A Discrete Hard EM Approach for Weakly Supervised Question Answering

Sewon Min[1], **Danqi Chen**[2,3], **Hannaneh Hajishirzi**[1,4], **Luke Zettlemoyer**[1,3]
[1]University of Washington, Seattle, WA
[2]Princeton University, Princeton, NJ
[3]Facebook AI Research, Seattle, WA
[4]Allen Institute for Artificial Intelligence, Seattle, WA
{sewon,hannaneh,lsz}@cs.washington.edu danqic@cs.princeton.edu

Abstract

Many question answering (QA) tasks only provide weak supervision for how the answer should be computed. For example, TRIVIAQA answers are entities that can be mentioned multiple times in supporting documents, while DROP answers can be computed by deriving many different equations from numbers in the reference text. In this paper, we show it is possible to convert such tasks into discrete latent variable learning problems with a precomputed, task-specific set of possible *solutions* (e.g. different mentions or equations) that contains one correct option. We then develop a hard EM learning scheme that computes gradients relative to the most likely solution at each update. Despite its simplicity, we show that this approach significantly outperforms previous methods on six QA tasks, including absolute gains of 2–10%, and achieves the state-of-the-art on five of them. Using hard updates instead of maximizing marginal likelihood is key to these results as it encourages the model to find the one correct answer, which we show through detailed qualitative analysis.[1]

1 Introduction

A natural setting in many question answering (QA) tasks is to provide weak supervision to determine how the question should be answered given the evidence text. For example, as seen in Figure 1, TRIVIAQA answers are entities that can be mentioned multiple times in supporting documents, while DROP answers can be computed by deriving many different equations from numbers in the reference text. Such weak supervision is attractive because it is relatively easy to gather, allowing for large datasets, but complicates learning because there are many different spurious ways to derive the correct answer. It is natural to model such ambiguities with a latent variable during learning, but most prior work on reading comprehension has rather focused on the model architecture and used heuristics to map the weak signal to full supervision (e.g. by selecting the first answer span in TRIVIAQA (Joshi et al., 2017; Tay et al., 2018; Talmor and Berant, 2019)). Some models are trained with maximum marginal likelihood (MML) (Kadlec et al., 2016; Swayamdipta et al., 2018; Clark and Gardner, 2018; Lee et al., 2019), but it is unclear if it gives a meaningful improvement over the heuristics.

In this paper, we show it is possible to formulate a wide range of weakly supervised QA tasks as discrete latent-variable learning problems. First, we define a *solution* to be a particular derivation of a model to predict the answer (e.g. a span in

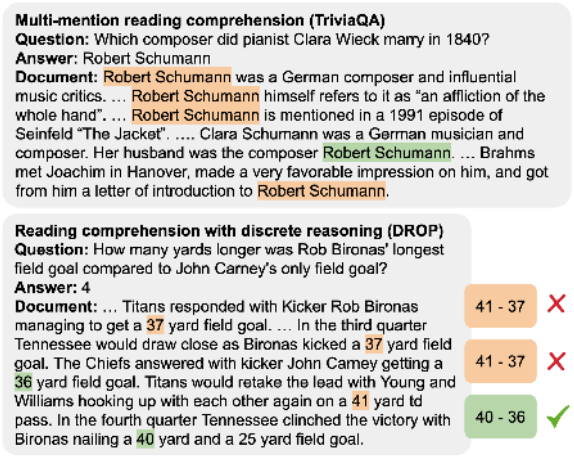


Figure 1: Examples from two different question answering tasks. **(Top) Multi-mention reading comprehension.** The answer text is mentioned five times in the given document, however, only the fourth span actually answers the question. **(Bottom) Reading comprehension with discrete reasoning.** There are many potential equations which execute the answer ('4'), but only one of them is the correct equation ('40-36') and the others are false positives.

[1]Our code is publicly available at https://github.com/shmsw25/qa-hard-em.

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 2851–2864, Hong Kong, China, November 3–7, 2019.

Task & Dataset	# Examples			$\lvert Z \rvert$	
	Train	Dev	Test	Avg	Median
1. Multi-mention reading comprehension					
TRIVIAQA (Joshi et al., 2017)	61,888	7,993	7,701	2.7	2
NARRATIVEQA (Kočiskỳ et al., 2018)	32,747	3,461	10,557	4.3	5
TRIVIAQA-OPEN (Joshi et al., 2017)	78,785	8,837	11,313	6.7	4
NATURALQUESTIONS-OPEN (Kwiatkowski et al., 2019)	79,168	8,757	3,610	1.8	1
2. Reading comprehension with discrete reasoning					
DROP_{num} (Dua et al., 2019)	46,973	5,850	-	8.2	3
3. Semantic Parsing					
WIKISQL (Zhong et al., 2017)	56,355	8,421	15,878	346.1	5

Table 1: Six QA datasets in three different categories used in this paper (detailed in Section 5) along with the size of each dataset. An average and median of the size of precomputed solution sets (denoted by Z) are also reported. Details on how to obtain Z are given in Section 4.

the document or an equation to compute the answer). We demonstrate that for many recently introduced tasks, which we group into three categories as given in Table 1, it is relatively easy to precompute a discrete, task-specific set of possible solutions that contains the correct solution along with a modest number of spurious options. The learning challenge is then to determine which solution in the set is the correct one, while estimating a complete QA model.

We model the set of possible solutions as a discrete latent variable, and develop a learning strategy that uses hard-EM-style parameter updates. This algorithm repeatedly (i) predicts the most likely solution according to the current model from the precomputed set, and (ii) updates the model parameters to further encourage its own prediction. Intuitively, these hard updates more strongly enforce our prior beliefs that there is a single correct solution. This method can be applied to any problem that fits our weak supervision assumptions and can be used with any model architecture.

We experiment on six different datasets (Table 1) using strong task-specific model architectures (Devlin et al., 2019; Dua et al., 2019; Hwang et al., 2019). Our learning approach significantly outperforms previous methods which use heuristic supervision and MML updates, including absolute gains of 2–10%, and achives the state-of-the-art on five datasets. It outperforms recent state-of-the-art reward-based semantic parsing algorithms (Liang et al., 2018; Agarwal et al., 2019) by 13% absolute percentage on WIKISQL, strongly suggesting that having a small precomputed set of possible solutions is a key ingredient. Finally, we present a detailed analysis showing that, in practice, the introduction of hard updates encourages models to assign much higher probability to the correct solution.

2 Related Work

Reading Comprehension. Large-scale reading comprehension (RC) tasks that provide full supervision for answer spans (Rajpurkar et al., 2016) have seen significant progress recently (Seo et al., 2017; Xiong et al., 2018; Yu et al., 2018a; Devlin et al., 2019). More recently, the community has moved towards more challenging tasks such as distantly supervised RC (Joshi et al., 2017), RC with free-form human generated answers (Kočiskỳ et al., 2018) and RC requiring discrete or multi-hop reasoning (Dua et al., 2019; Yang et al., 2018). These tasks introduce new learning challenges since the gold solution that is required to answer the question (e.g. a span or an equation) is not given.

Nevertheless, not much work has been done for this particular learning challenge. Most work on RC focuses on the model architecture and simply chooses the first span or a random span from the document (Joshi et al., 2017; Tay et al., 2018; Talmor and Berant, 2019), rather than modeling this uncertainty as a latent choice. Others maximize the sum of the likelihood of multiple spans (Kadlec et al., 2016; Swayamdipta et al., 2018; Clark and Gardner, 2018; Lee et al., 2019), but it is unclear if it gives a meaningful improvement. In this paper, we highlight the learning challenge and show that our learning method, independent of the model architecture, can give a significant gain. Specifically, we assume that one of

mentions are related to the question and others are false positives because (i) this happens for most cases, as the first example in Table 2, and (ii) even in the case where multiple mentions contribute to the answer, there is often a single span which fits the question the best.

Semantic Parsing. Latent-variable learning has been extensively studied in the literature of semantic parsing (Zettlemoyer and Collins, 2005; Clarke et al., 2010; Liang et al., 2013; Berant et al., 2013; Artzi and Zettlemoyer, 2013). For example, a question and an answer pair (x, y) is given but the logical form that is used to compute the answer is not. Two common learning paradigms are maximum marginal likelihood (MML) and reward-based methods. In MML, the objective maximizes $\sum_{z \in \hat{Z}} \mathbb{P}(z|x)$, where $\hat{Z}$ is an approximation of a set of logical forms executing y (Liang et al., 2013; Berant et al., 2013; Krishnamurthy et al., 2017). In reward-based methods, a reward function is defined as a prior, and the model parameters are updated with respect to it (Iyyer et al., 2017; Liang et al., 2017, 2018). Since it is computationally expensive to obtain a precomputed set in semantic parsing, these methods typically recompute the set of logical forms with respect to the beam at every parameter update. In contrast, our learning method targets tasks that a set of solutions can be precomputed, which include many recent QA tasks such as reading comprehension, open-domain QA and a recent SQL-based semantic parsing task (Zhong et al., 2017).

3 Method

In this section, we first formally define our general setup, which we will instantiate for specific tasks in Section 4 and then we describe our learning approach.

3.1 Setup

Let x be the input of a QA system (e.g. a question and a document) and y be the answer text (e.g. 'Robert Schumann' or '4'). We define a *solution* as a particular derivation that a model is supposed to produce for the answer prediction (e.g. a span in the document or an equation to compute the answer, see Table 2). Let f denote a task-specific, deterministic function which maps a solution to the textual form of the answer (e.g. by simply returning the string associated with a particular selected mention or solving an equation to get the final number, see Table 2). Our goal is to learn a model (with parameters θ) which takes an input x and outputs a solution z such that $f(z) = y$.

In a fully supervised scenario, a true solution $\bar{z}$ is given, and θ is estimated based on a collection of $(x, \bar{z})$ pairs. In this work, we focus on a weakly supervised scenario in which $\bar{z}$ is not given. For this setup, we define Z_{tot} as a finite set of all the possible solutions. In the case that the search space is very large or infinite, we can usually approximate Z_{tot} with a high coverage in practice. Then, we obtain $Z = \{z \in Z_{\text{tot}} : f(z) = y\}$ by enumerating all $z \in Z_{\text{tot}}$. This results a set of all the possible solutions that lead to the correct answer. We assume it contains one solution that we want to learn to produce, and potentially many other spurious ones. In practice, Z is defined in a task-specific manner, as we will see in Section 4.

At inference time, the model produces a solution $\tilde{z} \in Z_{\text{tot}}$ from an input x with respect to θ and predicts the final answer as $f(\tilde{z})$. Note that we cannot compute Z at inference time because the groundtruth y is not given.[2]

3.2 Learning Method

In a fully-supervised setting where $\bar{z}$ is given, we can learn θ by optimizing the negative log likelihood of $\bar{z}$ given the input x with respect to θ.

$$J_{\text{Sup}}(\theta|x, \bar{z}) = -\log \mathbb{P}(\bar{z}|x; \theta)$$

In our setup, the model has access to x and $Z = \{z_1, z_2, \ldots, z_n\}$. In this scenario, the selection of the best solution in Z can be modeled as a latent variable. We can compute the maximum marginal likelihood (MML) estimate, which marginalizes the likelihood of each $z_i \in Z$ given the input x with respect to θ. Formally,

$$\begin{aligned}\mathbb{P}(y|x; \theta) &= \sum_{z_i \in Z_{\text{tot}}} \mathbb{P}(y|z_i)\mathbb{P}(z_i|x; \theta) \\ &= \sum_{z_i \in Z} \mathbb{P}(z_i|x; \theta)\end{aligned}$$

is used to compute the objective as follows:

$$J_{\text{MML}}(\theta|x, Z) = -\log \sum_{z_i \in Z} \mathbb{P}(z_i|x; \theta)$$

However, there are two major problems in the MML objective in our settings. First, MML can be

[2] This is a main difference from multi-instance learning (MIL) (Zhou et al., 2009), since a bag of input instances is given at inference time in MIL.

maximized by assigning high probabilities to any subset of $z_i \in Z$; whereas in our problems, instances in Z other than one correct z are spurious solutions which the model should ideally assign very low probability. Second, in MML we optimize the sum over probabilities of Z during training but typically predict the maximum probability solution during inference, creating a discrepancy between training and testing.

We introduce a learning strategy with a hard-EM approach. First, the model computes the likelihood of each z_i given the input x with respect to θ, $\mathbb{P}(z_i|x;\theta)$, and picks one of Z with the largest likelihood:

$$\tilde{z} = \text{argmax}_{z_i \in Z} \mathbb{P}(z_i|x;\theta)$$

Then, the model optimizes on a standard negative log likelihood objective, assuming $\tilde{z}$ is a true solution. The objective can be re-written as follows:

$$\begin{aligned} J_{\text{Hard}}(\theta|x,Z) &= -\log \mathbb{P}(\tilde{z}|x;\theta) \\ &= -\log \max_{z_i \in Z} \mathbb{P}(z_i|x;\theta) \\ &= -\max_{z_i \in Z} \log \mathbb{P}(z_i|x;\theta) \\ &= \min_{z_i \in Z} J_{\text{Sup}}(\theta|x,z_i) \end{aligned}$$

This loss can be seen as a variant of MML, where the sum is replaced with a max.

4 Task Setup

We apply our approach to three different types of QA tasks: multi-mention reading comprehension, RC with discrete reasoning and a semantic parsing task. In this section, we describe each task in detail: how we define a solution z and pre-compute a set Z based on input x and answer y. The statistics of $|Z|$ and examples on each task are shown in Table 1 and Table 2 respectively.

4.1 Multi-Mention Reading Comprehension

Multi-mention reading comprehension naturally occurs in several QA tasks such as (i) distantly-supervised reading comprehension where a question and answer are collected first before the evidence document is gathered (e.g. TRIVIAQA), (ii) abstractive reading comprehension which requires a free-form text to answer the question (e.g. NARRATIVEQA), and (iii) open-domain QA where only question-answer pairs are provided.

Given a question $Q = [q_1, \ldots, q_l]$ and a document $D = [d_1, \ldots, d_L]$, where q_i and d_j denote the tokens in the question and document, the output y is an answer text, which is usually mentioned multiple times in the document.

Previous work has dealt with this setting by detecting spans in the document through text matching (Joshi et al., 2017; Clark and Gardner, 2018). Following previous approaches, we define a solution z as a span in the document. We obtain a set of possible solutions $Z = \{z_1, \ldots, z_n\}$ by finding exact match or similar mentions of y, where $z_i = (s_i, e_i)$ is a span of text with start and end token indices s_i and e_i. Specifically,

$$\begin{aligned} g_{\text{max}} &= \max_{1 \le s_i \le e_i \le L} g([d_{s_i}, \ldots, d_{e_i}], y) \\ Z &= \{z_i = (s_i, e_i) \text{ s.t. } g(s_i, e_i) = g_{\text{max}}\}, \end{aligned}$$

where g is a string matching function. If the answer is guaranteed to be a span in the document D, g is a binary function which returns 1 if two strings are the same, and 0 otherwise. If the answer is free-form text, we choose g as the ROUGE-L metric (Lin, 2004).

This complicates the learning because the given document contains many spans matching to the text, while most of them are not related to answering the question. As an example shown in Table 2, only the fourth span out of six is relevant to the question.

4.2 Reading Comprehension with Discrete Reasoning

Some reading comprehension tasks require reasoning in several discrete steps by finding clues from the document and aggregating them. One such example is mathematical reasoning, where the model must pick numbers from a document and compute an answer through arithmetic operations (Dua et al., 2019).

In this task, the input is also a question Q and a document D, and the output y is given as a numeric value. We define a solution z to be an executable arithmetic equation. Since there is an infinite set of potential equations, we approximate Z_{tot} as a finite set of arithmetic equations with two numeric values and one operation, following Dua et al. (2019).[3] Specifically,

$$\begin{aligned} Z_{\text{tot}} = \big\{ & z_i = (o_1, n_1, o_2, n_2) \text{ s.t.} \\ & o_1, o_2 \in \{+, -, \%\}, \\ & n_1, n_2 \in N_D \cup N_Q \cup S \big\}, \end{aligned}$$

[3] This approximation covers 93% of the examples in the development set.

1. Multi-Mention Reading Comprehension (TRIVIAQA, NARRATIVEQA, TRIVIAQA-OPEN & NATURALQUESTIONS-OPEN)

Question: Which composer did pianist Clara Wieck marry in 1840?
Document: Robert Schumann was a German composer and influential music critic. He is widely regarded as one of the greatest composers of the Romantic era. (...) Robert Schumann himself refers to it as "an affliction of the whole hand". (...) Robert Schumann is mentioned in a 1991 episode of Seinfeld "The Jacket". (...) Clara Schumann was a German musician and composer, considered one of the most distinguished pianists of the Romantic era. Her husband was the composer Robert Schumann . <Childhood> (...) At the age of eight, the young Clara Wieck performed at the Leipzig home of Dr. Ernst Carus. There she met another gifted young pianist who had been invited to the musical evening, named Robert Schumann , who was nine years older. Schumann admired Clara' s playing so much that he asked permission from his mother to discontinue his law studies. (...) In the spring of 1853, the then unknown 20-year-old Brahms met Joachim in Hanover, made a very favorable impression on him, and got from him a letter of introduction to Robert Schumann .
Answer (y): Robert Schumann
f: Text match
Z_{tot}: All spans in the document
Z: Spans which match 'Robert schumann' (red text)

2. Reading Comprehension with Discrete Reasoning (DROP_{num})

Question: How many yards longer was Rob Bironas' longest field goal compared to John Carney's only field goal?
Document: (...) The Chiefs tied the game with QB Brodie Croyle completing a `10` yard td pass to WR Samie Parker. Afterwards the Titans responded with Kicker Rob Bironas managing to get a `37` yard field goal. Kansas city would take the lead prior to halftime with croyle completing a `9` yard td pass to FB Kris Wilson. In the third quarter Tennessee would draw close as Bironas kicked a `37` yard field goal. The Chiefs answered with kicker John Carney getting a `36` yard field goal. Afterwards the Titans would retake the lead with Young and Williams hooking up with each other again on a `41` yard td pass. (...) Tennessee clinched the victory with Bironas nailing a `40` yard and a `25` yard field goal. With the win the Titans kept their playoff hopes alive at `8 6` .
Answer (y): 4
f: Equation executor
Z_{tot}: Equations with two numeric values and one arithmetic operation
Z: { 41-37, 40-36, 10-6, ... }

3. SQL Generation (WIKISQL)

Question: What player played guard for Toronto in 1996-1997?
Table Header: `player, year, position, ...`
Answer (y): John Long
f: SQL executor
Z_{tot}: Non-nested SQL queries with up to 3 conditions
Z:
```
Select player where position=guard and year in toronto=1996-97
Select max(player) where position=guard and year in toronto=1996-97
Select min(player) where position=guard
Select min(player) where year in toronto=1996-97
Select min(player) where position=guard and year in toronto=1996-97
```

Table 2: Examples of the input, answer text (y), f, Z_{tot} and Z. First, in multi-mention reading comprehension, the answer text 'Robert Schumann' is mentioned six times but only the fourth span is related to the question. Second, in reading comprehension with discrete reasoning, many equations yield to the answer 4, but only '40-37' answers the question. Lastly, in SQL query generation, five SQL queries lead to the answer but only the first one is the correct query. See Section 4 for more details.

where N_D and N_Q are all numeric values appearing in D and Q, respectively, and S are a set of predefined special numbers. Then

$$Z = \{z_i \in Z_{\text{tot}} \text{ s.t. } f(z_i) = y\}$$

where f is an execution function of equations.

Figure 1 shows an example Z given a question and a document. We can see that one equation is correct, while the others are false positives which coincidentally lead to the correct answer.

4.3 SQL Query Generation

To evaluate if our training strategy generalizes to other weak supervision problems, we also study a semantic parsing task where a question and an answer are given but the logical form to execute the answer is not. In particular, we consider a task of answering questions about a given table by generating SQL queries.

The input is a question $Q = [q_1, \ldots, q_l]$ and a table header $H = [h_1, \ldots, h_{n_L}]$, where q_i is a token, h_i is a multi-token title of each column, and n_L is the number of headers. The supervision is given as the SQL query result y, which is always a text string.

We define a solution to be an SQL query. Since the set of potential queries is infinite, we approximate Z_{tot} as a set of non-nested SQL queries with at most three conditions.[4] Specifically, given A as a set of aggregating operators $\{\text{sum}, \text{mean}, \text{max}, \text{min}, \text{count}\}$ and C as a set of possible conditions $\{(h, o, t) \text{ s.t. } h \in [1, n_L], o \in \{=, <, >\}, t \in \text{spans in } Q\}$, we define Z_{tot}:

$$\begin{aligned} Z_{\text{tot}} = \{z_i &= (z_i^{\text{sel}}, z_i^{\text{agg}}, \{z_{i,j}^{\text{cond}}\}_{j=1}^{3}\} \text{ s.t.} \\ z_i^{\text{sel}} &\in [1, n_L] \\ z_i^{\text{agg}} &\in \{\text{none}\} \cup A \\ z_{i,j}^{\text{cond}} &\in \{\text{none}\} \cup C \text{ for } j \in [1, 3]\}, \end{aligned}$$

then,

$$Z = \{z_i \in Z_{\text{tot}} \text{ s.t. } f(z_i) = y\},$$

[4] This approximation covers 99% of the examples in the development set.

where f is an SQL executor. The third example in Table 2 shows Z may contain many spurious SQL querie, e.g. the third query in Z coincidentally executes the answer because 'John Long' is ranked first among all the guards in alphabetical order.

5 Experiments

We experiment on a range of question answering tasks with varied model architectures to demonstrate the effectiveness of our approach. Built on top of strong base models, our learning method is able to achieve state-of-the-art on NARRATIVEQA, TRIVIAQA-OPEN, NATURALQUESTIONS-OPEN, DROP_{num} and WIKISQL.

5.1 Multi-mention Reading Comprehension

We experiment on two reading comprehension datasets and two open-domain QA datasets. For reading comprehension, we evaluate on TRIVIAQA (Wikipedia) (Joshi et al., 2017) and NARRATIVEQA (summary) (Kočiskỳ et al., 2018).

For open-domain QA, we follow the settings in Lee et al. (2019) and use the QA pairs from TRIVIAQA-unfiltered (Joshi et al., 2017) and NATURAL QUESTIONS (Kwiatkowski et al., 2019) with short answers and discard the given evidence documents. We refer to these two datasets as TRIVIAQA-OPEN and NATURALQUESTIONS-OPEN.[5]

We experiment with three learning methods as follows.

- First Only: $J(\theta) = -\log\mathbb{P}(z_1|x;\theta)$, where z_1 appears first in the given document among all $z_i \in Z$.
- MML: $J(\theta) = -\log\Sigma_{i=1}^{n}\mathbb{P}(z_i|x;\theta)$.
- Ours: $J(\theta) = -\log\max_{1\le i\le n}\mathbb{P}(z_i|x;\theta)$.

$\mathbb{P}(z_i|Q,D)$ can be obtained by any model which outputs the start and end positions of the input document. In this work, we use a modified version of BERT (Devlin et al., 2019) for multi-paragraph reading comprehension (Min et al., 2019).

Training details. We use uncased version of $\text{BERT}_{\text{base}}$. For all datasets, we split documents into a set of segments up to 300 tokens because BERT limits the size of the input. We use batch size of 20 for two reading comprehension tasks and 192 for two open-domain QA tasks. Following Clark and Gardner (2018), we filter a subset of segments in TRIVIAQA through TF-IDF similarity between a segment and a question to maintain a reasonable length. For open-domain QA tasks, we retrieve 50 Wikipedia articles through TF-IDF (Chen et al., 2017) and further run BM25 (Robertson et al., 2009) to retrieve 20 (for train) or 80 (for development and test) paragraphs. We try 10, 20, 40 and 80 paragraphs on the development set to choose the number of paragraphs to use on the test set.

To avoid local optima, we perform annealing: at training step t, the model optimizes on MML objective with a probability of $\min(t/\tau, 1)$ and otherwise use our objective, where τ is a hyperparameter. We observe that the performance is improved by annealing while not being overly sensitive to the hyperparameter τ. We include full hyperparameters and detailed ablations in Appendix B.

Results. Table 3 compares the results of baselines, our method and the state-of-the-art on four datasets.[6] First of all, we observe that First-Only is a strong baseline across all the datasets. We hypothesize that this is due to the bias in the dataset that answers are likely to appear earlier in the paragraph. Second, while MML achieves comparable result to the First-Only baseline, our learning method outperforms others by 2+ F1/ROUGE-L/EM consistently on all datasets. Lastly, our method achieves the new state-of-the-art on NARRATIVEQA, TRIVIAQA-OPEN and NATURALQUESTIONS-OPEN, and is comparable to the state-of-the-art on TRIVIAQA, despite our aggressive truncation of documents.

5.2 Reading Comprehension with Discrete Reasoning

We experiment on a subset of DROP (Dua et al., 2019) with numeric answers (67% of the entire dataset) focusing on mathematical reasoning. We refer to this subset as DROP_{num}. The current state-of-the-art model is an augmented version of QANET (Yu et al., 2018a) which selects two numeric values from the document or the question and performs addition or subtraction to get the answer. The equation to derive the answer is not

[5] Datasets and their split can be downloaded from https://bit.ly/2HK1Fqn.

[6] For NARRATIVEQA, we compare with models trained on NARRATIVEQA only. For open-domain QA, we only compare with models using pipeline approach.

	TriviaQA (F1)		NarrativeQA (ROUGE-L)		TriviaQA -open (EM)		NaturalQ -open (EM)		DROP_{num} w/ BERT (EM)	DROP_{num} w/ QANet (EM)
	dev	test	dev	test	dev	test	dev	test	dev	dev
First Only	64.4	64.9	55.3	57.4	48.6	48.1	23.6	23.6	42.9	36.1
MML	64.8	65.5	55.8	56.1	47.0	47.4	26.6	25.8	39.7	43.8
Ours	66.9	67.1	**58.1**	**58.8**	**50.7**	**50.9**	**28.8**	**28.1**	**52.8**	**45.0**
SOTA	-	**71.4**	-	54.7	47.2	47.1	24.8	26.5	43.8	

Table 3: **Results on multi-mention reading comprehension & discrete reasoning tasks.** We report performance on five datasets with different base models. Note that we are not able to obtain the test result on the subset DROP_{num}. Previous state-of-the-art are from Wang et al. (2018), Nishida et al. (2019), Lee et al. (2019), Lee et al. (2019) and Dua et al. (2019), respectively. Our training method consistently outperforms the First-Only and MML by a large margin in all the scenarios.

Model	Accuracy dev	Accuracy test
Weakly-supervised setting		
REINFORCE (Williams, 1992)	< 10	
Iterative ML (Liang et al., 2017)	70.1	
Hard EM (Liang et al., 2018)	70.2	
Beam-based MML (Liang et al., 2018)	70.7	
MAPO (Liang et al., 2018)	71.8	72.4
MAPOX (Agarwal et al., 2019)	74.5	74.2
MAPOX+MeRL (Agarwal et al., 2019)	74.9	74.8
MML	70.6	70.5
Ours	**84.4**	**83.9**
Fully-supervised setting		
SQLNet (Xu et al., 2018)	69.8	68.0
TypeSQL (Yu et al., 2018b)	74.5	73.5
Coarse2Fine (Dong and Lapata, 2018)	79.0	78.5
SQLova (Hwang et al., 2019)	87.2	86.2
X-SQL (He et al., 2019)	**89.5**	**88.7**

Table 4: **Results on WikiSQL.** We compare accuracy with weakly-supervised or fully-supervised settings. Our method outperforms previous weakly-supervised methods and most of published fully-supervised methods.

given and Dua et al. (2019) adopted the MML objective.

$\mathbb{P}(z_i|Q, D)$ can take as any model which generates equations based on the question and document. Inspired by Dua et al. (2019), we take a sequence tagging approach on top of two competitive models: (i) augmented QANet, the same model as Dua et al. (2019) but only supporting addition, subtraction and counting, and (ii) augmented BERT, which supports addition, subtraction and percentiles.[7]

Training details. We truncate the document to be up to 400 words. We use the batch size of 14 and 10 for QANet and BERT, respectively.

Results. Table 3 shows the results on DROP_{num}. Our training strategy outperforms the First-Only baseline and MML by a large margin, consistently across two base models. In particular, with BERT, we achieve an absolute gain of 10%.

5.3 SQL Query Generation

Finally, we experiment on the weakly-supervised setting of WikiSQL (Zhong et al., 2017), in which only the question & answer pair is used and the SQL query (z) is treated as a latent variable.

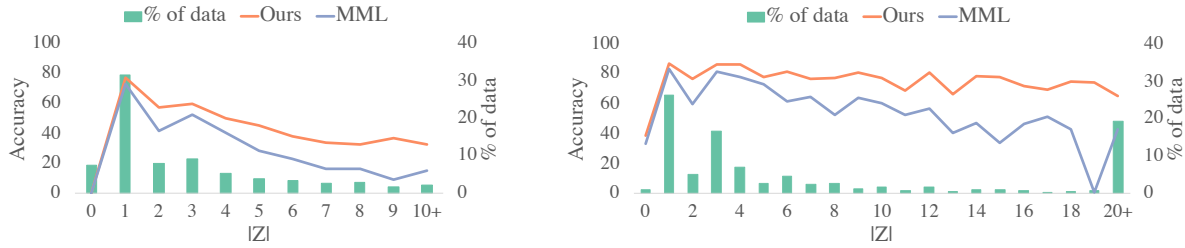
$\mathbb{P}(z_i|Q, H)$ can be computed by any query generation or semantic parsing models. We choose SQLova (Hwang et al., 2019), a competitive model on WikiSQL (designed for fully supervised setting), as our base model. We modify the model to incorporate either the MML objective or our hard-EM learning approach for the weakly-supervised setting.

We compare with both traditional and recently-developed reward-based algorithms for weak supervision, including beam-based MML (MML which keeps a beam during training), conventional hard EM[8], REINFORCE (Williams, 1992), iterative ML (Liang et al., 2017; Abolafia et al., 2018) and a family of MAPO (Memory-augmented policy optimization) (Liang et al., 2018; Agarwal et al., 2019). For a fair comparison, we only consider single models without execution-guided decoding.

Training details. We adopt the same set of hyperparameters as in Hwang et al. (2019), except that we change the batch size to 10 and truncate the input to be up to 180 words for memory efficiency.

[7] As we use a different set of operations for the two models, they are not directly comparable. Details of the model architecture are shown in Appendix A.

[8] This method differs from ours in that it does not have a precomputed set, and uses a beam of candidate predictions to execute for each update.



(a) DROP_{num} (b) WikiSQL

Figure 2: **Varying the size of solution set ($|Z|$) at test time.** We compare the model trained on MML objective (blue) and our training strategy (orange). Our approach consistently outperforms MML on DROP_{num} and WIKISQL, especially when $|Z|$ is large.

Group	Avg $\lvert Z\rvert$	Median $\lvert Z\rvert$	# train
3	3.0	3	10k
10	10.2	9	10k
30	30.0	22	10k
100	100.6	42	10k
300	300.0	66	10k

Figure 3: **Varying the size of solution set ($|Z|$) at training.** (Left) Subsets of the train set on WIKISQL varying in the size of solution set ($|Z|$). All subsets contain 10k training examples (total in the original train set is 55k). All subsets are evaluated on the same, original development set for a fair comparison. (Right) Performance across subsets of the training set with varying $|Z|$. Our method achieves substantial gains over MML.

Results. Table 4 shows that our training method significantly outperforms all the weakly-supervised learning algorithms, including 10% gain over the previous state of the art. These results indicate that precomputing a solution set and training a model through hard updates play a significant role to the performance. Given that our method does not require SQL executions at training time (unlike MAPO), it provides a simpler, more effective and time-efficient strategy. Comparing to previous models with full supervision, our results are still on par and outperform most of the published results.

6 Analysis

In this section, we will conduct thorough analyses and ablation studies, to better understand how our model learns to find a solution from a precomputed set of possible solutions. We also provide more examples and analyses in Appendix C.

Varying the size of solution set at inference time. Figure 2 shows a breakdown of the model accuracy with respect to the size of a solution set ($|Z|$) at test time. We observe that the model with our training method outperforms the model with MML objective consistently across different values of $|Z|$. The gap between MML and our method is marginal when $|Z| = 0$ or 1, and gradually increases as $|Z|$ grows.

Varying the size of solution set at training. To see how our learning method works with respect to the size of a solution set ($|Z|$) of the training data, particularly with large $|Z|$, we take 5 subsets of the training set on WIKISQL with $|Z| = 3, 10, 30, 100, 300$. We train a model with those subsets and evaluate it on the original development set, both with our training method and MML objective. Figure 3 shows statistics of each subset and results. We observe that (i) our learning method outperforms MML consistently over different values of $|Z|$, and (ii) the gain is particularly large when $|Z| > 3$.

Model predictions over training. We analyze the top 1 prediction and the likelihood of $z \in Z$ assigned by the model on DROP_{num} with different number of training iterations (steps from 1k to 32k). Table 5 shows one example on DROP_{num} with the answer text '4', along with the model's top 1 prediction and a subset of Z. We observe that

Q: How many yards longer was Rob Bironas' longest field goal compared to John Carney's only field goal? (**Answer:** 4)
P: ... The Titans responded with Kicker Rob Bironas managing to get a 37 yard field goal. ...Tennessee would draw close as Bironas kicked a 37 yard field goal. The Chiefs answered with kicker John Carney getting a 36 yard field goal. The Titans would retake the lead with Young and Williams hooking up with each other again on a 41 yard td pass. ...Tennessee clinched the victory with Bironas nailing a 40 yard and a 25 yard field goal.

t	Pred	Z (ordered by $\mathbb{P}(z\|x;\theta_t)$)			
1k	10-9	10-6	41-37	40-36	41-37‡
2k	37-36	40-36	41-37	41-37‡	10-6
4k	40-36	40-36	41-37‡	41-37	10-6
8k	40-36	40-36	41-37‡	41-37	10-6
16k	37-36	40-36	41-37	41-37‡	10-6
32k	40-36	40-36	41-37	41-37‡	10-6

Table 5: An example from DROP_{num} (same as Figure 1 and Table 2), with its answer text '4' and a subset of the solution set (Z), containing two of '41-38' (which '41' come from different mentions; one denoted by ‡ for distinction), '40-36' and '10-4'. For each training step t, the top 1 prediction and Z ordered by $P(z|x;\theta_t)$, a probability of $z \in Z$ with respect to the model at t through training procedure are shown. Note that at inference time Z is not given, so top 1 prediction is not necessarily an element of Z.

the model first begins by assigning a small, uniform probability distribution to Z, but gradually learns to favor the true solution. The model sometimes gives the wrong prediction—for example, at $t = 16$k, and changes its prediction from the true solution to the wrong solution, '37-36'—but again changes its prediction to be a true solution afterward. In addition, its intermediate wrong solution, '37-36' indicates the model was confused with distinguishing the longest field goal of Rob Bironas (40 vs. 37), which is an understandable mistake.

We also compare the predictions from the model with our method to those from the model with MML, which is shown in Appendix C.

Quality of the predicted solution. We analyze if the model outputs the correct solution, since the solution executing the correct answer could be spurious. First, on NARRATIVEQA and DROP_{num}, we manually analyze 50 samples from the development set and find that 98% and 92% of correct cases produce the correct solution respectively. Next, on WIKISQL, we compare the predictions from the model to the annotated SQL queries on the development set. This is possible because gold SQL queries are available in the dataset for the full supervision. Out of 8,421 examples, 7,110 predictions execute the correct answers. Among those, 88.5% of the predictions are exactly same as the annotated queries. Others are the cases where (i) both queries are correct, (ii) the model prediction is correct but the annotated query is incorrect, and (iii) the annotated query is correct and the model prediction is spurious. We show a full analysis in Appendix C.

Robustness to the noise in $|Z|$. Sometimes noise arises during the construction of $|Z|$, such as $|Z|$ constructed based on ROUGE-L for NARRATIVEQA. To explore the effect of noise in Z, we experiment with more noisy solution set by picking all the spans with scores that is equal to or larger than the 5th highest. The new construction method increases $|Z|$ from 4.3 to 7.1 on NARRATIVEQA. The result by MML objective drops significantly (56.07→ 51.14) while the result by ours drops marginally (58.77→ 57.97), suggesting that MML suffers more with a noisier Z while ours is more robust.

7 Conclusion

In this paper, we demonstrated that, for many QA tasks which only provide the answer text as supervision, it is possible to precompute a discrete set of possible solutions that contains one correct option. Then, we introduced a discrete latent variable learning algorithm which iterates a procedure of predicting the most likely solution in the precomputed set and further increasing the likelihood of that solution. We showed that this approach significantly outperforms previous approaches on six QA tasks including reading comprehension, open-domain QA, discrete reasoning task and semantic parsing, achieving absolute gains of 2–10% and setting the new state-of-the-art on five well-studied datasets.

Acknowledgements

This research was supported by ONR (N00014-18-1-2826, N00014-17-S-B001), DARPA N66001-19-2-403, NSF (IIS-1616112, IIS-1252835, IIS-1562364), ARO (W911NF-16-1-0121), an Allen Distinguished Investigator Award, Samsung GRO and gifts from Allen Institute for AI, Google and Amazon.

The authors would like to thank the anonymous reviewers, Eunsol Choi, Christopher Clark, Victor Zhong and UW NLP members for their valuable feedback.

References

Daniel A Abolafia, Mohammad Norouzi, Jonathan Shen, Rui Zhao, and Quoc V Le. 2018. Neural program synthesis with priority queue training. *arXiv preprint arXiv:1801.03526*.

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

Chris Alberti, Kenton Lee, and Michael Collins. 2019. A BERT baseline for the Natural Questions. *arXiv preprint arXiv:1901.08634*.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. In *ACL*.

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

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *ACL*.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *ACL*.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *CoNLL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *ACL*.

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 *NAACL*.

Pengcheng He, Yi Mao, Kaushik Chakrabarti, and Weizhu Chen. 2019. X-SQL: reinforce schema representation with context. *arXiv preprint arXiv:1908.08113*.

Niall Hurley and Scott Rickard. 2009. Comparing measures of sparsity. *IEEE Transactions on Information Theory*.

Wonseok Hwang, Jinyeung Yim, Seunghyun Park, and Minjoon Seo. 2019. A comprehensive exploration on WikiSQL with table-aware word contextualization. *arXiv preprint arXiv:1902.01069*.

Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang. 2017. Search-based neural structured learning for sequential question answering. In *ACL*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*.

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *ACL*.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The NarrativeQA reading comprehension challenge. *TACL*.

Jayant Krishnamurthy, Pradeep Dasigi, and Matt Gardner. 2017. Neural semantic parsing with type constraints for semi-structured tables. In *EMNLP*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Change, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *TACL*.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering. In *ACL*.

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 *ACL*.

Chen Liang, Mohammad Norouzi, Jonathan Berant, Quoc V Le, and Ni Lao. 2018. Memory augmented policy optimization for program synthesis and semantic parsing. In *NIPS*.

Percy Liang, Michael I Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Sewon Min, Eric Wallace, Sameer Singh, Matt Gardner, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2019. Compositional questions do not necessitate multi-hop reasoning. In *ACL*.

Kyosuke Nishida, Itsumi Saito, Kosuke Nishida, Kazutoshi Shinoda, Atsushi Otsuka, Hisako Asano, and Junji Tomita. 2019. Multi-style generative reading comprehension. In *ACL*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *ICLR*.

Swabha Swayamdipta, Ankur P Parikh, and Tom Kwiatkowski. 2018. Multi-mention learning for reading comprehension with neural cascades. In *ICLR*.

Alon Talmor and Jonathan Berant. 2019. MultiQA: An empirical investigation of generalization and transfer in reading comprehension. In *ACL*.

Yi Tay, Anh Tuan Luu, Siu Cheung Hui, and Jian Su. 2018. Densely connected attention propagation for reading comprehension. In *NIPS*.

Wei Wang, Ming Yan, and Chen Wu. 2018. Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. In *ACL*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Caiming Xiong, Victor Zhong, and Richard Socher. 2018. DCN+: Mixed objective and deep residual coattention for question answering. In *ICLR*.

Xiaojun Xu, Chang Liu, and Dawn Song. 2018. SQLNet: Generating structured queries from natural language without reinforcement learning. In *ICLR*.

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 *EMNLP*.

Adams Wei Yu, David Dohan, Quoc Le, Thang Luong, Rui Zhao, and Kai Chen. 2018a. Fast and accurate reading comprehension by combining self-attention and convolution. In *ICLR*.

Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018b. TypeSQL: Knowledge-based type-aware neural text-to-sql generation. In *NAACL*.

Luke S. Zettlemoyer and Michael Collins. 2005. Learning to map sentences to logical form: Structured classification with probabilistic categorial grammars. In *UAI*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2017. Seq2SQL: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*.

Zhi-Hua Zhou, Yu-Yin Sun, and Yu-Feng Li. 2009. Multi-instance learning by treating instances as non-i.i.d. samples. In *ICML*.

A Model details

We describe the detailed model architecture used as a base model. In other words, we describe how we obtain $\mathbb{P}(z|x;\theta)$.

The following paragraphs describe (i) BERT extractive model used for multi-mention RC (Section 5.1) and (ii) BERT sequence tagging model for discrete reasoning task (Section 5.2), respectively. For QANET for discrete reasoning and the model for SQL generation (Section 5.3), we use the open-sourced code of the original implementation[9] of Dua et al. (2019) and Hwang et al. (2019) and do not make any modification except the objective function, so we refer to original papers.

All implementations are done in Pytorch (Paszke et al., 2017). For BERT, we modify the open-sourced implementation in PyTorch[10] and use the uncased version of $\text{BERT}_{\text{base}}$.

Extractive QA model for multi-mention RC The model architecture is closed to that of Min et al. (2019) and Alberti et al. (2019), where the model operates independently on each paragraph, and selects the best matching paragraph and its associated answer span.

The input is a question Q and a set of paragraphs $\{P_1, \ldots, P_N\}$, and the desired output is a span from one of paragraphs. Since our goal is to compute a probability of a specific span, z, let's say z is s-th through e-th word in k-th paragraph.

The model receives a question Q and a single paragraph P_i in parallel. Then, $S_i = Q : [\texttt{SEP}] : P$, a list of $m + n_i + 1$ words, where : indicates a concatenation, [SEP] is a special token, m is the length of Q, and n_i is the length of P_i. This S_i is fed into BERT:

$$\bar{S}_i = \text{BERT}(S) \in \mathbb{R}^{h\times(m+n_i+1)}$$

where h is the hidden dimension of BERT. Then,

$$\begin{aligned} p_{i,\text{start}} &= \text{Softmax}\big(\bar{S}_i^T W_1\big) \in \mathbb{R}^{m+n_i+1} \\ p_{i,\text{end}} &= \text{Softmax}\big(\bar{S}_i^T W_2\big) \in \mathbb{R}^{m+n_i+1} \end{aligned}$$

where $W_1, W_2 \in \mathbb{R}^h$ are learnable vectors.

Finally, the probability of z, s-th through e-th word in i-th paragraph, is obtained by:

$$\mathbb{P}(z|Q,P_i) = p^s_{i,\text{start}} \times p^e_{i,\text{end}}$$

where p^d denotes d-th element of the vector p.

Separately, a paragraph selector is trained through $p_{i,\text{exit}} = \text{Softmax}\big(W_3 \text{maxpool}(\bar{S}_i)\big) \in \mathbb{R}^2$ where $W_3 \in \mathbb{R}^{h\times 2}$ is learnable vector. At inference time, $k = \text{argmax}_i p^1_{i,\text{exit}}$ is computed and $\mathbb{P}(z|Q,P_k)$ is only considered to output a span.

Sequence Tagging model for discrete reasoning The basic idea of the model is closed to that of Dua et al. (2019). The input is a question Q and a paragraph P. Our goal is to compute a probability of an equation, $z = (o_1, n_1, o_2, n_2)$, where $o_1, o_2 \in \{+, -, *0.01\}$ and $n_1, n_2 \in N_P \cup N_Q \cup S$, N_P and N_Q are all numeric values appearing in P and Q, and S are a set of predefined special numbers.[11].

First, BERT encodings of the question and the paragraph is obtained via

$$\bar{S} = \text{BERT}(Q : [\texttt{SEP}] : P) \in \mathbb{R}^{h\times(m+n+1)}$$

where : indicates a concatenation, [SEP] is a special token, m is the length of Q, n is the length of P, and h is the hidden dimension of BERT. Then,

$$\begin{aligned} p_{\text{input}} &= \text{Softmax}\big(\bar{S}^T W_1\big) \in \mathbb{R}^{(m+n+1)\times 4} \\ p_{\text{special}} &= \text{Softmax}\big(\text{maxpool}(\bar{S}) W_2\big) \in \mathbb{R}^{|S|\times 4} \end{aligned}$$

where $W_1 \in \mathbb{R}^{h\times 4}$ and $W_2 \in \mathbb{R}^{h\times|S|\times 4}$ are learnable matrices. Then,

$$\mathbb{P}(z|x) = \Pi_{i=1}^{m+n+1} p^{g(i)}_{\text{input}} \Pi_{j=1}^{|S|} p^{h(j)}_{\text{special}}$$

where

$$g(i) = \begin{cases} \alpha(o_1) & \text{if } [Q : [\texttt{SEP}] : P]^i = n_1 \\ \alpha(o_2) & \text{if } [Q : [\texttt{SEP}] : P]^i = n_2 \\ 0 & \text{o.w.} \end{cases}$$

$$h(j) = \begin{cases} \alpha(o_1) & \text{if } S^j = n_1 \\ \alpha(o_2) & \text{if } S^j = n_2 \\ 0 & \text{o.w.} \end{cases}$$

$$\alpha(o) = \begin{cases} 1 & \text{if } o = \text{`+'} \\ 2 & \text{if } o = \text{`-'} \\ 3 & \text{if } o = \text{`*0.01'} \end{cases}$$

Here, subscript i of the vector or the sequence indicate i-th dimension of the vector or i-th element of the sequence, respectively.

[9] https://github.com/allenai/allennlp/blob/master/allennlp/models/reading_comprehension/naqanet.py and https://github.com/naver/sqlova

[10] https://github.com/huggingface/pytorch-pretrained-BERT

[11] $S = \{1, 2, 3, 4, 5, 7, 10, 12, 100, 1000\}$

τ	TRIVIAQA F1
None	55.83
20k	58.05
30k	57.99
40k	56.66

τ	DROP_{num} EM
None	52.31
5k	51.98
10k	52.82
20k	51.74

Dataset	batch size	τ
TRIVIAQA	20	15K
NARRATIVEQA	20	20K
TRIVIAQA-OPEN	192	4K
NATURALQUESTIONS-OPEN	192	8K
DROP_{num} with BERT	14	10K
DROP_{num} with QANET	14	None
WIKISQL	10	None

Table 6: **(Left)** Ablations with varying values of τ on TRIVIAQA. **(Middle)** Ablations with varying values of τ on DROP_{num} with BERT. **(Right)** Final τ chosen for the main results on each dataset. Note that for DROP_{num} with QANET and WIKISQL, we just report the number without annealing.

B Annealing

To prevent the model to be optimized on early decision of the model, we perform annealing: at training step t, the model optimizes on MML objective with a probability of $\min(t/\tau, 1)$ and otherwise use our objective, where τ is a hyperparameter. We observe that the performance is improved by annealing while not being sensitive to the hyperparameter τ. Ablations and chosen τ for each dataset are shown in Table 6. Note that for DROP_{num} with QANET and WIKISQL, we do not ablate with varying τ and just report the number without annealing.

C Examples

To see if the prediction from the model is the correct solution to derive the answer, we analyze outputs from the model.

TRIVIAQA. Table 7 shows one example from TRIVIAQA where the answer text (Montgomery) is mentioned in the paragraph multiple times. Predictions from the model with our training method and that with MML objective are shown in the red text and the blue text, respectively. The span predicted by the model with our method actually answers to the question, while other spans with the answer text is not related to the question.

DROP_{num}. Table 8 shows predictions from the model with our method and that with MML objective over training procedure. We observe that the model with our method learns to assign a high probability to the best solution ('34-24'), while the model with MML objective fails to do so. Another notable observation is that the model with our method assign sparse distribution of likelihood over Z, compared to the model with MML objective. We quantitatively define sparsity as

$$\frac{|\{z \in Z \text{ s.t. } \mathbb{P}(z|x;\theta) < \epsilon\}|}{|Z|}$$

(Hurley and Rickard, 2009) and show that the model with our method gives higher sparsity than the model with MML (59 vs. 36 with $\epsilon = 10^{-3}$, 54 vs. 17 with $\epsilon = 10^{-4}$ on DROP).

WIKISQL. WIKISQL provides the annotated SQL queries, makes it easy to compare the predictions from the model to the annotated queries. Out of 8421 examples from the development set, 7110 predictions execute the correct answers. Among those, 6296 predictions are exactly same as the annotated queries. For cases where the predictions execute the correct answers but are not exactly same as the groundtruth queries, we show four examples in Table 9. In the first example, both the annotated query and the prediction are correct, because the selected column does not matter for counting. Similarly in the second example, both queries are correct because `Capital (exonym)` and `Capital (endonym)` both indicate the capital city. In the third example, the prediction makes more sense than the annotated query because the question does not imply anything about `min`. In the last example, the annotated query makes more sense than the prediction because the prediction misses `Ship Type = battleship`. We conjecture that the model might learn to ignore some information in the question if the table header implies the table is specific about that information, hence does not need to condition on that information.

Question & Document
Q: What is the state capital of Alabama? (Groundtruth: Montgomery) **D:** Alabama is nicknamed the Yellowhammer State, after the state bird. Alabama is also known as the "Heart of Dixie" and the "Cotton State". The state tree is the longleaf pine, and the state flower is the camellia. Alabama's capital is Montgomery. (...) From 1826 to 1846, Tuscaloosa served as Alabama's capital. On January 30, 1846, the Alabama legislature announced it had voted to move the capital city from Tuscaloosa to Montgomery. The first legislative session in the new capital met in December 1847. A new capitol building was erected under the direction of Stephen Decatur Button of Philadelphia. The first structure burned down in 1849, but was rebuilt on the same site in 1851. This second capitol building in Montgomery remains to the present day.

Table 7: An example from TRIVIAQA with multiple spans of the answer text (underlined). The model trained with self-training technique outputs the correct answer (red) and the model trained on MML objective does not (blue).

Question & Passage
Q: How many sports are not olympic sports but are featured in the asian games ? (A: 10) **P:** The first 30 sports were announced by the singapore national olympic council on 10 december 2013 on the sidelines of the 27th sea games in myanmar. It announced then that there was room for as many as eight more sports. On 29 april 2014 the final six sports namely boxing equestrian floorball petanque rowing and volleyball were added to the programme. Floorball will feature in the event for the first time after being a demonstration sport in the 2013 edition. In its selection of events the organising committee indicated their desire to set a model for subsequent games in trimming the number of 'traditional' sports to refocus on the seag' s initial intent to increase the level of sporting excellence in key sports. Hence despite room for up to eight traditional sports only two floorball and netball were included in the programme. Amongst the other 34 sports 24 are olympic sports and all remaining sports are featured in the asian games.

Ours

t	Pred	Z (ordered by $\mathbb{P}(z\|x;\theta_t)$)			
1k	10+two	eight+two	eight+two‡	34-24	10
2k	30-24	34-24	eight+two	eight+two‡	10
4k	34+24	34-24	eight+two	eight+two‡	10
8k	34+24	34-24	eight+two	10	eight+two‡
16k	34-24	34-24	eight+two	eight+two‡	10
32k	34-24	34-24	eight+two	eight+two‡	10

MML

t	Pred	Z (ordered by $\mathbb{P}(z\|x;\theta_t)$)			
1K	10+two	eight+two	eight+two‡	10	34-24
2k	eight+two	eight+two	eight+two‡	34-24	10
4k	24+5	eight+two	eight+two‡	34-24	10
8k	24+5	eight+two	eight+two‡	34-24	10
16k	34-24	34-24	eight+two	eight+two‡	10
32k	24+5	34-24	eight+two	eight+two‡	10

Table 8: An example from DROP_{num}, with its answer text '10' and a subset of Z, containing '10', two of 'eight+two' (which 'eight' come from different mentions; one denoted by '‡' for distinction) and '34-24'. The below tables are predictions from the model with our training strategy (left) and MML (right). For each training step t, the top 1 prediction and Z ordered by $P(z|x;\theta_t)$, a probability of $z \in Z$ with respect to the model at t are shown. Note that at inference time Z cannot be obtained, so top 1 prediction is not necessarily in Z.

Q	How many times was the # of total votes 2582322?
H	Election, # of candidates nominated, # of seats won, # of total votes, % of popular vote
A	`Select count(# of seats won) where # of total votes = 2582322`
P	`Select count(Election) where # of total votes = 2582322`
Q	What official or native languages are spoken in the country whose capital city is Canberra?
H	Country (exonym), Capital (exonym), Country (endonym) Capital (endonym), Official or native language
A	`Select Official or native languages where Capital (exonym) = Canberra`
P	`Select Official or native languages where Capital (endonym) = Canberra`
Q	What is the episode number that has production code 8abx15?
H	No. in set, No. in series, Title, Directed by, Written by, Original air date, Production code
A	`Select min(No.in series) where Production code = 8ABX15`
P	`Select No.in series where Production code = 8abx15`
Q	what is the name of the battleship with the battle listed on May 13, 1915?
H	Estimate, Name, Nat., Ship Type, Principal victims, Date
A	`Select Name where Ship Type = battleship and Date = may 13, 1915`
P	`Select Name where Date = may 13, 1915`

Table 9: Four examples from WIKISQL where the prediction from the model is different from annotated SQL query, although the executed answers are the same. **Q**, **H**, **A** and **P** indicate the given question, the given table header, annotated SQL query and predicted SQL query. First two example shows the case where both queries are correct. Next example shows the case that the model prediction makes more sense than the annotated query. The last example shows the cases that the annotated query makes more sense than the model prediction.