A Multi-Type Multi-Span Network for Reading Comprehension that Requires Discrete Reasoning

Minghao Hu, Yuxing Peng, Zhen Huang, Dongsheng Li
National University of Defense Technology, Changsha, China
{huminghao09,pengyuxing,huangzhen,dsli}@nudt.edu.cn

Abstract

Rapid progress has been made in the field of reading comprehension and question answering, where several systems have achieved human parity in some simplified settings. However, the performance of these models degrades significantly when they are applied to more realistic scenarios, where answers are involved with various types, multiple text strings are correct answers, or discrete reasoning abilities are required. In this paper, we introduce the Multi-Type Multi-Span Network (MTMSN), a neural reading comprehension model that combines a multi-type answer predictor designed to support various answer types (e.g., span, count, negation, and arithmetic expression) with a multi-span extraction method for dynamically producing one or multiple text spans. In addition, an arithmetic expression reranking mechanism is proposed to rank expression candidates for further confirming the prediction. Experiments show that our model achieves 79.9 F1 on the DROP hidden test set, creating new state-of-the-art results. Source code[1] is released to facilitate future work.

1 Introduction

This paper considers the reading comprehension task in which some *discrete-reasoning* abilities are needed to correctly answer questions. Specifically, we focus on a new reading comprehension dataset called DROP (Dua et al., 2019), which requires Discrete Reasoning Over the content of Paragraphs to obtain the final answer. Unlike previous benchmarks such as CNN/DM (Hermann et al., 2015) and SQuAD (Rajpurkar et al., 2016) that have been well solved (Chen et al., 2016; Devlin et al., 2019), DROP is substantially more challenging in three ways. First, the answers to the questions involve a wide range of types such as numbers, dates, or text strings. Therefore, various kinds of prediction strategies are required to successfully find the answers. Second, rather than restricting the answer to be a span of text, DROP loosens the constraint so that answers may be a set of multiple text strings. Third, for questions that require discrete reasoning, a system must have a more comprehensive understanding of the context and be able to perform numerical operations such as addition, counting, or sorting.

Existing approaches, when applied to this more realistic scenario, have three problems. First, to produce various answer types, Dua et al. (2019) extend previous one-type answer prediction (Seo et al., 2017) to multi-type prediction that supports span extraction, counting, and addition/subtraction. However, they have not fully considered all potential types. Take the question "*What percent are not non-families?*" and the passage snippet "*39.9% were non-families*" as an example, a *negation* operation is required to infer the answer. Second, previous reading comprehension models (Wang et al., 2017; Yu et al., 2018; Hu et al., 2018) are designed to produce one single span as the answer. But for some questions such as "*Which ancestral groups are smaller than 11%?*", there may exist several spans as correct answers (e.g., "*Italian*", "*English*", and "*Polish*"), which can not be well handled by these works. Third, to support numerical reasoning, prior work (Dua et al., 2019) learns to predict signed numbers for obtaining an arithmetic expression that can be executed by a symbolic system. Nevertheless, the prediction of each signed number is isolated, and the expression's context information has not been considered. As a result, obviously-wrong expressions, such as all predicted signs are either minus or zero, are likely produced.

To address the above issues, we introduce

[1] https://github.com/huminghao16/MTMSN

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 1596–1606, Hong Kong, China, November 3–7, 2019.

the Multi-Type Multi-Span Network (MTMSN), a neural reading comprehension model for predicting various types of answers as well as dynamically extracting one or multiple spans. MTMSN utilizes a series of pre-trained Transformer blocks (Devlin et al., 2019) to obtain a deep bidirectional context representation. On top of it, a multi-type answer predictor is proposed to not only support previous prediction strategies such as span, count number, and arithmetic expression, but also add a new type of logical negation. This results in a wider range of coverage of answer types, which turns out to be crucial to performance. Besides, rather than always producing one single span, we present a multi-span extraction method to produce multiple answers. The model first predicts the number of answers, and then extracts non-overlapped spans to the specific amount. In this way, the model can learn to dynamically extract one or multiple spans, thus being beneficial for multi-answer cases. In addition, we propose an arithmetic expression reranking mechanism to rank expression candidates that are decoded by beam search, so that their context information can be considered during reranking to further confirm the prediction.

Our MTMSN model outperforms all existing approaches on the DROP hidden test set by achieving 79.9 F1 score, a 32.9% absolute gain over prior best work at the time of submission. To make a fair comparison, we also construct a baseline that uses the same BERT-based encoder. Again, MTMSN surpasses it by obtaining a 13.2 F1 increase on the development set. We also provide an in-depth ablation study to show the effectiveness of our proposed methods, analyze performance breakdown by different answer types, and give some qualitative examples as well as error analysis.

2 Task Description

In the reading comprehension task that requires discrete reasoning, a passage and a question are given. The goal is to predict an answer to the question by reading and understanding the passage. Unlike previous dataset such as SQuAD (Rajpurkar et al., 2016) where the answer is limited to be a single span of text, DROP loosens the constraint so that the answer involves various types such as number, date, or span of text (Figure 1). Moreover, the answer can be multiple text strings instead of single continuous span ($\mathbf{A}_2$). To successfully find the answer, some discrete reasoning abilities, such as sorting ($\mathbf{A}_1$), subtraction ($\mathbf{A}_3$), and negation ($\mathbf{A}_4$), are required.

Passage: As of the census of 2000, there were 218,590 people, 79,667 households, ... 22.5% were of German people, 13.1% Irish people, 9.8% Italian people, ...
$\mathbf{Q}_1$: Which group from the census is larger: German or Irish?
$\mathbf{A}_1$: German
$\mathbf{Q}_2$: Which ancestral groups are at least 10%?
$\mathbf{A}_2$: German, Irish
$\mathbf{Q}_3$: How many more people are there than households?
$\mathbf{A}_3$: 138,923
$\mathbf{Q}_4$: How many percent were not German?
$\mathbf{A}_4$: 77.5

Figure 1: Question-answer pairs along with a passage from the DROP dataset.

3 Our Approach

Figure 2 gives an overview of our model that aims to combine neural reading comprehension with numerical reasoning. Our model uses BERT (Devlin et al., 2019) as encoder: we map word embeddings into contextualized representations using pre-trained Transformer blocks (Vaswani et al., 2017) (§3.1). Based on the representations, we employ a multi-type answer predictor that is able to produce four answer types: (1) span from the text; (2) arithmetic expression; (3) count number; (4) negation on numbers (§3.2). Following Dua et al. (2019), we first predict the answer type of a given passage-question pair, and then adopt individual prediction strategies. To support multi-span extraction (§3.3), the model explicitly predicts the number of answer spans. It then outputs non-overlapped spans until the specific amount is reached. Moreover, we do not directly use the arithmetic expression that possesses the maximum probability, but instead re-rank several expression candidates that are decoded by beam search to further confirm the prediction (§3.4). Finally, the model is trained under weakly-supervised signals to maximize the marginal likelihood over all possible annotations (§3.5).

3.1 BERT-Based Encoder

To obtain a universal representation for both the question and the passage, we utilize BERT (Devlin et al., 2019), a pre-trained deep bidirectional Transformer model that achieves state-of-the-art performance across various tasks, as the encoder.

Specifically, we first tokenize the question and

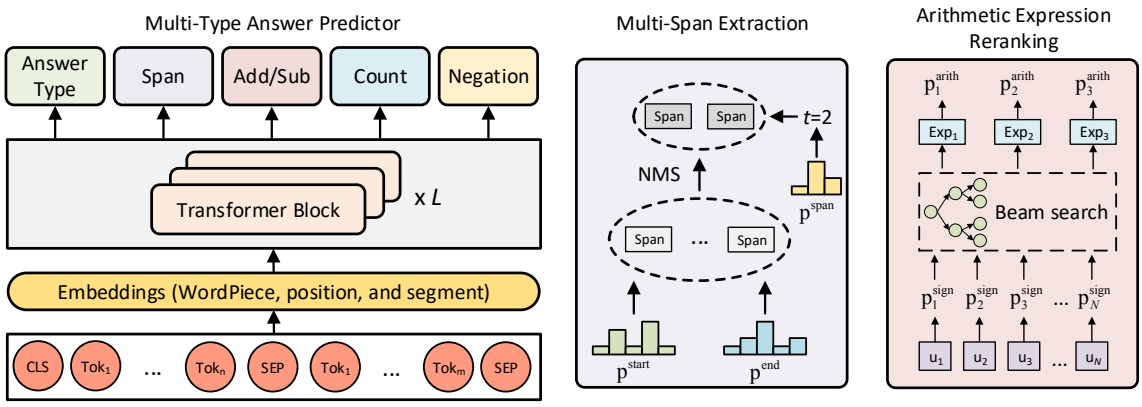

Figure 2: An illustration of MTMSN architecture. The multi-type answer predictor supports four kinds of answer types including span, addition/subtraction, count, and negation. A multi-span extraction method is proposed to dynamically produce one or several spans. The arithmetic expression reranking mechanism aims to rank expression candidates that are decoded by beam search for further validating the prediction.

the passage using the WordPiece vocabulary (Wu et al., 2016), and then generate the input sequence by concatenating a `[CLS]` token, the tokenized question, a `[SEP]` token, the tokenized passage, and a final `[SEP]` token. For each token in the sequence, its input representation is the element-wise addition of WordPiece embeddings, positional embeddings, and segment embeddings (Devlin et al., 2019). As a result, a list of input embeddings $\mathbf{H}_0 \in \mathbb{R}^{T \times D}$ can be obtained, where D is the hidden size and T is the sequence length. A series of L pre-trained Transformer blocks are then used to project the input embeddings into contextualized representations $\mathbf{H}_i$ as:

$$\mathbf{H}_i = \text{TransformerBlock}(\mathbf{H}_{i-1}), \forall i \in [1, L]$$

Here, we omit a detailed introduction of the block architecture and refer readers to Vaswani et al. (2017) for more details.

3.2 Multi-Type Answer Predictor

Rather than restricting the answer to always be a span of text, the discrete-reasoning reading comprehension task involves different answer types (e.g., number, date, span of text). Following Dua et al. (2019), we design a multi-type answer predictor to selectively produce different kinds of answers such as span, count number, and arithmetic expression. To further increase answer coverage, we propose adding a new answer type to support logical negation. Moreover, unlike prior work that separately predicts passage spans and question spans, our approach directly extracts spans from the input sequence.

Answer type prediction Inspired by the Augmented QANet model (Dua et al., 2019), we use the contextualized token representations from the last four blocks ($\mathbf{H}_{L-3}$, ..., $\mathbf{H}_L$) as the inputs to our answer predictor, which are denoted as $\mathbf{M}_0$, $\mathbf{M}_1$, $\mathbf{M}_2$, $\mathbf{M}_3$, respectively. To predict the answer type, we first split the representation $\mathbf{M}_2$ into a question representation $\mathbf{Q}_2$ and a passage representation $\mathbf{P}_2$ according to the index of intermediate `[SEP]` token. Then the model computes two vectors $\mathbf{h}^{\mathbf{Q}_2}$ and $\mathbf{h}^{\mathbf{P}_2}$ that summarize the question and passage information respectively:

$$\boldsymbol{\alpha}^Q = \text{softmax}(\mathbf{W}^Q \mathbf{Q}_2), \quad \mathbf{h}^{\mathbf{Q}_2} = \boldsymbol{\alpha}^Q \mathbf{Q}_2$$

where $\mathbf{h}^{\mathbf{P}_2}$ is computed in a similar way over $\mathbf{P}_2$.

Next, we calculate a probability distribution to represent the choices of different answer types as:

$$\mathbf{p}^{\text{type}} = \text{softmax}(\text{FFN}([\mathbf{h}^{\mathbf{Q}_2}; \mathbf{h}^{\mathbf{P}_2}; \mathbf{h}^{\text{CLS}}]))$$

Here, $\mathbf{h}^{\text{CLS}}$ is the first vector in the final contextualized representation $\mathbf{M}_3$, and FFN denotes a feed-forward network consisting of two linear projections with a GeLU activation (Hendrycks and Gimpel, 2016) followed by a layer normalization (Lei Ba et al., 2016) in between.

Span To extract the answer either from the passage or from the question, we combine the gating mechanism of Wang et al. (2017) with the standard decoding strategy of Seo et al. (2017) to predict the starting and ending positions across the entire sequence. Specifically, we first compute three vectors, namely $\mathbf{g}^{\mathbf{Q}_0}$, $\mathbf{g}^{\mathbf{Q}_1}$, $\mathbf{g}^{\mathbf{Q}_2}$, that summarize

the question information among different levels of question representations:

$$\boldsymbol{\beta}^{Q} = \text{softmax}(\text{FFN}(\mathbf{Q}_2), \quad \mathbf{g}^{\mathbf{Q}_2} = \boldsymbol{\beta}^{Q}\mathbf{Q}_2$$

where $\mathbf{g}^{\mathbf{Q}_0}$ and $\mathbf{g}^{\mathbf{Q}_1}$ are computed over $\mathbf{Q}_0$ and $\mathbf{Q}_1$ respectively, in a similar way as described above.

Then we compute the probabilities of the starting and ending indices of the answer span from the input sequence as:

$$\begin{aligned}
\bar{\mathbf{M}}^{\text{start}} &= [\mathbf{M}_2; \mathbf{M}_0; \mathbf{g}^{\mathbf{Q}_2} \otimes \mathbf{M}_2; \mathbf{g}^{\mathbf{Q}_0} \otimes \mathbf{M}_0], \\
\bar{\mathbf{M}}^{\text{end}} &= [\mathbf{M}_2; \mathbf{M}_1; \mathbf{g}^{\mathbf{Q}_2} \otimes \mathbf{M}_2; \mathbf{g}^{\mathbf{Q}_1} \otimes \mathbf{M}_1], \\
\mathbf{p}^{\text{start}} &= \text{softmax}(\mathbf{W}^{S}\bar{\mathbf{M}}^{\text{start}}), \\
\mathbf{p}^{\text{end}} &= \text{softmax}(\mathbf{W}^{E}\bar{\mathbf{M}}^{\text{end}})
\end{aligned}$$

where $\otimes$ denotes the outer product between the vector $\mathbf{g}$ and each token representation in $\mathbf{M}$.

Arithmetic expression In order to model the process of performing addition or subtraction among multiple numbers mentioned in the passage, we assign a three-way categorical variable (plus, minus, or zero) for each number to indicate its sign, similar to Dua et al. (2019). As a result, an arithmetic expression that has a number as the final answer can be obtained and easily evaluated.

Specifically, for each number mentioned in the passage, we gather its corresponding representation from the concatenation of $\mathbf{M}_2$ and $\mathbf{M}_3$, eventually yielding $\mathbf{U} = (\mathbf{u}_1, ..., \mathbf{u}_N) \in \mathbb{R}^{N \times 2*D}$ where N numbers exist. Then the probabilities of the i-th number being assigned a plus, minus or zero is computed as:

$$\mathbf{p}_i^{\text{sign}} = \text{softmax}(\text{FFN}([\mathbf{u}_i; \mathbf{h}^{\mathbf{Q}_2}; \mathbf{h}^{\mathbf{P}_2}; \mathbf{h}^{\text{CLS}}]))$$

Count We consider the ability of counting entities and model it as a multi-class classification problem. To achieve this, the model first produces a vector $\mathbf{h}^{\mathbf{U}}$ that summarizes the important information among all mentioned numbers, and then computes a counting probability distribution as:

$$\begin{aligned}
\boldsymbol{\alpha}^{U} &= \text{softmax}(\mathbf{W}^{U}\mathbf{U}), \quad \mathbf{h}^{\mathbf{U}} = \boldsymbol{\alpha}^{U}\mathbf{U}, \\
\mathbf{p}^{\text{count}} &= \text{softmax}(\text{FFN}([\mathbf{h}^{\mathbf{U}}; \mathbf{h}^{\mathbf{Q}_2}; \mathbf{h}^{\mathbf{P}_2}; \mathbf{h}^{\text{CLS}}]))
\end{aligned}$$

Negation One obvious but important linguistic phenomenon that prior work fails to capture is *negation*. We find there are many cases in DROP that require to perform logical negation on numbers. The question ($\mathbf{Q}_4$) in Figure 1 gives a qualitative example of this phenomenon. To model this phenomenon, we assign a two-way categorical variable for each number to indicate whether a negation operation should be performed. Then we compute the probabilities of logical negation on the i-th number as:

$$\mathbf{p}_i^{\text{negation}} = \text{softmax}(\text{FFN}([\mathbf{u}_i; \mathbf{h}^{\mathbf{Q}_2}; \mathbf{h}^{\mathbf{P}_2}; \mathbf{h}^{\text{CLS}}]))$$

3.3 Multi-Span Extraction

Although existing reading comprehension tasks focus exclusively on finding one span of text as the final answer, DROP loosens the restriction so that the answer to the question may be several text spans. Therefore, specific adaption should be made to extend previous single-span extraction to multi-span scenario.

To do this, we propose directly predicting the number of spans and model it as a classification problem. This is achieved by computing a probability distribution on span amount as

$$\mathbf{p}^{\text{span}} = \text{softmax}(\text{FFN}([\mathbf{h}^{\mathbf{Q}_2}; \mathbf{h}^{\mathbf{P}_2}; \mathbf{h}^{\text{CLS}}]))$$

To extract non-overlapped spans to the specific amount, we adopt the non-maximum suppression (NMS) algorithm (Rosenfeld and Thurston, 1971) that is widely used in computer vision for pruning redundant bounding boxes, as shown in Algorithm 1. Concretely, the model first proposes a set of top-K spans $\mathbf{S}$ according to the descending order of the span score, which is computed as $\mathbf{p}_k^{\text{start}}\mathbf{p}_l^{\text{end}}$ for the span (k, l). It also predicts the amount of extracted spans t from $\mathbf{p}^{\text{span}}$, and initializes a new set $\tilde{\mathbf{S}}$. Next, we add the span $\mathbf{s}_i$ that possesses the maximum span score to the set $\tilde{\mathbf{S}}$, and remove it from $\mathbf{S}$. We also delete any remaining span $\mathbf{s}_j$ that overlaps with $\mathbf{s}_i$, where the degree of overlap is measured using the text-level F1 function. This process is repeated for remaining spans in $\mathbf{S}$, until $\mathbf{S}$ is empty or the size of $\tilde{\mathbf{S}}$ reaches t.

3.4 Arithmetic Expression Reranking

As discussed in §3.2, we model the phenomenon of discrete reasoning on numbers by learning to predict a plus, minus, or zero for each number in the passage. In this way, an arithmetic expression composed of signed numbers can be obtained, where the final answer can be deduced by performing simple arithmetic computation. However, since the sign of each number is only determined by the number representation and some coarse-grained global representations, the context information of the expression itself has not been considered. As a result, the model may predict some

Algorithm 1 Multi-span extraction

Input: $\mathbf{p}^{\text{start}}$; $\mathbf{p}^{\text{end}}$; $\mathbf{p}^{\text{span}}$

Generate the set $\mathbf{S}$ by extracting top-K spans
Sort $\mathbf{S}$ in descending order of span scores
$t = \arg\max \mathbf{p}^{\text{span}} + 1$
Initialize $\tilde{\mathbf{S}} = \{\}$
while $\mathbf{S} \neq \{\}$ and $|\tilde{\mathbf{S}}| < t$ **do**
    for $\mathbf{s}_i$ in $\mathbf{S}$ **do**
        Add span $\mathbf{s}_i$ to $\tilde{\mathbf{S}}$
        Remove span $\mathbf{s}_i$ from $\mathbf{S}$
        for $\mathbf{s}_j$ in $\mathbf{S}$ **do**
            if $\text{f1}(\mathbf{s}_i, \mathbf{s}_j) > 0$ **then**
                Remove span $\mathbf{s}_j$ from $\mathbf{S}$
return $\tilde{\mathbf{S}}$

obviously wrong expressions (e.g., the signs that have maximum probabilities are either minus or zero, resulting in a large negative value). Therefore, in order to further validate the prediction, it is necessary to rank several highly confident expression candidates using the representation summarized from the expression's context.

Specifically, we use beam search to produce top-ranked arithmetic expressions, which are sent back to the network for reranking. Since each expression consists of several signed numbers, we construct an expression representation by taking both the numbers and the signs into account. For each number in the expression, we gather its corresponding vector from the representation $\mathbf{U}$. As for the signs, we initialize an embedding matrix $\mathbf{E} \in \mathbb{R}^{3\times 2*D}$, and find the sign embeddings for each signed number. In this way, given the i-th expression that contains M signed numbers at most, we can obtain number vectors $\mathbf{V}_i \in \mathbb{R}^{M\times 2*D}$ as well as sign embeddings $\mathbf{C}_i \in \mathbb{R}^{M\times 2*D}$. Then the expression representation along with the reranking probability can be calculated as:

$$\boldsymbol{\alpha}_i^V = \text{softmax}(\mathbf{W}^V(\mathbf{V}_i + \mathbf{C}_i)),$$
$$\mathbf{h}_i^{\mathbf{V}} = \boldsymbol{\alpha}_i^V(\mathbf{V}_i + \mathbf{C}_i),$$
$$\mathbf{p}_i^{\text{arith}} = \text{softmax}(\text{FFN}([\mathbf{h}_i^{\mathbf{V}}; \mathbf{h}^{\mathbf{Q}_2}; \mathbf{h}^{\mathbf{P}_2}; \mathbf{h}^{\text{CLS}}]))$$

3.5 Training and Inference

Since DROP does not indicate the answer type but only provides the answer string, we therefore adopt the weakly supervised annotation scheme, as suggested in Berant et al. (2013); Dua et al. (2019). We find all possible annotations that point to the gold answer, including matching spans, arithmetic expressions, correct count numbers, negation operations, and the number of spans. We use simple rules to search over all mentioned numbers to find potential negations. That is, if 100 minus a number is equal to the answer, then a negation occurs on this number. Besides, we only search the addition/subtraction of three numbers at most due to the exponential search space.

To train our model, we propose using a two-step training method composed of an inference step and a training step. In the first step, we use the model to predict the probabilities of sign assignments for numbers. If there exists any annotation of arithmetic expressions, we run beam search to produce expression candidates and label them as either correct or wrong, which are later used for supervising the reranking component. In the second step, we adopt the marginal likelihood objective function (Clark and Gardner, 2018), which sums over the probabilities of all possible annotations including the above labeled expressions. Notice that there are two objective functions for the multi-span component: one is a distantly-supervised loss that maximizes the probabilities of all matching spans, and the other is a classification loss that maximizes the probability on span amount.

At test time, the model first chooses the answer type and then performs specific prediction strategies. For the span type, we use Algorithm 1 for decoding. If the type is addition/subtraction, arithmetic expression candidates will be proposed and further reranked. The expression with the maximum product of cumulative sign probability and reranking probability is chosen. As for the counting type, we choose the number that has the maximum counting probability. Finally, if the type is negation, we find the number that possesses the largest negation probability, and then output the answer as 100 minus this number.

4 Experiments

4.1 Implementation Details

Dataset We consider the reading comprehension benchmark that requires Discrete Reasoning Over Paragraphs (DROP) (Dua et al., 2019) to train and evaluate our model. DROP contains crowdsourced, adversarially-created, 96.6K question-answer pairs, with 77.4K for training, 9.5K for validation, and another 9.6K hidden examples for testing. Passages are extracted from Wikipedia articles and the answer to each question involves various types such as number, date, or text string. Some answers may even be a set of multiple spans of text in the passage. To find the answers, a com-

Model	Dev		Test	
	EM	F1	EM	F1
Heuristic Baseline (Dua et al., 2019)	4.28	8.07	4.18	8.59
Semantic Role Labeling (Carreras and Màrquez, 2004)	11.03	13.67	10.87	13.35
BiDAF (Seo et al., 2017)	26.06	28.85	24.75	27.49
QANet+ELMo (Yu et al., 2018)	27.71	30.33	27.08	29.67
$\text{BERT}_{\text{BASE}}$ (Devlin et al., 2019)	30.10	33.36	29.45	32.70
NAQANet (Dua et al., 2019)	46.20	49.24	44.07	47.01
$\text{NABERT}_{\text{BASE}}$	55.82	58.75	-	-
$\text{NABERT}_{\text{LARGE}}$	64.61	67.35	-	-
$\text{MTMSN}_{\text{BASE}}$	68.17	72.81	-	-
$\text{MTMSN}_{\text{LARGE}}$	**76.68**	**80.54**	**75.85**	**79.88**
Human Performance (Dua et al., 2019)	-	-	92.38	95.98

Table 1: The performance of MTMSN and other competing approaches on DROP dev and test set.

Model	BASE		LARGE	
	EM	F1	EM	F1
MTMSN	68.2	72.8	76.7	80.5
w/o Add/Sub	46.7	51.3	53.8	58.0
w/o Count	62.5	66.4	71.8	75.6
w/o Negation	59.4	63.6	67.2	70.9
w/o Multi-Span	67.5	70.7	75.6	78.4
w/o Reranking	66.9	71.2	74.9	78.7

Table 2: Ablation tests of base and large models on the DROP dev set.

Model	EM	F1
MTMSN	76.7	80.5
w/o Q/P Vectors	75.1	79.2
w/o CLS Vector	74.0	78.4
Q/P Vectors Using Last Hidden	76.5	80.2
w/o Gated Span Prediction	75.8	79.7
Combine Add/Sub with Negation	75.5	79.4

Table 3: Ablation tests of different architecture choices using $\text{MTMSN}_{\text{LARGE}}$.

prehensive understanding of the context as well as the ability of numerical reasoning are required.

Model settings We build our model upon two publicly available uncased versions of BERT: $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$[2], and refer readers to Devlin et al. (2019) for details on model sizes. We use Adam optimizer with a learning rate of 3e-5 and warmup over the first 5% steps to train. The maximum number of epochs is set to 10 for base models and 5 for large models, while the batch size is 12 or 24 respectively. A dropout probability of 0.1 is used unless stated otherwise. The number of counting class is set to 10, and the maximum number of spans is 8. The beam size is 3 by default, while the maximum amount of signed numbers M is set to 4. All texts are tokenized using WordPiece vocabulary (Wu et al., 2016), and truncated to sequences no longer than 512 tokens.

Baselines Following the implementation of Augmented QANet (NAQANet) (Dua et al., 2019), we introduce a similar baseline called Augmented BERT (NABERT). The main difference is that we replace the encoder of QANet (Yu et al., 2018) with the pre-trained Transformer blocks (Devlin et al., 2019). Moreover, it also supports the prediction of various answer types such as span, arithmetic expression, and count number.

4.2 Main Results

Two metrics, namely Exact Match (EM) and F1 score, are utilized to evaluate models. We use the official script to compute these scores. Since the test set is hidden, we only submit the best single model to obtain test results.

Table 1 shows the performance of our model and other competitive approaches on the develop-

[2] $\text{BERT}_{\text{BASE}}$ is the original version while $\text{BERT}_{\text{LARGE}}$ is the model augmented with n-gram masking and synthetic self-training: https://github.com/google-research/bert.

Type	(%)	NABERT		MTMSN	
		EM	F1	EM	F1
Date	1.6	55.7	60.8	55.7	69.0
Number	61.9	63.8	64.0	80.9	81.1
Single Span	31.7	75.9	80.6	77.5	82.8
Multi Span	4.8	0	22.7	25.1	62.8

Table 4: Performance breakdown of $\text{NABERT}_{\text{LARGE}}$ and $\text{MTMSN}_{\text{LARGE}}$ by gold answer types.

Type	NABERT			MTMSN		
	(%)	EM	F1	(%)	EM	F1
Span	43.0	67.9	74.2	42.7	72.2	81.0
Add/Sub	43.6	62.0	62.1	32.4	78.1	78.2
Count	13.4	62.4	62.4	13.4	70.4	70.4
Negation	0	0	0	11.5	96.3	96.3

Table 5: Performance breakdown of $\text{NABERT}_{\text{LARGE}}$ and $\text{MTMSN}_{\text{LARGE}}$ by predicted answer types.

ment and test sets. MTMSN outperforms all existing approaches by a large margin, and creates new state-of-the-art results by achieving an EM score of 75.85 and a F1 score of 79.88 on the test set. Since our best model utilizes $\text{BERT}_{\text{LARGE}}$ as encoder, we therefore compare $\text{MTMSN}_{\text{LARGE}}$ with the $\text{NABERT}_{\text{LARGE}}$ baseline. As we can see, our model obtains 12.07/13.19 absolute gain of EM/F1 over the baseline, demonstrating the effectiveness of our approach. However, as the human achieves 95.98 F1 on the test set, our results suggest that there is still room for improvement.

4.3 Ablation Study

Component ablation To analyze the effect of the proposed components, we conduct ablation studies on the development set. As illustrated in Table 2, the use of addition and subtraction is extremely crucial: the EM/F1 performance of both the base and large models drop drastically by more than 20 points if it is removed. Predicting count numbers is also an important component that contributes nearly 5% gain on both metrics. Moreover, enhancing the model with the negation type significantly increases the F1 by roughly 9 percent on both models. In brief, the above results show that multi-type answer prediction is vitally important for handling different forms of answers, especially in cases where discrete reasoning abilities are required.

We also report the performance after removing the multi-span extraction method. The results reveal that it has a more negative impact on the F1 score. We interpret this phenomenon as follows: producing multiple spans that are partially matched with ground-truth answers is much easier than generating an exactly-matched set of multiple answers. Hence for multi-span scenarios, the gain of our method on F1 is relatively easier to obtain than the one on EM. Finally, to ablate arithmetic expression reranking, we simply use the arithmetic expression that has the maximum cumulative sign probability instead. We find that our reranking mechanism gives 1.8% gain on both metrics for the large model. This confirms that validating expression candidates with their context information is beneficial for filtering out highly-confident but wrong predictions.

Architecture ablation We further conduct a detailed ablation in Table 3 to evaluate our architecture designs. First, we investigate the effects of some "global vectors" used in our model. Specifically, we find that removing the question and passage vectors from all involved computation leads to 1.3 % drop on F1. Ablating the representation of `[CLS]` token leads to even worse results. We also try to use the last hidden representation (denoted as $\mathbf{M}_3$) to calculate question and passage vectors, but find that does not work. Next, we remove the gating mechanism used during span prediction, and observe a nearly 0.8% decline on both metrics. Finally, we share parameters between the arithmetic expression component and the negation component, and find the performance drops by 1.1% on F1.

4.4 Analysis and Discussion

Performance breakdown We now provide a quantitative analysis by showing performance breakdown on the development set. Table 4 shows that our gains mainly come from the most frequent number type, which requires various types of symbolic, discrete reasoning operations. Moreover, significant improvements are also obtained in the multi-span category, where the F1 score increases by more than 40 points. This result further proves the validity of our multi-span extraction method.

We also give the performance statistics that are categorized according to the predicted answer types in Table 5. As shown in the Table, the main improvements are due to the addition/subtraction and negation types. We conjecture that there are two reasons for these improvements. First, our

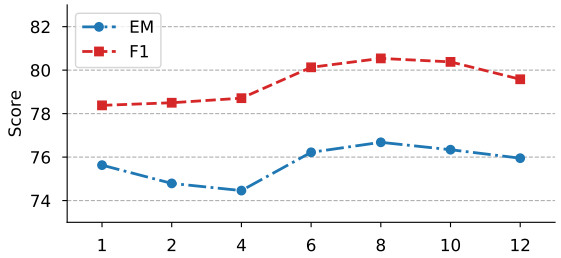

Figure 3: EM/F1 scores of $\text{MTMSN}_{\text{LARGE}}$ with different maximum numbers of spans.

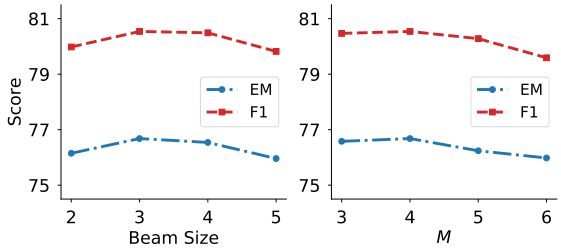

Figure 4: EM/F1 scores of $\text{MTMSN}_{\text{LARGE}}$ with different beam sizes and amounts of signed numbers (M).

proposed expression reranking mechanism helps validate candidate expressions. Second, a new inductive bias that enables the model to perform logical negation has been introduced. The impressive performance on the negation type confirms our judgement, and suggests that the model is able to find most of negation operations. In addition, we also observe promising gains brought by the span and count types. We think the gains are mainly due to the multi-span extraction method as well as architecture designs.

Effect of maximum number of spans To investigate the effect of maximum number of spans on multi-span extraction, we conduct an experiment on the dev set and show the curves in Figure 3. We vary the value from 2 to 12, increased by 2, and also include the extreme value 1. According to the Figure, the best results are obtained at 8. A higher value could potentially increase the answer recall but damage the precision by making more predictions, and a smaller value may force the model to produce limited number of answers, resulting in high precision but low recall. Therefore, a value of 8 turns out to be a good trade-off between recall and precision. Moreover, when the value decreases to 1, the multi-span extraction degrades to previous single-span scenario, and the performance drops significantly.

Configuration	Skipped	Kept	Ratio (%)	F1
Span	33752	43657	56.4	38.9
+ ♣	6384	71025	91.7	59.2
+ ♣ + ♠	4282	73127	94.4	63.6
+ ♣ + ♠ + ♡	1595	75814	97.9	72.8

Table 6: Annotation statistics under different combinations of answer types in the DROP train set. "Kept" and "Skipped" mean the number of examples with or without annotation, respectively. ♣ refers to Add/Sub, ♠ denotes Count, and ♡ indicates Negation. F1 scores are benchmarked using $\text{MTMSN}_{\text{BASE}}$ on the dev set.

Effect of beam size and M We further investigate the effect of beam size and maximum amount of signed numbers in Figure 4. As we can see, a beam size of 3 leads to the best performance, likely because a larger beam size might confuse the model as too many candidates are ranked, on the other hand, a small size could be not sufficient to cover the correct expression. In addition, we find that the performance constantly decreases as the maximum threshold M increases, suggesting that most of expressions only contain two or three signed numbers, and setting a larger threshold could bring in additional distractions.

Annotation statistics We list the annotation statistics on the DROP train set in Table 6. As we can see, only annotating matching spans results in a labeled ratio of 56.4%, indicating that DROP includes various answer types beyond text spans. By further considering the arithmetic expression, the ratio increase sharply to 91.7%, suggesting more than 35% answers need to be inferred with numeral reasoning. Continuing adding counting leads to a percentage of 94.4%, and a final 97.9% coverage is achieved by additionally taking negation into account. More importantly, the F1 score constantly increases as more answer types are considered. This result is consistent with our observations in ablation study.

Error analysis Finally, to better understand the remaining challenges, we randomly sample 100 incorrectly predicted examples based on EM and categorize them into 7 classes. 38% of errors are incorrect arithmetic computations, 18% require sorting over multiple entities, 13% are due to mistakes on multi-span extraction, 10% are single-span extraction problems, 8% involve miscounting, another 8% are wrong predictions on span number, the rest (5%) are due to various reasons

such as incorrect preprocessing, negation error, and so on. See Appendix for some examples of the above error cases.

5 Related Work

Reading comprehension benchmarks Promising advancements have been made for reading comprehension due to the creation of many large datasets. While early research used cloze-style tests (Hermann et al., 2015; Hill et al., 2016), most of recent works (Rajpurkar et al., 2016; Joshi et al., 2017) are designed to extract answers from the passage. Despite their success, these datasets only require shallow pattern matching and simple logical reasoning, thus being well solved (Chen et al., 2016; Devlin et al., 2019). Recently, Dua et al. (2019) released a new benchmark named DROP that demands discrete reasoning as well as deeper paragraph understanding to find the answers. Saxton et al. (2019) introduced a dataset consisting of different types of mathematics problems to focuses on mathematical computation. We choose to work on DROP to test both the numerical reasoning and linguistic comprehension abilities.

Neural reading models Previous neural reading models, such as BiDAF (Seo et al., 2017), R-Net (Wang et al., 2017), QANet (Yu et al., 2018), Reinforced Mreader (Hu et al., 2018), are usually designed to extract a continuous span of text as the answer. Dua et al. (2019) enhanced prior single-type prediction to support various answer types such as span, count number, and addition/subtraction. Different from these approaches, our model additionally supports a new negation type to increase answer coverage, and learns to dynamically extract one or multiple spans. Morevoer, answer reranking has been well studied in several prior works (Cui et al., 2016; Wang et al., 2018a,b,c; Hu et al., 2019). We follow this line of work, but propose ranking arithmetic expressions instead of candidate answers.

End-to-end symbolic reasoning Combining neural methods with symbolic reasoning was considered by Graves et al. (2014); Sukhbaatar et al. (2015), where neural networks augmented with external memory are trained to execute simple programs. Later works on program induction (Reed and De Freitas, 2016; Neelakantan et al., 2016; Liang et al., 2017) extended this idea by using several built-in logic operations along with a key-value memory to learn different types of compositional programs such as addition or sorting. In contrast to these works, MTMSN does not model various types of reasoning with a universal memory mechanism but instead deals each type with individual predicting strategies.

Visual question answering In computer vision community, the most similar work to our approach is Neural Module Networks (Andreas et al., 2016b), where a dependency parser is used to lay out a neural network composed of several pre-defined modules. Later, Andreas et al. (2016a) proposed dynamically choosing an optimal layout structure from a list of layout candidates that are produced by off-the-shelf parsers. Hu et al. (2017) introduced an end-to-end module network that learns to predict instance-specific network layouts without the aid of a parser. Compared to these approaches, MTMSN has a static network layout that can not be changed during training and evaluation, where pre-defined "modules" are used to handle different types of answers.

6 Conclusion

We introduce MTMSN, a multi-type multi-span network for reading comprehension that requires discrete reasoning over the content of paragraphs. We enhance a multi-type answer predictor to support logical negation, propose a multi-span extraction method for producing multiple answers, and design an arithmetic expression reranking mechanism to further confirm the prediction. Our model achieves 79.9 F1 on the DROP hidden test set, creating new state-of-the-art results. As future work, we would like to consider handling additional types such as sorting or multiplication/division. We also plan to explore more advanced methods for performing complex numerical reasoning.

Acknowledgments

We would like to thank the anonymous reviewers for their thoughtful comments and insightful feedback. This work was supported by the National Key Research and Development Program of China (2016YFB100101).

References

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016a. Learning to compose neural net-

works for question answering. In *Proceedings of NAACL*.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016b. Neural module networks. In *Proceedings of CVPR*.

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

Xavier Carreras and Lluís Màrquez. 2004. Introduction to the conll-2004 shared task: Semantic role labeling. In *Proceedings of CONLL*.

Danqi Chen, Jason Bolton, and Christopher D Manning. 2016. A thorough examination of the cnn/daily mail reading comprehension task. *arXiv preprint arXiv:1606.02858*.

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

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2016. Attention-over-attention neural networks for reading comprehension. *arXiv preprint arXiv:1607.04423*.

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

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 NAACL*.

Alex Graves, Greg Wayne, and Ivo Danihelka. 2014. Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *CoRR, abs/1606.08415*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of NIPS*.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2016. The goldilocks principle: Reading childrens books with explicit memory representations. In *Proceedings of ICLR*.

Minghao Hu, Yuxing Peng, Zhen Huang, and Dongsheng Li. 2019. Retrieve, read, rerank: Towards end-to-end multi-document reading comprehension. In *Proceedings of ACL*.

Minghao Hu, Yuxing Peng, Zhen Huang, Xipeng Qiu, Furu Wei, and Ming Zhou. 2018. Reinforced mnemonic reader for machine reading comprehension. In *Proceedings of IJCAI*.

Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. 2017. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of ICCV*.

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

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Chen Liang, Jonathan Berant, Quoc Le, Kenneth D Forbus, and Ni Lao. 2017. Neural symbolic machines: Learning semantic parsers on freebase with weak supervision. In *Proceedings of ACL*.

Arvind Neelakantan, Quoc V Le, and Ilya Sutskever. 2016. Neural programmer: Inducing latent programs with gradient descent. In *Proceedings of ICLR*.

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

Scott Reed and Nando De Freitas. 2016. Neural programmer-interpreters. In *Proceedings of ICLR*.

Azriel Rosenfeld and Mark Thurston. 1971. Edge and curve detection for visual scene analysis. *IEEE Transactions on computers*, (5):562–569.

David Saxton, Edward Grefenstette, Felix Hill, and Pushmeet Kohli. 2019. Analysing mathematical reasoning abilities of neural models. In *Proceedings of ICLR*.

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

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Proceedings of NIPS*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of NIPS*.

Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. 2018a. Evidence aggregation for answer re-ranking in open-domain question answering. In *Proceedings of ICLR*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of ACL*.

Yizhong Wang, Kai Liu, Jing Liu, Wei He, Yajuan Lyu, Hua Wu, Sujian Li, and Haifeng Wang. 2018b. Multi-passage machine reading comprehension with cross-passage answer verification. In *Proceedings of ACL*.

Zhen Wang, Jiachen Liu, Xinyan Xiao, Yajuan Lyu, and Tian Wu. 2018c. Joint training of candidate extraction and answer selection for reading comprehension. In *Proceedings of ACL*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

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