \in B} \log P_\theta(z|d, q) + \lambda E_{P_\theta(z|d,q)} \log \frac{P_\phi(q|d, z)}{P_\theta(z|d, q)}$$

where θ denotes a task-specific model and ϕ is our question reconstructor. $\mathcal{L}^{mle}(\theta)$ is the total log likelihood of the set of model-predicted solutions (denoted by B) which derive the correct answer. $\mathcal{L}^{vae}(\theta, \phi)$ is the evidence lower bound of the log likelihood of questions. λ is the coefficient of $\mathcal{L}^{vae}(\theta, \phi)$. This method needs pre-training both θ and ϕ before optimizing the overall objective $\mathcal{L}(\theta, \phi)$. Notably, model θ optimizes on $\mathcal{L}^{vae}(\theta, \phi)$ via reinforcement learning. We tried stabilizing training by reducing the variance of rewards and setting a small λ.

4.2 Multi-Mention Reading Comprehension

Multi-mention reading comprehension is a natural feature of many QA tasks. Given a document d and a question q, a task-specific model is required to locate the answer text a which is usually mentioned many times in the document(s). A solution is defined as a document span. The solution set Z is computed by finding exact match of a:

$$Z = \{z = (s, e)^d | [d_s, ..., d_e] = a\}$$

We experimented on two open domain QA datasets, i.e., Quasar-T (Dhingra et al., 2017) and WebQuestions (Berant et al., 2013). For Quasar-T, we retrieved 50 reference sentences from ClueWeb09 for each question; for WebQuestions, we used the 2016-12-21 dump of Wikipedia as the knowledge source and retrieved 50 reference paragraphs for each question using a Lucene index system. We used the same $\text{BERT}_{\text{base}}$ (Devlin et al., 2019) reading comprehension model and data preprocessing from (Min et al., 2019).

	Quasar-T				WebQuestions	
	Dev		Test		Test	
	EM	F1	EM	F1	EM	F1
First Only	36.0	43.9	35.6	42.8	16.7	22.6
MML	40.1	47.4	39.1	46.5	18.4	25.0
HardEM	41.5	49.1	40.7	47.7	18.0	24.2
HardEM-thres	42.8	50.2	41.9	49.4	19.0	25.3
Ours	**44.7**‡	**52.6**‡	**44.0**‡	**51.5**‡	**20.4**‡	**27.2**‡

Table 2: Evaluation on multi-mention reading comprehension datasets. Numbers marked with ‡ are significantly better than the others (t-test, p-value < 0.05).

Results: Our method outperforms all baselines on both datasets (Table 2). The improvements can be attributed to the effectiveness of solution encoding, as solutions for this task are typically different spans with the same surface form, e.g., in Qusart-T, all $z \in Z$ share the same surface form.

4.3 Discrete Reasoning over Paragraphs

Some reading comprehension tasks pose the challenge of comprehensive analysis of texts by requiring discrete reasoning (e.g., arithmetic calculation, sorting, and counting) (Dua et al., 2019). In this task, given a paragraph d and a question q, an answer a can be one of the four types: numeric value, a paragraph span or a question span, a sequence of paragraph spans, and a date from the paragraph. The definitions of z depend on answer types (Table 4). These solutions can be searched by following Chen et al. (2020). Note that some solutions involve numbers in d. We treated those numbers as spans while reconstructing q from z.

We experimented on DROP (Dua et al., 2019). As the original test set is hidden, for convenience of

	Overall Test		Number (61.97%)		Span (31.47%)		Spans (4.99%)		Date (1.57%)	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
MML	58.99‡	62.30‡	55.38	55.58	69.96	75.51	39.29	66.01	42.57	49.05
HardEM	68.52‡	71.88‡	68.40	68.70	73.50	79.25	44.79	69.63	49.32	56.87
HardEM-thres	69.06	72.35‡	69.05	69.39	74.61	79.79	39.50	66.38	52.67	58.75
VAE	32.34‡	36.28‡	51.65	52.35	0.37	10.01	0.00	8.89	0.00	4.11
Ours	**69.35**	**72.92**	**69.96**	**70.27**	73.38	79.32	42.86	**70.42**	48.67	57.47

Table 3: Evaluation on DROP. We used the public development set of DROP as our test set. We also provide performance breakdown of different question types on our test set. Results on the overall test set marked with ‡ are significantly worse than the best one (t-test, p-value < 0.05).

Numeric Answers	
Arithmetic	$z = n_1[, o_1, n_2[, o_2, n_3]]$, s.t. $o_1, o_2 \in \{+, -\}$, $n_1, n_2, n_3 \in N_d \cup S$
Sorting	$z = o\{n_k\}_{k\geq 1}$, s.t. $o \in \{max, min\}, n_k \in N_d$
Counting	$z = \lvert\{(s_k, e_k)^d\}_{k\geq 1}\rvert$
Non-numeric Answers	
Span(s)	$z = \{(s_k, e_k)^t\}_{k\geq 1}$, s.t. $t \in \{d, q\}$
Sorting	$z = o\{kv\langle(s_k, e_k)^d, n_k\rangle\}_{k\geq 1}$, s.t. $o \in \{argmax, argmin\}, n_k \in N_d$

Table 4: Definitions of solutions for numeric answers and non-numeric answers. N_d is the set of numbers in d, and S is a set of pre-defined numbers. For arithmetic solutions for numeric answers, $z = n_1[, o_1, n_2[, o_2, n_3]]$ denotes equations with no more than three operands. For solutions of sorting type for non-numeric answers, $kv\langle\cdot,\cdot\rangle$ is a key-value pair where the key is a span in d and the value is its associated number from d. $argmax$ ($argmin$) returns the key with the largest (smallest) value.

analysis, we used the public development set as our test set, and split the public train set into 90%/10% for training and development. We used Neural Symbolic Reader (NeRd) (Chen et al., 2020) as the task-specific model. NeRd is a Seq2Seq model which encodes a question and a paragraph, and decodes a solution (e.g., *count (paragraph_span(s_1, e_1), paragraph_span(s_2, e_2))* where *paragraph_span(s_i, e_i)* means a paragraph span starting at s_i and ending at e_i). We used the precomputed solution sets provided by Chen et al. (2020)[1]. Data preprocessing was also kept the same.

Results: As shown in Table 3, our method significantly outperforms all baselines in terms of F1 score on our test set.

We also compared our method with the baseline VAE which uses a question reconstructor ϕ to adjust the task-specific model θ via maximizing a variational lower bound of $\log P(q|d)$ as the regularization term $\mathcal{L}^{vae}(\theta, \phi)$. To pre-train the task-specific model for this method, we simply obtained the best task-specific model trained with HardEM-thres. VAE optimizes the task-specific model on $\mathcal{L}^{vae}(\theta, \phi)$ with reinforcement learning where $P_\phi(q|d, z)$ is used as learning signals for the task-specific model. Despite our efforts to stabilize training, the F1 score still dropped to 36.28 after optimizing the overall objective $\mathcal{L}(\theta, \phi)$ for 1,000 steps. By contrast, our method does not use $P_\phi(q|d, z)$ to compute learning signals for the task-specific model but rather uses it to select solutions to train the task-specific model, which makes a better use of the question reconstructor.

4.4 Semantic Parsing

Text2SQL is a popular semantic parsing task. Given a question q and a table header $d = [h_1, ..., h_L]$ where h_l is a multi-token column, a parser is required to parse q into a SQL query z and return the execution results. Under the weakly supervised setting, only the final answer is provided while the SQL query is not. Following Min et al. (2019), Z is approximated as a set of non-nested SQL queries with no more than three conditions:

$$Z = \{z = (z^{sel}, z^{agg}, \{z_k^{cond}\}_{k=1}^{3}) | f(z) = a,$$
$$z^{sel} \in \{h_1, ..., h_L\}, z_k^{cond} \in \{none\} \cup C,$$
$$z^{agg} \in \{none, sum, mean, max, min, count\}\}$$

[1]Our implementation of NeRd has four major differences from that of (Chen et al., 2020). (1) Instead of choosing $\text{BERT}_{\text{large}}$ as encoder, we chose the discriminator of $\text{Electra}_{\text{base}}$ (Clark et al., 2020) which is of a smaller size. (2) We did not use moving averages of trained parameters. (3) We did not use the full public train set for training but used 10% of it for development. (4) For some questions, it is hard to guarantee that a precomputed solution set covers the ground-truth solution. For example, the question *How many touchdowns did Brady throw?* needs counting, but the related mentions are not known. (Chen et al., 2020) partly solved this problem by adding model-predicted solutions (with correct answer) into the initial solution sets as learning proceeds. In this paper, we kept the initial solution sets unchanged during training, so that different QA tasks share the same experimental setting.

where z^{agg} is an aggregating operator and z^{sel} is the operated column (a span of d). $C = \{(h, o, v)\}$ is the set of all possible conditions, where h is a column, $o \in \{=, <, >\}$, and v is a question span.

We experimented on WikiSQL (Zhong et al., 2017) under the weakly supervised setting[2]. We chose SQLova (Hwang et al., 2019) as the task-specific model which is a competitive text2SQL parser on WikiSQL. Hyperparameters were kept the same as in (Hwang et al., 2019). We used the solution sets provided by Min et al. (2019).

Results: All models in Table 5 do not apply execution-guided decoding during inference. Our method achieves new state-of-the-art results under the weakly supervised setting. Though without supervision of ground-truth solutions, our execution accuracy (i.e., accuracy of execution results) on the test set is close to that of the fully supervised SQLova. Notably, GRAPPA focused on representation learning and used a stronger task-specific model while we focus on the learning method and outperform GRAPPA with a weaker model.

5 Ablation Study

5.1 Performance on Test Data with Different Size of Solution Set

Fig 4 shows the performance on test data with different size of solution set[3]. Our method consistently outperforms HardEM-thres and by a large margin when test examples have a large solution set.

5.2 Effect of $|Z|$ at Training

The more complex a question is, the larger the set of possible solutions tends to be, the more likely a model will suffer from the spurious solution problem. We therefore investigated whether our learning method can deal with extremely noisy solution sets. Specifically, we extracted a hard train set from the original train set of WikiSQL. The hard train set consists of 10K training data with the largest Z. The average size of Z on the hard train set is 1,554.6, much larger than that of the original train set (315.4). We then compared models trained on the original train set and the hard train set using different learning methods.

[2]WikiSQL has annotated ground-truth SQL queries. We only used them for evaluation but not for training.

[3]In this experiment, $|Z|$ is only seen as a property of an example. Evaluated solutions are predicted by the task-specific model but not from Z.

Model	Execution Accuracy Dev	Test
Fully-supervised Setting		
SQLova (Hwang et al., 2019)	87.2	86.2
HydraNet (Lyu et al., 2020)	**89.1**	**89.2**
Weakly-supervised Setting		
MeRL (Agarwal et al., 2019)	74.9	74.8
GRAPPA (Yu et al., 2021)	85.9	84.7
MML(Min et al., 2019)	70.6	70.5
HardEM	84.5‡	84.1‡
HardEM-thres	85.2†	84.1‡
Ours	**85.9**	**85.6**

Table 5: Evaluation on WikiSQL. Accuracy that is significantly lower than the highest one is marked with † for p-value < 0.1, and ‡ for p-value < 0.05 (t-test).

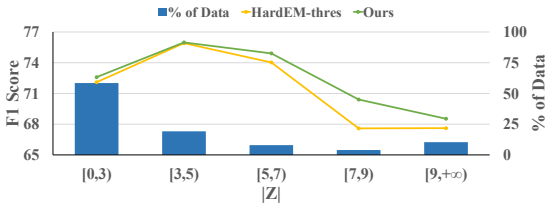

Figure 4: Performance on test examples with different size of Z on DROP.

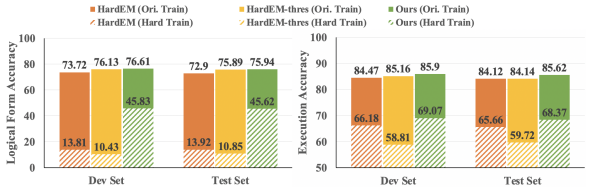

Figure 5: Logical form accuracy (left) and execution accuracy (right) on dev set and test set of WikiSQL. A method marked with *Ori. Train* or *Hard Train* means the evaluated model is trained on the original train set or a hard subset of training data, respectively. The hard train set consists of 10K training data with the largest solution set; the average size of solution set is 1,554.6.

As shown in Fig 5, models trained with our method consistently outperform baselines in terms of logical form accuracy (i.e., accuracy of predicted solutions) and execution accuracy. When using the hard train set, the logical form accuracy of models trained with HardEM or HardEM-thres drop to below 14%. Compared with HardEM, HardEM-thres is better when trained on the original train set but is worse when trained on the hard train set. These indicate that model confidence can be unreliable and thus insufficient to filter out spurious solutions. By contrast, our method explicitly exploits the semantic correlations between a question and a solution, thus much more resistant to spurious solutions.

Training Epochs	2	4	6	8	10
$BART_{base}$ w/ HardEM	65.1	60.8	59.7	58.6	61.0
SQLova w/ HardEM	61.3	62.2	61.8	61.8	61.7
SQLova w/ Ours	**79.7**	**82.8**	**79.8**	**81.2**	**87.4**

Table 6: Accuracy on the SQL selection task. The hard train set was used for training. *$BART_{base}$ w/ HardEM* and *SQLova w/ HardEM* are a $BART_{base}$ parser and SQLova, respectively; both were trained with HardEM. *SQLova w/ Ours* is SQLova trained with the proposed mutual information maximization approach (using $BART_{base}$ question reconstructor).

5.3 Effect of the Question Reconstructor

As we used $BART_{base}$ as the question resconstructor, we investigated how our question reconstructor contributes to performance improvements.

We first investigated whether $BART_{base}$ itself is less affected by the spurious solution problem than the task-specific models. Specifically, we viewed text2SQL as a sequence generation task and fine-tuned a $BART_{base}$ on the hard train set of WikiSQL with HardEM. The input of BART shares the same format as that of SQLova, which is the concatenation of a question and a table header. The output of BART is a SQL query. Without constraints on decoding, BART might not produce valid SQL queries. We therefore evaluated models on a SQL selection task instead: for each question in the development set of WikiSQL, a model picks out the correct SQL from at most 10 candidates by selecting the one with the highest prediction probability. As shown in Table 6, when trained with HardEM, both $BART_{base}$ parser and SQLova perform similarly, and underperform our method by a large margin. This indicates that using $BART_{base}$ as a task-specific model can not avoid the spurious solution problem. It is our mutual information maximization objective that makes a difference.

	DROP				WikiSQL (Hard Train Set)			
	Dev		Test		Dev		Test	
	EM	F1	EM	F1	LF. Acc	Exe. Acc	LF. Acc	Exe. Acc
T-scratch	**61.5**	66.3	69.0	72.4	24.7	67.9	24.9	67.5
T-DAE	**61.5**	66.3	**69.4**	72.7	**49.4**	68.9	**48.5**	**68.4**
$BART_{base}$	**61.5**	**66.4**	69.3	**72.9**	45.8	**69.1**	45.6	**68.4**

Table 7: Results with different question reconstructors. *LF. Acc* and *Exe. Acc* are logical form accuracy and execution accuracy, respectively. *T-scratch* is a Transformer without pre-training. *T-DAE* is a Transformer pre-trained as a denoising auto-encoder of questions.

We further investigated the effect of the choice of question reconstructor. We compared $BART_{base}$ with two alternatives: (1) **T-scratch:** a three-layer Transformer (Vaswani et al., 2017) without pre-training and (2) **T-DAE:** a three-layer Transformer pre-trained as a denoising auto-encoder of questions on the train set; the text infilling pre-training task for BART was used. As shown in Table 7, our method with either of the three question reconstructors outperforms or is at least competitive with baselines, which verifies the effectiveness of our mutual information maximization objective. What's more, using T-DAE is competitive with $BART_{base}$, indicating that our training objective is compatible with other choices of question reconstructor besides BART, and that using a denoising auto-encoder to initialize the question reconstructor may be beneficial to exploit the semantic correlations between a question and its solution.

6 Evaluation of Solution Prediction

As solutions with correct answer can be spurious, we further analyzed the quality of predicted solutions. We randomly sampled 50 test examples from DROP for which our method produced the correct answer, and found that our method also produced the correct solution for 92% of them.

To investigate the effect of different learning methods on models' ability to produce correct solutions, we manually analyzed another 50 test samples for which HardEM, HardEM-thres, and our method produced the correct answer with different solutions. The percentage of samples for which our method produced the correct solution is 58%, much higher than that of HardEM (10%) and HardEM-thres (30%). **For experimental details, please refer to the appendix.**

7 Case Study

Fig 6 compares NeRd predictions on four types of questions from DROP when using different learning methods. An observation is that NeRd using our method shows more comprehensive understanding of questions, e.g., in the *Arithmetic* case, NeRd using our method is aware of the two key elements in the question including the year when *missionaries arrived in Ayutthaya* and the year when *the Seminary of Saint Joseph* was built, while NeRd using HardEM-thres misses the first element. What's more, NeRd using our method is more precise in locating relevant information, e.g., in the first *Sorting* case, NeRd with our method locates the second appearance of *2* whose contextual semantics matches the question, while NeRd using HardEM-thres locates the first appearance of *2* which is irrelevant.

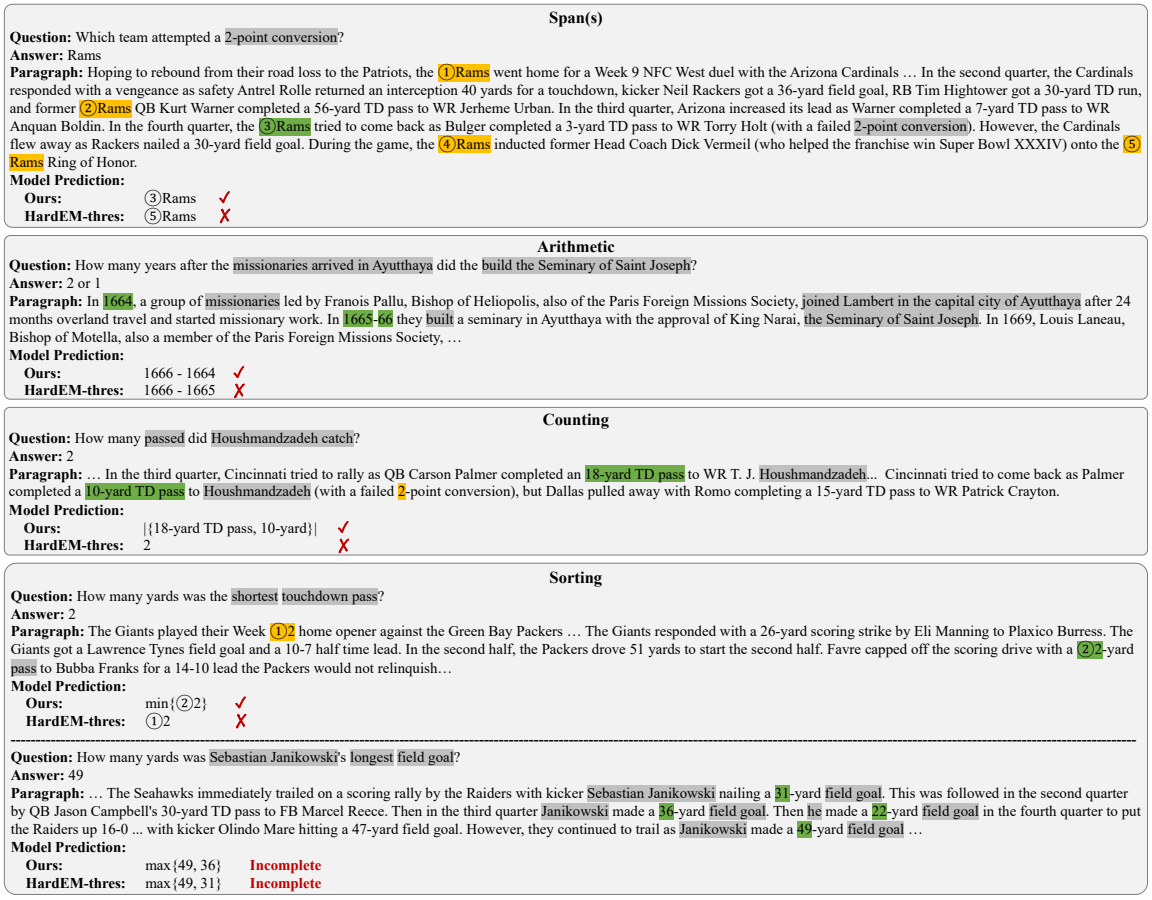

Figure 6: NeRd predictions on four types of questions from DROP when using different learning methods. Spans in dark gray and green denote semantic correlations between a question and its solution, while spans in orange are spurious information and should not be used in a solution.

These two observations can be attributed to our mutual information maximization objective which biases a task-specific model towards those solutions that align well with the questions.

However, we also observed that when there are multiple mentions of relevant information of the same type, NeRd trained with HardEM-thres or our method has difficulty in recalling them all, e.g., in the second *Sorting* case, the correct solution should locate all four mentions of *Sebastian Janikowski's field goals* while NeRd using either method locates only two of them. We conjecture that this is because the solution sets provided by Chen et al. (2020) are noisy. For example, all precomputed solutions of sorting type for numeric answers involve up to two numbers from reference information, which makes it hard for a model to learn to sort more than two numbers.

8 Conclusion

To alleviate the spurious solution problem in weakly supervised QA, we propose to explicitly exploit the semantic correlations between a question and its solution via mutual information maximization. During training, we pair a task-specific model with a question reconstructor which guides the task-specific model to predict solutions that are consistent with the questions. Experiments on four QA datasets demonstrate the effectiveness of our learning method. As shown by automatic and manual analyses, models trained with our method are more resistant to spurious solutions during training, and are more precise in locating information that is relevant to the questions during inference, leading to higher accuracy of both answers and solutions.

9 Acknowledgements

This work was partly supported by the NSFC projects (Key project with No. 61936010 and regular project with No. 61876096). This work was also supported by the Guoqiang Institute of Tsinghua University, with Grant No. 2019GQG1 and 2020GQG0005.

References

Rishabh Agarwal, Chen Liang, Dale Schuurmans, and Mohammad Norouzi. 2019. Learning to generalize from sparse and underspecified rewards. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 130–140. PMLR.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1533–1544. ACL.

Ruisheng Cao, Su Zhu, Chen Liu, Jieyu Li, and Kai Yu. 2019. Semantic parsing with dual learning. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 51–64. Association for Computational Linguistics.

Xinyun Chen, Chen Liang, Adams Wei Yu, Denny Zhou, Dawn Song, and Quoc V. Le. 2020. Neural symbolic reader: Scalable integration of distributed and symbolic representations for reading comprehension. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Jianpeng Cheng and Mirella Lapata. 2018. Weakly-supervised neural semantic parsing with a generative ranker. In *Proceedings of the 22nd Conference on Computational Natural Language Learning, CoNLL 2018, Brussels, Belgium, October 31 - November 1, 2018*, pages 356–367. Association for Computational Linguistics.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 845–855. Association for Computational Linguistics.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. ELECTRA: pre-training text encoders as discriminators rather than generators. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

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

Bhuwan Dhingra, Kathryn Mazaitis, and William W. Cohen. 2017. Quasar: Datasets for question answering by search and reading. *CoRR*, abs/1707.03904.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 2368–2378. Association for Computational Linguistics.

Wonseok Hwang, Jinyeung Yim, Seunghyun Park, and Minjoon Seo. 2019. A comprehensive exploration on wikisql with table-aware word contextualization. *CoRR*, abs/1902.01069.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1601–1611. Association for Computational Linguistics.

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

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Tomás Kociský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Trans. Assoc. Comput. Linguistics*, 6:317–328.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 6086–6096. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation,

and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics.

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

Chen Liang, Mohammad Norouzi, Jonathan Berant, Quoc V. Le, and Ni Lao. 2018. Memory augmented policy optimization for program synthesis and semantic parsing. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 10015–10027.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Qin Lyu, Kaushik Chakrabarti, Shobhit Hathi, Souvik Kundu, Jianwen Zhang, and Zheng Chen. 2020. Hybrid ranking network for text-to-sql. *CoRR*, abs/2008.04759.

Sewon Min, Danqi Chen, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019. A discrete hard EM approach for weakly supervised question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 2851–2864. Association for Computational Linguistics.

Panupong Pasupat and Percy Liang. 2015. Compositional semantic parsing on semi-structured tables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1470–1480. The Association for Computer Linguistics.

Swabha Swayamdipta, Ankur P. Parikh, and Tom Kwiatkowski. 2018. Multi-mention learning for reading comprehension with neural cascades. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Alon Talmor and Jonathan Berant. 2019. Multiqa: An empirical investigation of generalization and transfer in reading comprehension. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 4911–4921. Association for Computational Linguistics.

Duyu Tang, Nan Duan, Tao Qin, and Ming Zhou. 2017. Question answering and question generation as dual tasks. *CoRR*, abs/1706.02027.

Yi Tay, Anh Tuan Luu, Siu Cheung Hui, and Jian Su. 2018. Densely connected attention propagation for reading comprehension. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 4911–4922.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Bailin Wang, Ivan Titov, and Mirella Lapata. 2019. Learning semantic parsers from denotations with latent structured alignments and abstract programs. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 3772–3783. Association for Computational Linguistics.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics.

Hai Ye, Wenjie Li, and Lu Wang. 2019. Jointly learning semantic parser and natural language generator via dual information maximization. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 2090–2101. Association for Computational Linguistics.

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, bailin wang, Yi Chern Tan, Xinyi Yang, Dragomir Radev, richard socher, and Caiming Xiong. 2021. Gra{pp}a: Grammar-augmented pre-training for table semantic parsing. In *International Conference on Learning Representations*.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and

Dragomir R. Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3911–3921. Association for Computational Linguistics.

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

A Implementation Details

A.1 Learning Methods

HardEM: We followed Min et al. (2019) to apply annealing to HardEM on reading comprehension tasks: at the training step t, a model optimizes MML objective with a probability of $\min(t/\tau, 0.8)$ and optimizes HardEM objective otherwise. τ was chosen from $\{10K, 20K, 30K, 40K, 50K\}$ based on model performance on the development set.

HardEM-thres: We set the confidence threshold as $\gamma = 0.5^n$ where n was initialized as follows: we first computed the prediction probability of each solution with a task-specific model, and then set n to a value such that the model was trained on no less than half of training data at the first epoch. We halved γ after each epoch.

VAE(Cheng and Lapata, 2018): A method that views a solution as the latent variable for question generation and adopts the training objective of Variational Auto-Encoder (VAE) to regularize the task-specific model. The overall training objective is given by:

$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \mathcal{L}(\theta, \phi)$$

$$\begin{aligned}\mathcal{L}(\theta, \phi) &= \mathcal{L}^{mle}(\theta) + \lambda \mathcal{L}^{vae}(\theta, \phi) \\ &= \sum_{z \in B} \log P_\theta(z|d, q) + \lambda E_{P_\theta(z|d,q)} \log \frac{P_\phi(q|d, z)}{P_\theta(z|d, q)}\end{aligned}$$

where $\mathcal{L}^{mle}(\theta)$ is the total log likelihood of the set of model-predicted solutions (denoted by B) with correct answer. $\mathcal{L}^{vae}(\theta, \phi)$ is the evidence lower bound of the log likelihood of questions. λ is the coefficient of $\mathcal{L}^{vae}(\theta, \phi)$. The optimization process is divided into three stages: (1) the 1st stage pre-trains a task-specific model θ with HardEM-thres on solution sets[4]; (2) the 2nd stage pairs the task-specific model with our question reconstructor ϕ to optimize $\mathcal{L}(\theta, \phi)$ for one epoch, except that $\mathcal{L}^{vae}(\theta, \phi)$ is used to pre-train ϕ and is kept from back-propagating to θ; (3) the 3rd stage optimizes $\mathcal{L}(\theta, \phi)$ while allowing $\mathcal{L}^{vae}(\theta, \phi)$ to back-propagate to θ. The gradient of $\mathcal{L}^{vae}(\theta, \phi)$ w.r.t. θ is given by:

$$\nabla_\theta \mathcal{L}^{vae}(\theta, \phi) = E_{P_\theta(z|d,q)} R \nabla_\theta \log P_\theta(z|d, q)$$

$$R = \log \frac{P_\phi(q|d, z)}{P_\theta(z|d, q)}$$

where R is the reward function. To stabilize training, we use the average reward of 5 sampled so-

[4] Cheng and Lapata (2018) pre-trained the task-specific model θ by maximizing $\mathcal{L}^{mle}(\theta)$. We enhanced their method by pre-training θ with HardEM-thres.

lutions as a baseline b and re-define the reward function as $R^{'} = R - b$. λ is set to 0.1.

In section 4.3, we report performance of the best model in the 3rd stage. At the 2nd stage, as the task-specific model optimized on both correct solutions and spurious solutions equally, the F1 score dropped from 72.35 to 67.93 at the end of this stage, indicating that correct training solutions is vital for generalization. At the 3rd stage, model learning was further regularized with $\mathcal{L}^{vae}(\theta, \phi)$ which was optimized via reinforcement learning. Despite our efforts to stabilize training, the F1 score still dropped to 36.28 after training for 1,000 steps at the 3rd stage.

A.2 Experimental Settings

For all experiments, we used previously proposed task-specific models and optimized them with their original optimizer. We chose the best task-specific model according to its performance on the development set. As for our learning method, we used $\text{BART}_{\text{base}}$ as the question reconstructor. AdamW optimizer (Loshchilov and Hutter, 2019) was used to update the question reconstructor with learning rate set to 5e-5.

A.2.1 Multi-mention Reading Comprehension

We adopted the reading comprehension model, data preprocessing, and training configurations from Min et al. (2019).

Task-specific model: The model is based on uncased version of $\text{BERT}_{\text{base}}$, which takes as input the concatenation of a question and a paragraph, and outputs the probability distribution of the start and end position of the answer span. To deal with multi-paragraph reading comprehension, it also trains a paragraph selector; during inference, it outputs a span from the paragraph ranked 1st.

Data Preprocessing: Documents are split to segments up to 300 tokens. For Quasar-T, as retrieved sentences are short, we concatenated all sentences into one document in decreasing order of retrieval score (i.e., relevance with the question); for WebQuestions, we concatenated 5 retrieved paragraphs into one document, resulting in 10 reference documents per question.

Training: Batch size is 20. BertAdam optimizer was used to update the reading comprehension model with learning rate set to 5e-5. The number of training epochs is 10.

A.2.2 Discrete Reasoning over Paragraphs

We used NeRd (Chen et al., 2020) for discrete reasoning. The major differences with its original implementation have been discussed in section 4.3.

Task-specific Model: Chen et al. (2020) have designed a domain-specific language for discrete reasoning on DROP. The definitions of solutions for discrete reasoning introduced in section 4.3 are also expressed in this language except that we use different symbols (e.g., the minus sign *"-"* in our definitions has the same meaning as the symbol *"DIFF"* in their paper). NeRd is a Seq2Seq model which tasks as input the concatenation of a question and a paragraph, and generates the solution as a sequence. The answer is obtained by executing the solution.

Data Preprocessing: The input of the task-specific model is truncated to up to 512 words. We used the solution sets provided by Chen et al. (2020), which cover 93.2% of examples in the train set.

Training: Batch size is 32. Adam optimizer (Kingma and Ba, 2015) was used to update NeRd with learning rate set to 5e-5. The number of training epochs is 20.

A.2.3 Semantic Parsing

Following Min et al. (2019), we used SQLova (Hwang et al., 2019) on WikiSQL.

Task-specific Model: SQLova encodes the concatenation of a question and a table header with uncased $\text{BERT}_{\text{base}}$, and outputs a SQL query via slot filling with an NL2SQL (natural language to SQL) layer.

Data Preprocessing: Data preprocessing was kept the same as in (Min et al., 2019). We also used the solution sets provided by Min et al. (2019) which cover 98.8% of examples in the train set.

Training: Following Min et al. (2019), we set the batch size to 10. Following Hwang et al. (2019), Adam optimizer was used to update SQLova with learning rate of $\text{BERT}_{\text{base}}$ and NL2SQL layer set to 1e-5 and 1e-3, respectively. The number of training epochs is 15 and 20 when using the original train set and the hard train set of WikiSQL, respectively.

A.3 Computing Infrastructure

We conducted experiments on 24GB Quadro RTX 6000 GPUs. Most experiments used 1 GPU except that experiments on DROP used 4 GPUs in parallel.

B Details of Ablation Study

B.1 SQL Selection Task

We defined a SQL selection task on the development set of WikiSQL. Specifically, for each question, we randomly sampled $min(10, |Z|)$ solution candidates from the solution set Z without replacement while ensuring the ground-truth solution was one of the candidates. A model was required to pick out the ground-truth solution by selecting the candidate with the highest prediction probability.

In section 5.3, we only show model accuracy in the first 10 training epochs because for *BART$_{base}$ w/ HardEM*, *SQLova w/ HardEM*, and *SQLova w/ Ours*, model confidence (computed as the average log likelihood of selected SQLs) showed a downward trend after the 2^{nd}, 4^{th}, and $\geq 10^{th}$ epoch, respectively.

B.2 Choice of Question Reconstructor

We investigated how the choice of the question reconstructor affects results. One alternative choice is a Transformer pre-trained as a denoising autoencoder of questions on the train set. This question reconstructor is the same as $\text{BART}_{\text{base}}$ except that the number of encoder layers and the number of decoder layers are 3 respectively. We pre-trained the question reconstructor for one epoch to reconstruct original questions from corrupted ones. For 50% of the time, the input question is the original question; otherwise, we followed Lewis et al. (2020) to corrupt the original question by randomly masking a number of text spans with span lengths drawn from a Poisson distribution ($\lambda = 3$). Batch size is 4. AdamW optimizer was used with learning rate set to 5e-5.