OPERA: Operation-Pivoted Discrete Reasoning over Text

Yongwei Zhou[1], Junwei Bao[2]*, Chaoqun Duan[2], Haipeng Sun[2], Jiahui Liang[2], Yifan Wang[2], Jing Zhao[2], Youzheng Wu[2], Xiaodong He[2], Tiejun Zhao[1]†
[1]Harbin Institute of Technology [2]JD AI Research
ywzhou@hit-mtlab.net baojunwei@jd.com tjzhao@hit.edu.cn

Abstract

Machine reading comprehension (MRC) that requires discrete reasoning involving symbolic operations, e.g., addition, sorting, and counting, is a challenging task. According to this nature, semantic parsing-based methods predict interpretable but complex logical forms. However, logical form generation is nontrivial and even a little perturbation in a logical form will lead to wrong answers. To alleviate this issue, multi-predictor -based methods are proposed to directly predict different types of answers and achieve improvements. However, they ignore the utilization of symbolic operations and encounter a lack of reasoning ability and interpretability. To inherit the advantages of these two types of methods, we propose **OPERA**, an operation-pivoted discrete reasoning framework, where lightweight symbolic operations (compared with logical forms) as neural modules are utilized to facilitate the reasoning ability and interpretability. Specifically, operations are first selected and then softly executed to simulate the answer reasoning procedure. Extensive experiments on both DROP[1] and RACENum datasets show the reasoning ability of OPERA. Moreover, further analysis verifies its interpretability. [2]

1 Introduction

Machine reading comprehension (MRC) that requires discrete reasoning is a valuable and challenging task (Dua et al., 2019), especially involving symbolic operations such as addition, sorting, and counting. The examples in Table 1 illustrate the task. To answer the question "*Who threw the longest pass*", it requires a model to choose the person with the longest pass from all the people and corresponding distance pairs based on the given passage. This task has various application scenarios in the real world, such as analyzing financial reports and sports news.

Passage: Houston would tie the game in the second quarter with kicker Kris Brown getting a 53-yard and a 24-yard field goal. Oakland would take the lead in the third quarter with wide receiver Johnnie Lee Higgins catching a 29-yard touchdown pass from Russell, followed up by an 80-yard punt return...
Question: Who threw the longest pass?
Answer: Russell

Table 1: An example of question-answer pair along with a passage from DROP dataset (Dua et al., 2019). Question words in color indicate the potential operations for reasoning, i.e., `ARGMAX` and `KEY_VALUE`.

Existing approaches for this task can be roughly divided into two categories: the semantic-parsing-based (Chen et al., 2020b; Gupta et al., 2020) and the multi-predictor-based methods (Dua et al., 2019; Ran et al., 2019; Hu et al., 2019; Chen et al., 2020a; Zhou et al., 2021). The former maps the natural language utterances into logical forms and then executes them to derive the answers. For example, Chen et al. (2020b) propose NeRd. It includes a reader to encode the passage and question, and a programmer to generate a logical form for multi-step reasoning. Intuitively, this method has an advantage in interpretability. However, semantic parsing over text is nontrivial and even a little perturbation will result in wrong answers, which hinders the MRC performance.

To alleviate the heavy dependence on logical forms in the first category, the latter directly employs multiple predictors to derive different types of answers. For example, Dua et al. (2019) and Chen et al. (2020a) divide instances of the DROP dataset into several types and design a model with multi-predictors to deal with different answer types, i.e., question/passage span(s), count, and arithmetic

*Work was done during the first author's internship at JD AI mentored by Junwei Bao: baojunwei001@gmail.com.
†Corresponding author.
[1]https://leaderboard.allenai.org/drop/submissions/public
[2]Codes is released at https://github.com/JD-AI-Research-NLP/OPERA.

Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1655 - 1666

expression. It is the capability of deriving different types of answers that improves the performance of models. However, such methods are lack of the necessary components to imitate discrete reasoning, which leads to inadequacy in reasoning ability and interpretability.

To alleviate the shortcomings of the above methods and preserve their advantages, we attempt to summarize reasoning steps into a set of operations and adopt them as the pivot to connect the question and the answer, which makes it possible to perform discrete reasoning. For example, to answer the question in Table 1, it needs two steps: (1) finding all persons and the corresponding distance of touchdown pass, and (2) choosing the one with the longest pass among them. We attempt to convert them into two operations, `KEY_VALUE` and `ARGMAX`, respectively. We then use them to produce the answer. Specifically, we design a set of lightweight symbolic operations (compared with logical forms) to cover all of the questions in the datasets and utilize them as neural modules to facilitate reasoning ability and interpretability. We denote this method as **OPERA**, an operation-pivoted discrete reasoning MRC framework. To utilize the operations, we propose an operation-pivoted reasoning mechanism composed of an operation selector and an operation executor. Specifically, the operation selector automatically identifies relevant operations based on the input. To enhance the performance of this sub-mechanism, we further design an auxiliary task to learn the alignment from a question to operations according to a set of heuristics rules. The operation executor softly integrates the selected operations to perform discrete reasoning over text via an attention mechanism (Vaswani et al., 2017).

To verify the effectiveness of the proposed method, comprehensive experiments are conducted on both the DROP and RACENum datasets, where RACENum used in this paper is a subset of the RACE dataset following (Chen et al., 2020a). Experimental results indicate that our method outperforms strong baselines and achieves the state-of-the-art on both datasets under the single model setting. We further analyze the interpretability of OPERA. Overall, this paper primarily makes the following contributions:

(1) We propose OPERA, an operation-pivoted discrete reasoning MRC framework, improving both the reasoning ability and interpretability.

(2) Extensive experiments on DROP and RACENum dataset demonstrate the reasoning ability of OPERA. Moreover, statistic analysis and visualization indicate the interpretability of OPERA.

(3) We systematically design operations and heuristic rules to map questions to operations, aiming to facilitate research on symbolic reasoning.

2 Related Work

Recently, machine reading comprehension (MRC) methods tend to deal with more practical problems (Yang et al., 2018; Dua et al., 2019; Zhao et al., 2021), for example, answering complex questions that require discrete reasoning (Dua et al., 2019) such as arithmetic computing, sorting, and counting. Intuitively, semantic parsing-based methods, which are well explored to deal with discrete reasoning in question answering with structured knowledge graphs (Bao et al., 2016) or tables, have potential to address the discrete reasoning MRC problem. Therefore, semantic parsing-based methods for discrete reasoning over text are proposed to firstly convert the unstructured text into a table, and then answer questions over the structured table with a grammar-constrained semantic parser (Krishnamurthy et al., 2017). NeRd (Chen et al., 2020b) is a generative model that consists of a reader and a programmer, which are responsible for encoding the context into vector representation and generating grammar-constrained logical forms, respectively. NMNs (Gupta et al., 2020) learned to parse compositional questions as executable logical forms. However, it only adapts to limited question types matched with a few pre-defined templates.

Multi-predictor-based methods employ multiple predictors to derive different types of answers. NAQANET (Dua et al., 2019), a number-aware framework, employed multiple predictors to produce corresponding answer types, including a span, count, and arithmetic expression. Based on NAQANET, MTMSN(Hu et al., 2019) added a negation predictor to solve the negative question and re-ranked arithmetic expression candidates. To aggregate the relative magnitude relation between two numbers, NumNet (Ran et al., 2019) and NumNet+ leveraged a graph convolution network to perform multi-step reasoning over a number graph. QDGAT (Chen et al., 2020a) proposed a question-directed graph attention network for reasoning over a heterogeneous graph composed of entity and number nodes. EviDR (Zhou et al., 2021), an evidence-

Operations	Description	Examples
ADDITION/DIFF	Addition or subtraction	*How many more yards was Kris Browns's first field goal over his second?*
MAX/MIN	Select the maximum/minimum one from given numbers	*How many yards was the longest field goal in the game?*
ARGMAX/ARGMIN	Select key with highest/lowest value from key-value pairs	*Which player had the longest touchdown reception?*
ARGMORE/ARGLESS	Select key with higher/lower value from two key-value pairs	*Who scored more field goals, David Akers or John Potter?*
COUNT	Count the number of spans	*How many field goals did Kris Brown kick?*
KEY_VALUE	Extract key-value pairs	*How many percent of Forth Worth households owned a car?*
SPAN	Select spans from input sequence	*Which team scored the final TD of the game?*

Table 2: All the operations, descriptions and the corresponding examples.

emphasized MRC method, performed reasoning over a multi-grained evidence graph. Compared with these existing methods, our proposed OPERA focuses on bridging the gap from questions to answers with operations and integrating them to simulate discrete reasoning.

3 Approach

3.1 Task and Model Overview

Given a question Q and a passage P, MRC that requires discrete reasoning aims to predict an answer $\hat{A}$ with the maximum probability over the candidate space Ω as follows:

$$\hat{A} = \arg\max_{A \in \Omega} p(A|Q, P) \tag{1}$$

where the answer $\hat{A}$ in this task could not only be span(s) extracted from context but also a number calculated with some numbers in context. To handle this task, it generally requires not only natural language understanding but also performing discrete reasoning over text, such as comparison, sorting and arithmetic computing.

To address the aforementioned challenges in this task, we propose OPERA, an operation-pivoted discrete reasoning MRC framework and it is briefly illustrated in Figure 1. In our framework, a set of operations $\mathcal{OP}$, defined in Table 2, are introduced to support the modeling of answer probability $p(A|Q, P)$ as follows:

$$p(A|Q, P) = \sum_{O \in \mathcal{OP}} p(A|Q, P, O)p(O|Q, P), \tag{2}$$

where $O \in \mathcal{OP}$ represents one of the operations. Concretely, in our framework, we first design an operation selector $p(O|P, Q)$ for choosing the correct question-related operations. These selected operations are then *softly* executed over the given context. Eventually, answer predictor $p(A|Q, P, O)$ utilizes the execution result to predict the final answer.

3.2 Definition of Operations

To imitate discrete reasoning, we design a set of operations $\mathcal{OP}$ as shown in Table 2. The set contains 11 operations and each one represents a reasoning unit. Specifically, for questions that need to be answered by calculation, we design operations ADDITION/DIFF to represent addition and subtraction. For questions which need to be answered by counting or sorting, we also design operations COUNT, MAX/MIN, ARGMAX/ARGMIN, and ARGMORE/ARGLESS. The rest operations KEY_VALUE and SPAN are used to extract spans from the question and the passage. To incorporate them into OPERA, each operation is denoted as a tuple. Formally, i-th operation $\mathcal{OP}_i$ is $\langle \mathbf{E}_i^{OP}, f_i(\cdot) \rangle$, where $i \in \{1, 2, ..., n\}$ and n is the numbers of operations. $\mathbf{E}_i^{OP} \in \mathbb{R}^{d_h}$ represents the learnable embedding of the i-th operation. $f_i(\cdot)$ is a neural executor parameterized with trainable matrices $\mathbf{W}_{q,i}^{OP}$, $\mathbf{W}_{k,i}^{OP}$ and $\mathbf{W}_{v,i}^{OP} \in \mathbb{R}^{d_h \times d_h}$. The neural executor $f_i(\cdot)$ is capable of performing execution of $\mathcal{OP}_i$ on the given context. Specifically, it takes the representation of context as input and outputs the executed representation as $\mathbf{m}_i^{OP}$ (§ 3.3.2).

3.3 Architecture of OPERA

3.3.1 Context Encoder

The context encoder aims to learn the contextual representation of the input. Formally, given a question Q and a passage P, we concatenate them into a sequence and feed it into a pre-trained language model (Liu et al., 2019; Clark et al., 2020; Lan et al., 2020) to obtain their whole representation $\mathbf{H} \in \mathbb{R}^{l \times d_h}$. After that, we split $\mathbf{H}$ into the question and passage representations, which are respectively denoted as $\mathbf{H}^Q \in \mathbb{R}^{l_q \times d_h}$ and $\mathbf{H}^P \in \mathbb{R}^{l_p \times d_h}$. l_q, l_p, and l are the number of tokens in question, passage and concatenation of them. d_h is the dimension of the representations.

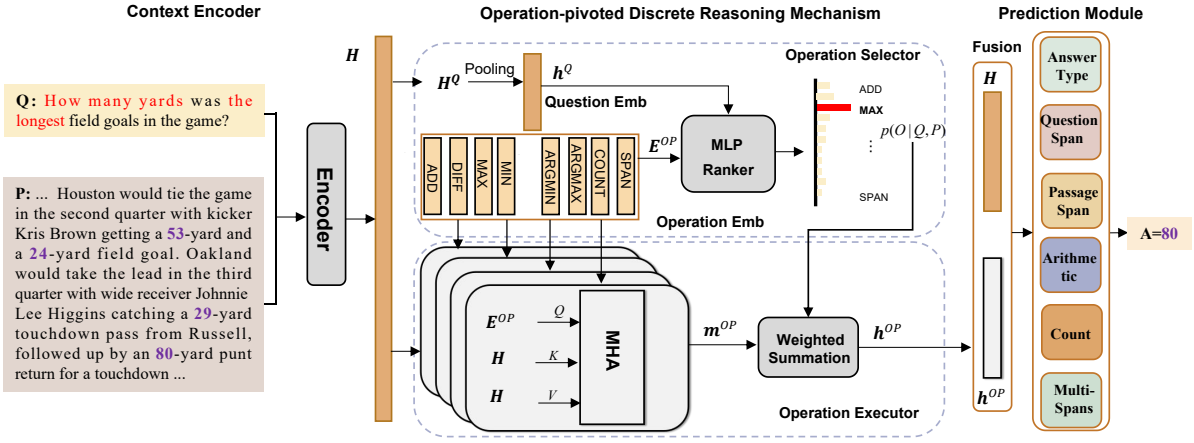

Figure 1: The architecture of OPERA. It consists of a context encoder, an operation-pivoted reasoning module, and a prediction module. The prediction module supports five types of answers, including question span, passage span, arithmetic expression, count, and multi-spans. MHA means a multi-head attention mechanism.

3.3.2 Operation-pivoted Discrete Reasoning

The operation-pivoted reasoning module is composed of an operation selector and an operation executor. The operation selector is adopted to select operations related to the given question. The operation executor is responsible for imitating the execution of the selected operations with an attention mechanism.

Operation Selector To imitate discrete reasoning, existing methods usually adopt a logical form generated by a semantic parser to address this task. However, these methods suffer severely from the cascade error, where a little perturbation in the logical form may result in wrong answers. Therefore, we propose to map each question into an operation set, instead of logical forms. Namely, we intend to select relevant operations from the $\mathcal{OP}$. To this end, we adopt a bilinear function to compute the similarity between each operation and the question and normalize them with a softmax as follows:

$$p(O|Q,P) = \texttt{softmax}(\mathbf{E}^{OP}\mathbf{W}\mathbf{h}^{Q}), \quad (3)$$

where $\mathbf{E}^{OP} \in \mathbb{R}^{n \times d_h}$ is a learnable parameter, which demotes the operation embedding matrix. $\mathbf{h}^{Q} \in \mathbb{R}^{d_h}$ is the representation of the question, which is obtained by executing weighted pooling on the $\mathbf{H}^{Q}$. $\mathbf{W} \in \mathbb{R}^{d_h \times d_h}$ is a parameter matrix and $p(O|Q,P)$ is the distribution over operations.

Operation Executor The operation executor is responsible for performing the execution of the selected operations over the given context. Inspired by previous studies (Andreas et al., 2016; Gupta et al., 2020), we implement the operation executor based on the neural module network, which takes advantage of neural network in fitting and generalization, and the composition characteristics of symbolic processing. Specifically, for each operation $\mathcal{OP}_i = \langle \mathbf{E}_i^{OP}, f_i(\cdot) \rangle$, $i = \{1, 2, ..., n\}$, we use a multi-head cross attention mechanism (Vaswani et al., 2017) to implement $f_i(\cdot)$. In detail, we leverage the embedding of each operation $\mathbf{E}_i^{OP}$ as query and the representations of the whole input sequence $\mathbf{H}$ as keys and values, respectively, to model $f_i(\cdot)$ as follows:

$$\alpha_i^{OP} = \texttt{softmax}(\frac{(\mathbf{E}_i^{OP}\mathbf{W}_{q,i}^{OP})(\mathbf{H}\mathbf{W}_{k,i}^{OP})^T}{\sqrt{d_h}}), \quad (4)$$

$$\mathbf{m}_i^{OP} = \alpha_i^{OP}(\mathbf{H}\mathbf{W}_{v,i}^{OP}), \quad (5)$$

where $\mathbf{W}_{q,i}^{OP}, \mathbf{W}_{k,i}^{OP}, \mathbf{W}_{v,i}^{OP} \in \mathbb{R}^{d_h \times d_h}$ are the parameter matrices in executor of operation $\mathcal{OP}_i$. $\mathbf{m}_i^{OP} \in \mathbb{R}^{d_h}$ is the representation of the execution result of the i-th operation.

Finally, we softly integrate all of the execution results as the final output $\mathbf{h}^{OP} \in \mathbb{R}^{d_h}$ with the distribution $p(O|Q,P)$ as follows:

$$\mathbf{h}^{OP} = \sum_{i=1}^{n} p(O = \mathcal{OP}_i|Q,P)\mathbf{m}_i^{OP}. \quad (6)$$

The operation-aware semantic representation $\mathbf{h}^{OP}$ is further fed into the prediction module to reason the final answer (§ 3.3.3).

As described above, OPERA introduces operations that assist in understanding questions and integrates them into the model to perform discrete reasoning. Therefore, it achieves an advantage in

the reasoning capability and interpretability over the previous multi-predictor-based methods (Hu et al., 2019; Chen et al., 2020a; Zhou et al., 2021). Moreover, soft execution and composition of operations in OPERA alleviate the cascaded error that the semantic parsing methods (Ran et al., 2019; Chen et al., 2020b) suffer from. More experiments and analyses about reasoning ability and interpretability are illustrated in § 4.4 and § 4.5.

3.3.3 Prediction Module

In this section, we introduce the prediction module to derive different types of answers via multi-predictors. Each predictor first reasons out a derivation and then performs execution to obtain the final answer. This answer prediction procedure is formalized as follows:

$$p(A|Q,P,O) = \sum_{D \in \mathcal{D}} \mathbb{I}(g(D) = A) p(D|Q,P,O), \tag{7}$$

where $\mathbb{I}(g(D) = A)$ is an indicator function with value 1 if the answer A can be derived from a derivation executor $g(\cdot)$ based on D, and 0 otherwise. $p(D|Q,P,O)$ models the derivation prediction. Specifically, a derivation $D = \langle T, L \rangle$ includes an answer type T and a corresponding label L. For example, in Table 3, the textual answer A of the question "*how many yards was the longest field goals in the game*" is "80". The possible derivations $\mathcal{D}$ to this answer include a span $\langle \texttt{Span}, (100, 102) \rangle$, and an arithmetic expression $\langle \texttt{AE}, (0 * 29) + (1 * 80) \rangle$. Inspired by previous studies (Chen et al., 2020a; Zhou et al., 2021), the derivation predictor

$$p(D|Q,P,O) = \sum_{T \in \mathcal{T}} p_T(L|Q,P,O) p(T|Q,P,O) \tag{8}$$

is decomposed into an answer type predictor $p(T|Q,P,O)$ and corresponding label predictors $p_{T \in \mathcal{T}}(L|Q,P,O)$ where $\mathcal{T} = \{$*Question Span*, *Passage Span*, *Count*, *Arithmetic Expression*, *Multi-spans*$\}$ includes all the answer types defined in this paper. Each label predictor takes question-passage representation $\mathbf{H}$ and the operation-pivo representation $\mathbf{h}^{OP}$ as input and calculates the probability of label L. Specifically, these label predictors are specified as follows and more details are shown in Appendix A.3.

Question / Passage Span The probability of a question/passage span is the product of the probabilities of the start index and the end index. Following MTMSN (Hu et al., 2019), we use a question-aware decoding strategy to predict the start and end index across the input sequence, respectively.

Count As indicated in QDGAT (Chen et al., 2020a), questions with 0-9 as answers account for 97% in all the count questions. Hence, such questions are modeled as a 10-class (0-9) classification problem.

Arithmetic Expression Similar to NAQANet (Dua et al., 2019), we first assign a sign (positive, negative, or zero) to each number in the context and then compute the answer by summing them.

Multi-spans Inspired by Segal et al. (2020), the multi-span answer (a set of non-contiguous spans) is derived with a sequence labeling method, in which each token of the input is tagged with `BIO` labels. Finally, each span which is tagged with continuous `B` and `I` is taken as a candidate span.

3.4 Learning with Weak Supervision

3.4.1 Training Instance Construction

Each training instance is originally composed of a passage P, a question Q, and answer text A. Since the derivations (i.e., labels for the spans, arithmetic expressions, and count) are not provided, weak supervision is adopted in OPERA. Specifically, for each training instance, given the golden textual answer A, we heuristically search all the possible derivations $\mathcal{D}$ as supervision signals, each of which can derive the correct answer A. Table 3 shows an example of $\mathcal{D}$.

In addition, we propose heuristic rules to map a question to its related operations denoted as $\mathcal{O} \subseteq \mathcal{OP}$. For example, to detect the operations intimated by the question Q in Table 3, we design a question template "*how many yards [Slot] longest [Slot]*" which maps matched questions to the operation `MAX`. Overall, a training instance can be constructed as a tuple $\langle P, Q, A, \mathcal{O}, \mathcal{D} \rangle$. The one-shot heuristic rules to obtain operation labels reduce the cost of human annotations. Moreover, when applying OPERA to other discrete reasoning MRC tasks, both the operations $\mathcal{OP}$ and the heuristic rules can be extended and adjusted if necessary. Fortunately, there is no need to construct strict logical forms in our architecture, but only the set of lightweight operations involved in the question. It tremendously reduces the difficulty of adapting OPERA to other discrete reasoning MRC tasks.

Meanwhile, we analyze the distribution of operations in the training set. More details about the

P	*...Oakland would take the lead in the third quarter with wide receiver Johnnie Lee Higgins catching a 29-yard touchdown pass from Russell, followed up by an 80-yard punt return for a touchdown ...*
Q	*How many yards was the longest field goals*
A	80
$\mathcal{O}$	MAX $\Leftarrow$ *How many yards [Slot] longest [Slot]*
$\mathcal{D}$	$\langle$Span, $(100, 102)\rangle$; $\langle$AE, $(0 * 29) + (1 * 80)\rangle$

Table 3: An example of building training instances.

heuristic rules for mapping questions to operations and the operation distribution in the dataset are respectively given in the Appendix A.1 and A.2.

3.4.2 Joint Training

The training objective consists of two parts, including the loss for answer prediction and operation selection. The loss for answer prediction $\mathcal{L}_a$ is

$$\mathcal{L}_a = -\log p(A|Q, P). \tag{9}$$

Note that the calculation of loss $\mathcal{L}_a$ takes all possible derivations that can obtain the correct answer A into account, which means that OPERA does not require labeling answer types for training. In addition, to learn better alignment from a question to operations, we introduce auxiliary supervision for the operation selector and calculate the loss

$$\mathcal{L}_{op} = -\sum_{O \in \mathcal{O}} \log p(O|Q, P), \tag{10}$$

where $\mathcal{O}$ indicates the operations provided by the heuristic rules. Finally, OPERA is optimized by minimizing the loss $\mathcal{L} = \mathcal{L}_a + \lambda \mathcal{L}_{op}$ where λ is a hyperparameter as a trade-off of the two objectives.

4 Experiment

4.1 Dataset and Evaluation

We conduct experiments on the following two MRC datasets to examine the discrete reasoning capability of our model. We employ Exact Match (EM) and F1 score as the evaluation metrics.

DROP Question-answer pairs in DROP dataset (Dua et al., 2019) are crowd-sourced based on passages collected from Wikipedia. In detail, it contains 96.6K question-answer pairs, where 77400/9536/9615 samples are for training/development/test. Three kinds of answers are involved in the raw dataset, i.e., NUMBER (60.69%), SPANS (37.72%), and DATE (1.59%).

RACENum To investigate the generalization capability of OPERA, we compare OPERA to other strong baselines on samples of RACE (Lai et al., 2017). Following Chen et al. (2020a), we sample instances from RACE, denoted as RACENum, where the question of each instance starts with "how many". To conveniently evaluate the models on RACENum, we convert the format of instances in RACENum into the same as DROP, since RACE is a multi-choice MRC dataset. RACENum is divided into two categories, i.e., middle/high school exam (RACENum-M/H). They respectively contain 609 and 565 questions, where the scale is a bit different from that reported in Chen et al. (2020a)[3].

4.2 Baselines

We compare OPERA with various prior systems in terms of reasoning capability and interpretability.
w/o Pre-trained Language Model: NAQANET (Dua et al., 2019) leverages several answer predictors to produce corresponding types of answers, including a span, count, and arithmetic expression. NumNet (Ran et al., 2019) leverages a graph convolution network to reason over a number graph aggregated relative magnitude among numbers.
w/ Pre-trained Language Model: GenBERT (Geva et al., 2020) injects reasoning capability into BERT by pre-training with large-scale numerical data. Based on NAQANET, MTMSN (Hu et al., 2019) adds a negation predictor to solve the negative question and re-rank arithmetic expression candidates. NeRd (Chen et al., 2020b) is essentially a generative semantic parser that maps questions and passages into executable logical forms. ALBERT-Calc was proposed for DROP by combining ALBERT with several predefined answer predictors (Andor et al., 2019). NumNet+ employs a pre-trained model to further boost the performance of NumNet. QDGAT (Chen et al., 2020a) builds a heterogeneous graph composed of entity and value nodes upon RoBERTa and utilizes a question-directed graph attention network to reason over the graph. EviDR (Zhou et al., 2021), an evidence-emphasized MRC model, performs reasoning over a multi-grained evidence graph based on ELECTRA.

4.3 Implementation Details

We utilize adam optimizer (Kingma and Ba, 2015) with a cosine warmup mechanism and set the

[3]Since no released code for dataset construction, we implement it referred to the algorithm in Chen et al. (2020a).

	BLR	LR	BWD	WD	Epochs	BS	n_h	d_h
RoBERTa	1.5e-5	5e-4	0.01	5e-5	12	16	16	1024
ELECTRA	1.5e-5	5e-4	0.01	5e-5	12	16	16	1024
ALBERT	3e-5	1e-4	0.01	5e-5	8	128	64	4096

Table 4: Hyperparameters settings for training OPERA.

weight of loss $\lambda = 0.3$ to train the model. The hyper-parameters are listed in Table 4, where BLR, LR, BWD, WD, BS, and d_h respectively represent the learning rate of the encoder, the learning rate of other parts of the model, the weight decay of the encoder, the weight decay of other parts of the model, batch size and hidden size of the model. Each operation is neutralized with a multi-head attention layer with n_h heads and d_h dimension.

4.4 Main Results

4.4.1 Results on DROP and Analysis

Table 5 shows the overall results of OPERA and all the baselines on the DROP dataset. OPERA achieves comparable and even higher performance than the recently available methods. Specifically, OPERA(RoBERTa) achieves comparable performance to QDGAT with advantages of 0.32 EM and 0.42 F1. OPERA(ELECTRA) exceeds EviDR by 0.89 EM and 0.90 F1 and OPERA(ALBERT) outperforms ALBERT-Calc by 4.84 EM and 4.24 F1. Moreover, the voting strategy is employed to ensemble 7 OPERA(ALBERT) models with different random seeds, achieving 86.26 EM and 89.12 F1 scores. We think the better performance comes from the modeling of discrete reasoning over text via operations, which mines more semantic information of context and explicitly integrates them into the answer prediction.

4.4.2 Results on RACENum

To investigate the generalization of OPERA for discrete reasoning, we additionally compare OPERA with QDGAT and NumNet+ on the RACENum dataset. We directly evaluate the three models without fine-tuning on RACENum due to its small scale. As Table 6 shows, the scores of models on the RACENum dataset are generally lower than that on the DROP dataset, which is attributed to the lack of in-domain training data. Nevertheless, the performance of OPERA significantly outperforms NumNet+ and QDGAT by a large margin of more than 3.49 EM and 3.53 F1 score on average. It indicates that OPERA has better generalization ability.

Method	Dev		Test	
	EM	F1	EM	F1
w/o Pre-trained Models				
NAQANet	46.20	49.24	44.07	47.01
NumNet	64.92	68.31	64.56	67.97
w/ Pre-trained Models				
GenBERT	68.80	72.30	68.6	72.35
MTMSN	76.68	80.54	75.88	79.99
NeRd	78.55	81.85	78.33	81.71
ALBERT-Calc	80.22	83.98	79.85	83.56
NumNet+	81.07	84.42	81.52	84.84
EviDR	82.09	85.14	82.55	85.80
QDGAT	82.74	85.85	83.23	86.38
Single Model Results				
OPERA(RoBERTa)	83.74	86.52	83.55	86.80
OPERA(ELECTRA)	83.86	86.66	83.46	86.70
OPERA(ALBERT)	**84.86**	**87.54**	**84.69**	**87.80**
Ensemble Results				
OPERA	86.79	89.41	86.26	89.12
Human			94.90	96.42

Table 5: Results on the DROP dataset. We solely compare with QDGAT, but leaving QDGAT$_p$ alone, since we focus on the reasoning mechanism in this work, while QDGAT$_p$ is a variant of QDGAT with data augmentation (Chen et al., 2020a).

Method	RACENum-M		RACENum-H		Avg	
	EM	F1	EM	F1	EM	F1
NumNet+	41.71	41.82	29.73	29.73	35.94	36.00
QDGAT	44.01	44.01	28.85	28.85	36.71	36.71
OPERA	**47.62**	**47.62**	**32.21**	**32.30**	**40.20**	**40.24**

Table 6: The performance of RoBERTa-based models on the RACENum dataset without finetuning.

4.5 Interpretability Analysis

Interpretability is an essential property for evaluating an MRC model. We analyze the interpretability of OPERA from the following two stages: (1) mapping from questions to operations, and (2) mapping from operations to answers.

Mapping from Question to Operation To explicitly show the correlations between questions and related operations, we manually evaluate the performance of the operation selection on 50 samples on the development set of DROP. Specifically, precision@n (P@n) is used as the evaluation metric, i.e., judging whether the top n predicted operations contain the correct ones according to questions. We finally achieve 0.88 on P@1 and 0.98 on P@2 for our model OPERA, which indicates that the operation selection module can accurately predict interpretable operations.

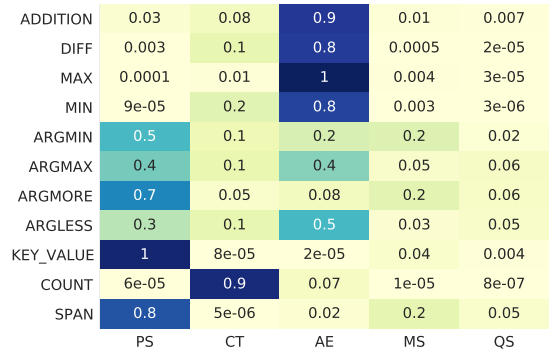

Figure 2: Statistical correlations between operations and answer types. The horizontal axis indicates answer types, and the vertical axis means operations. PS, CT, AE, MS and QS respectively means *passage span*, *count*, *arithmetic expression*, *multi-spans* and *question span*.

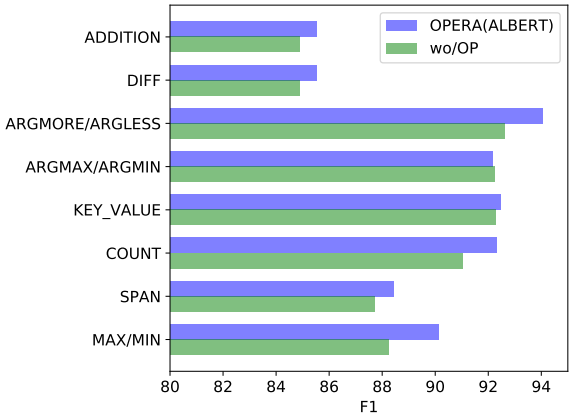

Figure 3: Ablation study on all the subsets of DROP containing some specific operations.

Method	NUM (60.69%)	SPANS (37.72%)	Overall
	EM / F1	EM / F1	EM / F1
OPERA(RB)	**85.91 / 86.14**	**83.89 / 88.90**	**83.74 / 86.52**
w/o $\mathcal{L}_{op}$	85.83 / 86.00	82.68 / 88.02	83.00 / 86.00
w/o OP	85.36 / 85.53	82.53 / 87.06	82.68 / 85.73
OPERA(AB)	**86.39 / 86.58**	86.30 / **90.99**	**84.86 / 87.54**
w/o $\mathcal{L}_{op}$	86.08 / 86.24	**86.39** / 90.96	84.72 / 87.36
w/o OP	85.62 / 85.89	84.95 / 89.83	84.01 / 86.79

Table 7: Ablation study on the dev set of DROP. RB and AB mean RoBERTa and ALBERT, respectively.

Mapping from Operation to Answer We explore the relations between operations and final answer types based on model predictions on the development set of DROP. Specifically, for each type of answer, the predicted operation distributions are summed over all samples and normalized, which derives a relation matrix as shown in Figure 2. We can observe that obvious correlations between operations and answer types exist. `ADDITION`, `DIFF`, `MAX` and `MIN` obviously correspond to *Arithmetic Expression*. The top 3 answer types for `KEY_VALUE` and `SPAN` are *Passage Span*, *Multiple Spans*, and *Question Span*. `COUNT` probably maps to *Count* answer type. `ARGMORE`, `ARGLESS`, `ARGMAX`, and `ARGMIN` are usually used for both *Passage Span* and *Arithmetic Expression*.

4.6 Ablation Study

Effect of Operations for OPERA As shown in Table 7, we first remove the loss of operation selection (w/o $\mathcal{L}_{op}$) by setting $\lambda = 0$, resulting in the performance degradation of 0.74 EM / 0.52 F1 points and 0.14 EM / 0.18 F1 points for OPERA based on RoBERTa and ALBERT, respectively. It indicates that the supervision for explicitly learning the alignment from a question to operations contributes to the reasoning capability of OPERA. Yet our approach is somewhat limited by the fact that the operation selector needs an auxiliary training task to work better, and heuristics rules are required to map questions into an operation set as training labels. Furthermore, we remove the total operation-pivoted reasoning module (w/o OP), the performance respectively declines by 1.06 EM / 0.79 F1 points and 0.85 EM / 0.75 F1 points for OPERA(RoBERTa) and OPERA(ALBERT). We also conduct the ablation study on the subsets containing a specific operation. As shown in Figure 3, OPERA achieves better performance than OPERA w/o OP on the majority of subsets. Overall, it confirms that integrating the operation-pivoted discrete reasoning mechanism contributes to the reasoning ability of the model.

Probe on Answer Types and Language Models As reported in Table 7, we observe that the performance on the NUMBER(NUM) and SPANS questions, which together account for 98.4% of the total, respectively declines by 0.55 EM / 0.61 F1 and 1.36 EM / 1.84 F1 when removing operation-pivoted reasoning mechanism from OPERA(RB). It demonstrates that this mechanism contributes to various answer types. Also, we respectively evaluate the performance of OPERA based on RoBERTa and ALBERT. We observe that OPERA(ALBERT) outperforms OPERA(RoBERTa) due to the stronger capability of semantic representation. Furthermore, integrating this mechanism consistently contributes to the performance of OPERA no matter it is based on RoBERTa or ALBERT. It indicates that OPERA

Question-Answer	Passage	NumNet+	QDGAT	OPERA
Q: How many total yards of touchdown passes were there? **A**: 73	... receiver Johnny Knox on a 23-yard touchdown pass. Afterwards, the Falcons took the lead as quarterback Matt Ryan completed a 40-yard touchdown pass to wide receiver Roddy White and a 10-yard touchdown pass to tight end Tony Gonzalez ...	**AnswerType**: Count **Answer**: 0	**AnswerType**: Count **Answer**: 0	**Top-1 OP**: ADDITION **AnswerType**: Arithmetic Expression **Answer**: 23+40+10=73
Q: Which period was Wolf executive and player personnel director with the Oakland Raiders longer for, 1963-1974 or 1979-1989? **A**: 1963-1974	Wolf only had a brief stint with the Jets between 1990 and 1991, while most of his major contributions occurred as an executive and player personnel director with the Oakland Raiders (1963-1974, 1979-1989), and later as General Manager...	**AnswerType**: Passage Span **Answer**: 1979-1989	**AnswerType**: Passage Span **Answer**: 1979-1989	**Top-1 OP**: SPAN **AnswerType**: Passage Span **Answer**: 1963-1974

Table 8: The cases from the development set of DROP. The predictions from the state-of-the-art model NumNet+ and QDGAT are shown. The last column indicates our predicted answers and Top-1 operations.

could compensate for the discrete reasoning capability of language models.

4.7 Case Study

We show two examples from the development set of DROP to illustrate the effectiveness of our model by comparing the results of different models in Table 8. The first example shows that operation is essential for the prediction of answer type. NumNet+ and QDGAT fail to predict the correct answer since the answer type of "how many" questions are wrongly predicted to *Count*. In contrast, OPERA can capture the ADDITION operation, which prompts the model to answer it with an *arithmetic expression* predictor. The second example shows that OPERA has stronger reasoning capability. In the example, though NumNet+ and QDGAT correctly predict the answer type, the final answer is wrong. OPERA can utilize more semantic information for answer prediction with the help of the operation-pivoted discrete reasoning mechanism.

5 Conclusion

We propose a novel framework OPERA for machine reading comprehension requiring discrete reasoning. Lightweight and one-shot operations and heuristic rules to map questions to an operation set are systematically designed. OPERA can leverage the operations to enhance the model's reasoning capability and interpretability. Experiments on DROP and RACENum demonstrate that OPERA achieves remarkable performance. Further visualization and analysis verify its interpretability.

6 Acknowledge

We would like to thank all the anonymous reviewers for their useful feedback. This work is supported by the project of the National Natural Science Foundation of China (No.U1908216) and the National Key Research and Development Program of China (No. 2020AAA0108600).

References

Daniel Andor, Luheng He, Kenton Lee, and Emily Pitler. 2019. Giving BERT a calculator: Finding operations and arguments with reading comprehension. 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)*, pages 5947–5952, Hong Kong, China. Association for Computational Linguistics.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Neural module networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 39–48. IEEE Computer Society.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *ArXiv preprint*, abs/1607.06450.

Junwei Bao, Nan Duan, Zhao Yan, Ming Zhou, and Tiejun Zhao. 2016. Constraint-based question answering with knowledge graph. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2503–2514, Osaka, Japan. The COLING 2016 Organizing Committee.

Kunlong Chen, Weidi Xu, Xingyi Cheng, Zou Xiaochuan, Yuyu Zhang, Le Song, Taifeng Wang, Yuan Qi, and Wei Chu. 2020a. Question directed graph attention network for numerical reasoning over text. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6759–6768, Online. Association for Computational Linguistics.

Xinyun Chen, Chen Liang, Adams Wei Yu, Denny Zhou, Dawn Song, and Quoc V. Le. 2020b. 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.

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.

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, Volume 1 (Long and Short Papers)*, pages 2368–2378, Minneapolis, Minnesota. Association for Computational Linguistics.

Mor Geva, Ankit Gupta, and Jonathan Berant. 2020. Injecting numerical reasoning skills into language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 946–958, Online. Association for Computational Linguistics.

Nitish Gupta, Kevin Lin, Dan Roth, Sameer Singh, and Matt Gardner. 2020. Neural module networks for reasoning over text. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Dan Hendrycks and Kevin Gimpel. 2016. Bridging non-linearities and stochastic regularizers with gaussian error linear units.

Minghao Hu, Yuxing Peng, Zhen Huang, and Dongsheng Li. 2019. A multi-type multi-span network for reading comprehension that requires discrete reasoning. 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)*, pages 1596–1606, Hong Kong, China. 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*.

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

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *ArXiv preprint*, abs/1907.11692.

Qiu Ran, Yankai Lin, Peng Li, Jie Zhou, and Zhiyuan Liu. 2019. NumNet: Machine reading comprehension with numerical reasoning. 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)*, pages 2474–2484, Hong Kong, China. Association for Computational Linguistics.

Elad Segal, Avia Efrat, Mor Shoham, Amir Globerson, and Jonathan Berant. 2020. A simple and effective model for answering multi-span questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3074–3080, Online. Association for Computational Linguistics.

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.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William 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*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Jing Zhao, Junwei Bao, Yifan Wang, Yongwei Zhou, Youzheng Wu, Xiaodong He, and Bowen Zhou. 2021. RoR: Read-over-read for long document machine reading comprehension. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1862–1872, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yongwei Zhou, Junwei Bao, Haipeng Sun, Jiahui Liang, Youzheng Wu, Xiaodong He, Bowen Zhou, and Tiejun Zhao. 2021. Evidr: Evidence-emphasized discrete reasoning for reasoning machine reading comprehension. *ArXiv preprint*, abs/2108.07994.

A Appendix

A.1 Template-Based Heuristic Rules

Operations / Examples / Templates
ADDITION/DIFF *How many more yards was Kris Browns's first field goal over his second?* How many [Slot] more/less [Slot] over [Slot]?
MAX/MIN *How many yards was the longest field goal in the game?* how many yards [Slot] longest/shortest [Slot]?
ARGMAX/ARGMIN *Which player had the longest touchdown reception?* Which player [Slot] longest/shortest [Slot]?
ARGMORE/ARGLESS *Who scored more field goals, David Akers or John Potter?* Who [Slot] more/less, [Slot] or [Slot]?
COUNT *How many field goals did Kris Brown kick?* How many field goals [slot]?
KEY_VALUE *How many percent of Forth Worth households owned a car?* How many percent of [Slot]
SPAN *Which team scored the final TD of the game?* Which team [Slot]

Table 9: All the operations, the corresponding examples and templates.

In this section, we propose some general template-based heuristic rules to illustrate mapping from questions to operations. For example, to detect the operations intimated by the question "*how many yards was the longest field goals*" in Table 9, we design a question template OP_{pat} "*how many yards [Slot] longest/shortest [Slot]*" which maps matched questions to the operation MAX. Meanwhile, we humanly evaluate the quality of the heuristic rules. Specifically, we sample 50 instances from the training set and ask three annotators to label the required operations for each question manually. The final average F1 score of the three annotators is 86%, which indicates that the heuristic rules can correctly predict most of the operations, while still 14% to be noise for training.

A.2 Distribution of the Operations

We analysis the distribution of operations in the training set of DROP, where ADDITION, DIFF and SPAN together accounts for more than 85%. For other questions with span answers that requires sorting or comparison, some specific operations are involved, such as ARGMAX / ARGMIN / ARGMORE / ARGLESS and KEY_VALUE.

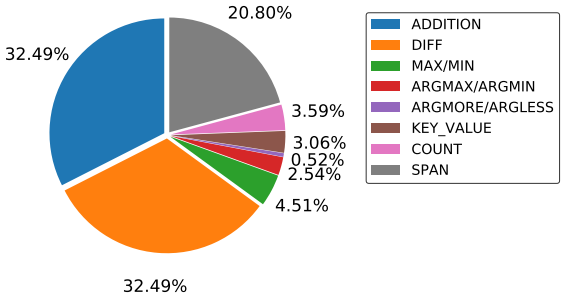

Figure 4: The Distribution of operations in the training set of DROP.

A.3 Details of Prediction Module

In this section, we reveal the architecture details of the prediction module, including a prediction module for answer type and five label predictors corresponding to different answer types. $\texttt{FFN}(\cdot)$ means a feed-forward network that consists of two linear projections with a GeLU activation function (Hendrycks and Gimpel, 2016) and a layer normalization mechanism (Ba et al., 2016).

Answer Type The probability distribution of answer type choices $p(T|Q,P,O)$ is derived by a $|\mathcal{T}|$-classifier with $\mathbf{h}^Q$, $\mathbf{h}^P$ and $\mathbf{h}^E$ as input:

$$\begin{aligned} \mathbf{h}^E &= \sum_{O_i \in \mathcal{OP}} p(O_i|Q,P)\mathbf{E}_i^{OP}, \\ p(T|Q,P,O) &\propto \texttt{FFN}([\mathbf{h}^E;\mathbf{h}^Q;\mathbf{h}^P]), \end{aligned} \tag{11}$$

where $\mathbf{h}^Q$ and $\mathbf{h}^P \in \mathbb{R}^{d_h}$ is the representation vector of question and passage calculated by weighted pooling with $\mathbf{H}^Q$ and $\mathbf{H}^P$, respectively. $\mathbf{E}^{OP}$ is the embedding matrix of operations.

Question/Passage Span Following MTMSN (Hu et al., 2019), we use a question-aware decoding strategy to predict the start and end indices of the answer span. Specifically, we first compute a question representation vector $\mathbf{g}^Q$ via weighted pooling. Then derive the probabilities of the start and end indices of the answer span denoted as p_s and p_e:

$$\begin{aligned} \mathbf{M} &= [\mathbf{h}^{OP};\mathbf{H};\mathbf{H}\odot\mathbf{g}^Q], \\ p_s, p_e &\propto \texttt{FFN}(\mathbf{M}), \end{aligned} \tag{12}$$

where $\odot$ denotes element-wised product. $\mathbf{h}^{OP}$ is derived by Eq. 6 and $\mathbf{H}$ is the representation of input sequence from context encoder.

Count The count predictor is a 10-classifier with the operation-aware representation, all the mentioned number representation, question and passage representations as input. Specifically, when N numbers exists, we gather the representation of all numbers $\mathbf{U} = (\mathbf{u}^1, \mathbf{u}^2, ..., \mathbf{u}^N) \in \mathbb{R}^{N \times d_h}$ from $\mathbf{H}$ and compute a global representation vector of numbers as $\mathbf{h}^U$. Then compute the probability distribution of count answer p_c:

$$\begin{aligned} \alpha^U &\propto \mathbf{U}\mathbf{W}^U, \;\; \mathbf{h}^U = \alpha^U \mathbf{U}, \\ p_c &\propto \texttt{FFN}([\mathbf{h}^{OP}; \mathbf{h}^U; \mathbf{h}^Q; \mathbf{h}^P]), \end{aligned} \tag{13}$$

Arithmetic Expression Similar to NAQANet (Dua et al., 2019), we perform addition and subtraction over all the numbers mentioned in the context by assigning a sign (plus, minus, or zero) to each number. The probability p^i_{sign} of the i-th number's sign is derived as below:

$$p^i_{sign} \propto \texttt{FFN}([\mathbf{h}^{OP}; \mathbf{u}^i; \mathbf{h}^Q; \mathbf{h}^P]). \tag{14}$$

Multi-Spans Inspired by Segal et al. (2020), the multi-span answer is derived with a sequence role labeling method over the input token sequence, denoted as $\texttt{SRL}(\cdot)$. p_{ms} is the probability distribution of token's `BIO` tag:

$$p_{ms} = \texttt{SRL}([\mathbf{H}; \mathbf{h}^{OP}]). \tag{15}$$