Spans, Not Tokens: A Span-Centric Model for Multi-Span Reading Comprehension

Anonymous ACL submission

Abstract

Multi-span reading comprehension (MSRC) requires machines to extract multiple non-contiguous spans from a given context to answer a question. Existing MSRC methods either predict the positions of the start and end tokens of answer spans, or predict the BIO tag of each token. Such token-centric paradigms can hardly capture dependencies among spans which are critical to MSRC. In this paper, we propose a span-centric scheme where spans, as opposed to tokens, are directly represented and scored to qualify as answers. Thanks to the explicit representation of spans in the scheme, our implementation called SpanQualifier beneficially models intra-span and inter-span interactions. Our extensive experiments on three MSRC datasets demonstrate the effectiveness of our span-centric scheme and show that SpanQualifier achieves state-of-the-art results.

1 Introduction

Machine reading comprehension (MRC) aims to automatically answer questions from a given context, which is a challenge task of natural language understanding and a basic module of many question answering systems (Chen et al., 2017; Liu et al., 2020). MRC was usually regarded as a single-span extraction task (Rajpurkar et al., 2016, 2018; Kwiatkowski et al., 2019); single-span MRC models extract only one span as the answer (Seo et al., 2017; Wang and Jiang, 2017; Yu et al., 2018).

Task: It may be unrealistic to assume a continuous single span as a good answer to all questions. A more general task is multi-span reading comprehension (MSRC) (Zhu et al., 2020; Ju et al., 2022; Cui et al., 2021; Li et al., 2022). As shown in Figure 1, MSRC requires extracting an unspecified number (two in this example) of non-contiguous spans from a given context to answer a question. Moreover, spans are not independently extracted, for example, the second answer span "his mentor Leucippus" is dependent on the first answer span "Democritus", which needs to be captured.

Question: Who were the greek philosophers who contributed the basic information about atoms?

Context: Democritus was an Ancient Greek pre-Socratic philosopher primarily remembered today for his formulation of an atomic theory of the universe. His exact contributions are difficult to disentangle from those of his mentor Leucippus, as they are often mentioned together in texts. ...

Answer spans: { Democritus, his mentor Leucippus }

Figure 1: An example of MSRC task. Answer spans in the context are underlined.

Challenges: Existing methods for MSRC either predict the positions of the start and end tokens of answer spans by adapting single-span MRC models to MSRC (Hu et al., 2019; Yang et al., 2021), or treat MSRC as a sequence tagging task by predicting the Beginning-Inside-Outside (BIO) tag of each token (Segal et al., 2020; Li et al., 2022). We regard these methods as *token-centric* which define features, train models, and predict answers from the perspective of tokens. Their effectiveness is limited in two aspects. Firstly, it is non-trivial and often suboptimal to convert token-level predictions to multiple spans as answers. Secondly, token-level

representation and optimization are not conducive to capturing dependencies among spans.

Our Work: To overcome the limitations, we propose a *span-centric* scheme for MSRC and present an effective implementation called SpanQualifier. In response to the first limitation, our span-centric scheme generates representations for all spans in the context and predicts a qualification threshold. Spans are scored, and a span will qualify as an answer if its score exceeds the qualification threshold, thus avoiding the conversion from token-level predictions. For the second limitation, SpanQualifier interacts intra-span and inter-span representations to capture their dependencies and, moreover, a global loss function is designed to jointly optimize over all spans and predict their scores.

To summarize, our contributions include

- a novel span-centric scheme for MSRC which more naturally fits the task and empirically outperforms existing token-centric methods,
- an implementation of the scheme which models intra/inter-span interactions and obtains state-of-the-art results on three datasets.

Our code and data are in the supplementary material and will be open on GitHub after acceptance.

2 Related Work

2.1 Pointer Paradigm for MSRC

The Pointer paradigm based on Pointer Network (Vinyals et al., 2015) is one of the most popular schemes for single-span MRC, first proposed by Wang and Jiang (2017), and various successors extend it with sophisticated matching mechanisms (Seo et al., 2017; Yu et al., 2018). As shown in Figure 2a, the predicted probabilities of being the start or end of an answer span are normalized over all tokens in the context, and a post-processing decoding strategy such as beam search is adopted to select a span as the answer. It is non-trivial to adapt this paradigm to MSRC due to the unspecified number of target spans. MTMSN (Hu et al., 2019) predicts the number of answer spans and extracts non-overlapping spans based on an algorithm in computer vision (Rosenfeld and Thurston, 1971). MUSST (Yang et al., 2021) iteratively extracts and masks a span with the highest probability until meeting a stop symbol. Essentially, these methods predict token-level probabilities and convert them into multiple spans by non-differentiable decoding strategies, which may under-utilize the training

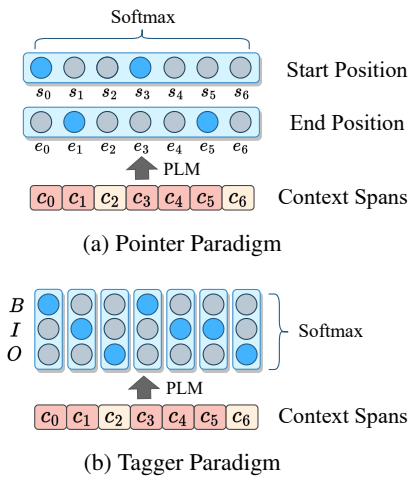

Figure 2: Two existing paradigms for MSRC.

data for MSRC. By contrast, RASOR (Lee et al., 2016) employs span representations to select a single span, which is extended by Chen et al. (2020) with span merging to extract at most two spans. This is still not enough for MSRC where answers are often more than two spans.

Unlike the above methods, in our proposed span-centric scheme, all possible spans are scored and may qualify as answers, and our end-to-end model is differentiable to exploit training data.

2.2 Tagger Paradigm for MSRC

The Tagger paradigm based on sequence tagging has been applied to many tasks, such as part-of-speech tagging and information extraction, and first applied to answer extraction by Yao et al. (2013). As shown in Figure 2b, a BIO tag is predicted for each token. This paradigm can be relatively easily adapted to MSRC (Yoon et al., 2022; Li et al., 2022). Further, TASE (Segal et al., 2020) employs a multi-head architecture to separately handle single-span and multi-span questions. Li et al. (2022) incorporates answer structure and answer number prediction into a multi-task model.

Tagger is trained by local normalization on each individual token, which suffers from label bias (Andor et al., 2016). It is less informative than global normalization performed in our implementation over all possible spans to consider their dependencies when calculating our loss function.

2.3 Information Extraction

Span-centric models have shown their effectiveness in information extraction tasks, including named entity recognition (NER), relation extraction, and

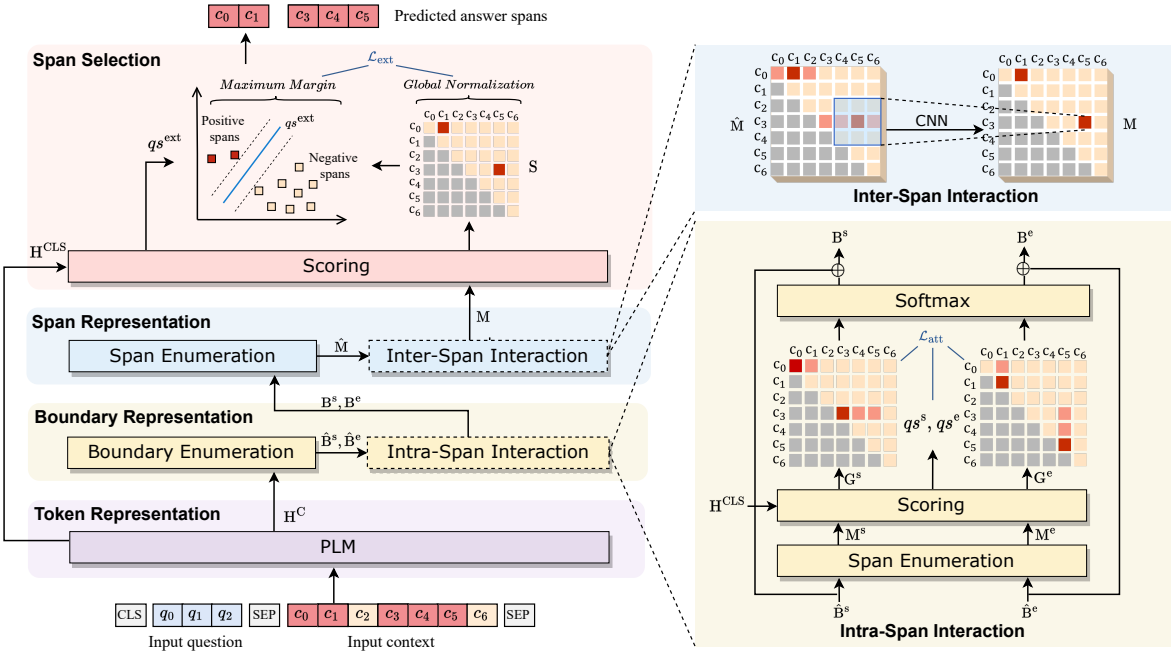

Figure 3: Overview of SpanQualifier. Dashed boxes represent extensions of the vanilla implementation.

event extraction (Luan et al., 2018, 2019; Wadden et al., 2019). They are trained by calculating a cross-entropy loss over all possible spans. However, this training objective is not suitable for MSRC due to the extremely unbalanced labels in its training data. Indeed, the context in a MSRC task typically contains tens of thousands of spans while only a few are answers.

Recently, Su et al. (2022) propose a NER model with a universal loss function that extends the softmax and cross-entropy loss for handling label imbalance, but their span representations obtained by interacting only between its start and end token representations are not sufficient for MSRC. As pointed out by Ju et al. (2022), different from NER which requires extracting all the spans of each given type, MSRC is more difficult and relies on a thorough understanding of the context because some spans of a given type are not correct answers. To enhance span representation, Tan et al. (2020) add a boundary detection task to predict the start and end tokens of entities, and some works use computationally expensive bi-affine (Yu et al., 2020) or even tri-affine (Yuan et al., 2022) mechanisms. By contrast, our SpanQualifier inexpensively obtains context-dependent span representations by two successive interaction layers where the former uses a supervised attention mechanism to highlight intra-span tokens to be aggregated and the latter captures inter-span dependencies.

3 Approach

Given a question Q and a context $C = c_0, \ldots, c_n$ with n tokens, the goal of MSRC is to extract a set of m answer indexes $A = \{[p_1^{\mathrm{s}}, p_1^{\mathrm{e}}], \ldots, [p_m^{\mathrm{s}}, p_m^{\mathrm{e}}]\}$ from the context, where p_i^{s} and p_i^{e} are the start and end positions of the i-th answer span.

Figure 3 shows an overview of our SpanQualifier, which simultaneously extracts multiple spans as answers by implementing a span-centric scheme with four stacked modules (Section 3.1), two extra interaction layers within and between spans (Section 3.2), and a loss function combining global normalization and maximum margin (Section 3.3).

3.1 (Vanilla) Span-Centric Scheme

The proposed scheme successively constructs token representations, boundary (i.e., start/end token) representations, and span representations, and then selects answers by predicting a qualification threshold for span score. In the remainder of this section, $\mathtt{MLP}(\cdot)$ represents a two-layer perceptron.

Token Representation We firstly capture the dependencies between the context C and the question Q to generate token representations. We feed Q and C into a pre-trained language model (PLM) such as BERT (Devlin et al., 2019) or ALBERT (Lan et al., 2020) in a standard way:

$$\mathbf{H} = \mathtt{PLM}([\mathrm{CLS}]\ Q\ [\mathrm{SEP}]\ C\ [\mathrm{SEP}])\,, \qquad (1)$$

where $\mathbf{H}$ represents the hidden states of the last layer of PLM. From $\mathbf{H}$ we extract the global representation $\mathbf{H}^{\text{CLS}} \in \mathbb{R}^{d_1}$ and the context representation $\mathbf{H}^{\text{C}} \in \mathbb{R}^{n \times d_1}$, where d_1 denotes the dimension of representation and n is the length of C.

Boundary Representation Then we aggregate the information about each possible span to its two boundary tokens. In our vanilla implementation, we take the context representation $\mathbf{H}^{\text{C}}$ as input and use a boundary enumeration layer to obtain the start and end token representations:

$$\begin{aligned} \mathbf{B}^{\text{s}}, \mathbf{B}^{\text{e}} &= \texttt{BoundaryEnum}(\mathbf{H}^{\text{C}}) \\ \text{where} \quad \mathbf{B}^{\text{s}} &= \texttt{MLP}(\mathbf{H}^{\text{C}}) \text{ and } \mathbf{B}^{\text{e}} = \texttt{MLP}(\mathbf{H}^{\text{C}}) \,, \end{aligned} \tag{2}$$

where $\mathbf{B}^{\text{s}}, \mathbf{B}^{\text{e}} \in \mathbb{R}^{n \times d_1}$. Note that an extra intra-span interaction layer can be inserted here to achieve more intensive information aggregation, which will be introduced in Section 3.2.

Span Representation Next, we generate the representations for all possible spans. In our vanilla implementation, the start and end token representations $\mathbf{B}^{\text{s}}, \mathbf{B}^{\text{e}}$ are fused by a span enumeration layer to obtain the representations of all possible spans:

$$\begin{aligned} \mathbf{M} &= \texttt{SpanEnum}(\mathbf{B}^{\text{s}}, \mathbf{B}^{\text{e}}) \\ &= \texttt{LayerNorm}(\mathbf{N}) \\ \text{where} \quad \mathbf{N}_{i,j} &= \mathbf{B}^{\text{s}}_i W^{\text{s}}_1 + \mathbf{B}^{\text{e}}_j W^{\text{e}}_1 + \mathbf{E}_{|j-i|} \,, \end{aligned} \tag{3}$$

where $\mathbf{M} \in \mathbb{R}^{n \times n \times d_2}$, $\mathbf{M}_{i,j}$ denotes the representation of the span starting at the i-th token and ending at the j-th token, and $\mathbf{E}_{|j-i|} \in \mathbb{R}^{d_2}$ is a learnable embedding of span length. $\texttt{LayerNorm}(\cdot)$ is a layer normalization operator (Ba et al., 2016), and $W^{\text{s}}_1, W^{\text{e}}_1 \in \mathbb{R}^{d_1 \times d_2}$ are learnable parameters used to reduce the dimension from d_1 to d_2 to improve efficiency. We empirically set $d_2 = 64$. Note that an extra interaction layer for capturing dependencies among spans can be inserted here, which will be introduced in Section 3.2.

Span Selection Finally, we score all possible spans and select answer spans by predicting a qualification threshold. We feed both the span representations $\mathbf{M}$ and the global representation $\mathbf{H}^{\text{CLS}}$ into a scoring layer:

$$\begin{aligned} \mathbf{S}, qs^{\text{ext}} &= \texttt{Scoring}(\mathbf{M}, \mathbf{H}^{\text{CLS}}) \\ \text{where} \quad \mathbf{S} &= \texttt{MLP}(\mathbf{M}) \text{ and } qs^{\text{ext}} = \texttt{MLP}(\mathbf{H}^{\text{CLS}}) \,, \end{aligned} \tag{4}$$

where $\mathbf{S} \in \mathbb{R}^{n \times n}$ contains the scores of all possible spans and $\mathbf{S}_{i,j} \in \mathbf{S}$ denotes the score of the span starting at the i-th token and ending at the j-th token. The lower triangular portion of $\mathbf{S}$ and the portion representing spans containing more than k tokens are masked, where k is the longest length of an answer in the training set.

At inference time, all the spans with score higher than qs^{ext} will qualify as answers, where $qs^{\text{ext}} \in \mathbb{R}$ represents a qualification threshold. If two spans overlap, we will keep the higher scored one.

3.2 Span Interaction

To extend the vanilla implementation, two interaction layers are added to enhance span representations and capture dependencies among spans.

Intra-Span Interaction Not only the start and end tokens but also those inside a span are crucial to its representation. Therefore, we insert an intra-span interaction layer into the boundary representation module after Equation (2) to incorporate the attention weights of internal tokens:

$$\begin{aligned} \hat{\mathbf{B}}^{\text{s}}, \hat{\mathbf{B}}^{\text{e}} &= \texttt{BoundaryEnum}(\mathbf{H}^{\text{C}}) \,, \\ \mathbf{B}^{\text{s}}, \mathbf{B}^{\text{e}} &= \texttt{IntraSpan}(\hat{\mathbf{B}}^{\text{s}}, \hat{\mathbf{B}}^{\text{e}}) \,. \end{aligned} \tag{5}$$

As for $\texttt{IntraSpan}(\cdot, \cdot)$, our implementation resembles Equation (3) and Equation (4). Start token representations are intensified as follows:

$$\begin{aligned} \mathbf{M}^{\text{s}} &= \texttt{SpanEnum}(\hat{\mathbf{B}}^{\text{s}}, \hat{\mathbf{B}}^{\text{e}}) \,, \\ \mathbf{G}^{\text{s}}, qs^{\text{s}} &= \texttt{Scoring}(\mathbf{M}^{\text{s}}, \mathbf{H}^{\text{CLS}}) \,, \\ \mathbf{B}^{\text{s}} &= \texttt{Softmax}(\mathbf{G}^{\text{s}})(\hat{\mathbf{B}}^{\text{s}} W^{\text{s}}_2) \,, \end{aligned} \tag{6}$$

and end token representations are intensified by

$$\begin{aligned} \mathbf{M}^{\text{e}} &= \texttt{SpanEnum}(\hat{\mathbf{B}}^{\text{s}}, \hat{\mathbf{B}}^{\text{e}}) \,, \\ \mathbf{G}^{\text{e}}, qs^{\text{e}} &= \texttt{Scoring}(\mathbf{M}^{\text{e}}, \mathbf{H}^{\text{CLS}}) \,, \\ \mathbf{B}^{\text{e}} &= \texttt{Softmax}((\mathbf{G}^{\text{e}})^{\intercal})(\hat{\mathbf{B}}^{\text{e}} W^{\text{e}}_2) \,, \end{aligned} \tag{7}$$

where $\mathbf{G}^{\text{s}}, \mathbf{G}^{\text{e}}$ represent attention weights, and $W^{\text{s}}_2, W^{\text{e}}_2 \in \mathbb{R}^{d_1 \times d_1}$ are learnable parameters.

Inter-Span Interaction To capture the dependencies among spans which are useful in MSRC, we insert an inter-span interaction layer into the span representation module after Equation (3):

$$\begin{aligned} \hat{\mathbf{M}} &= \texttt{SpanEnum}(\mathbf{B}^{\text{s}}, \mathbf{B}^{\text{e}}) \,, \\ \mathbf{M} &= \texttt{InterSpan}(\hat{\mathbf{M}}) \,, \end{aligned} \tag{8}$$

where we implement $\texttt{InterSpan}(\cdot)$ by an one-layer 2D convolutional neural network (CNN).

3.3 Training

Since answer spans in MSRC correlate with each other, a global qualification loss function is introduced for supervising span selection. It is also adapted to supervise intra-span interaction.

	# Train	# Dev	# Test	Distribution by answer number			Average answer number	Average context length
				0	1	≥ 2		
MultiSpanQA	4,577	653	653	-	-	100.00%	2.89	251
MultiSpanQA-E	13,731	1,959	1,959	33.43%	32.59%	33.98%	1.30	226
QUOREF	16,083	2,186	2,276	-	90.56%	9.44%	1.14	324

Table 1: Dataset statistics.

Global Qualification Loss The Global Qualification (GQ) loss consists of two parts: the Global Normalization (GN) loss and the Maximum Margin (MM) loss. Considering the correlation between positive spans, we use the GN loss to jointly optimize the scores of all positive spans. Moreover, we use the MM loss to maximize the margin between positive and negative spans.

Given a set of scores of positive spans $\Omega^+ \subseteq \mathbf{S}$, a set of scores of negative spans $\Omega^- \subseteq \mathbf{S}$, and a qualification threshold qs^{ext}, the GN loss is

$$\mathrm{L}_{\text{GN}}(\Omega^+, \Omega^-) = -\log \frac{\sum\limits_{s \in \Omega^+} \exp(s)}{\sum\limits_{s \in \Omega^+} \exp(s) + \sum\limits_{s \in \Omega^-} \exp(s)} . \tag{9}$$

The objective is to maximize the sum of probabilities of all positive spans. The MM loss comprises

$$\begin{aligned} \mathrm{L}^{+}_{\text{MM}}(\Omega^+, qs^{\text{ext}}) &= \texttt{Max}(1 - (\texttt{Min}(\Omega^+) - qs^{\text{ext}}), 0) , \\ \mathrm{L}^{-}_{\text{MM}}(\Omega^-, qs^{\text{ext}}) &= \texttt{Max}(1 - (qs^{\text{ext}} - \texttt{Max}(\Omega^-)), 0) . \end{aligned} \tag{10}$$

The objective is to maximize the soft margin between positive and negative spans. The GQ loss combines the GN loss and the MM loss:

$$\begin{aligned} &\mathrm{L}_{\text{GQ}}(\Omega^+, \Omega^-, qs^{\text{ext}}) \\ &= \texttt{Mean}(\mathrm{L}_{\text{GN}}(\Omega^+, \Omega^-), \mathrm{L}^{+}_{\text{MM}}(\Omega^+, qs^{\text{ext}}), \mathrm{L}^{-}_{\text{MM}}(\Omega^-, qs^{\text{ext}})) . \end{aligned} \tag{11}$$

Supervised Extraction We employ the GQ loss as our main loss for supervising span extraction:

$$\mathcal{L}_{\text{ext}} = \mathrm{L}_{\text{GQ}}(\Omega^+, \Omega^-, qs^{\text{ext}}) . \tag{12}$$

Supervised Attention We also adapt the GQ loss to supervise the attention weights in the intra-span interaction layer. Let $\Psi^{\text{s}+} \subseteq \mathbf{G}^{\text{s}}$ and $\Psi^{\text{e}+} \subseteq \mathbf{G}^{\text{e}}$ be the sets of attention weights of tokens in positive spans. For each positive span $[p^{\text{s}}, p^{\text{e}}]$, $\Psi^{\text{s}+}$ contains $\mathbf{G}^{\text{s}}_{p^{\text{s}},p^{\text{s}}}, \mathbf{G}^{\text{s}}_{p^{\text{s}},p^{\text{s}}+1}, \ldots, \mathbf{G}^{\text{s}}_{p^{\text{s}},p^{\text{e}}} \in \mathbf{G}^{\text{s}}$, and $\Psi^{\text{e}+}$ contains $\mathbf{G}^{\text{s}}_{p^{\text{s}},p^{\text{e}}}, \mathbf{G}^{\text{s}}_{p^{\text{s}}+1,p^{\text{e}}}, \ldots, \mathbf{G}^{\text{s}}_{p^{\text{e}},p^{\text{e}}} \in \mathbf{G}^{\text{e}}$. Let $\Psi^{\text{s}-}$ and $\Psi^{\text{e}-}$ be the sets of the remaining attention weights in $\mathbf{G}^{\text{s}}$ and $\mathbf{G}^{\text{e}}$, respectively. Given the thresholds qs^{s} and qs^{e} computed by Equation (6) and Equation (7), respectively, our attention loss is

$$\mathcal{L}_{\text{att}} = \mathrm{L}_{\text{GQ}}(\Psi^{\text{s}+}, \Psi^{\text{s}-}, qs^{\text{s}}) + \mathrm{L}_{\text{GQ}}(\Psi^{\text{e}+}, \Psi^{\text{e}-}, qs^{\text{e}}) . \tag{13}$$

Total Loss Finally, we sum the extraction loss and the attention loss for joint training:

$$\mathcal{L} = \mathcal{L}_{\text{ext}} + \mathcal{L}_{\text{att}} . \tag{14}$$

4 Experimental Setup

4.1 Datasets

We used three MSRC datasets. **MultiSpanQA** and **MultiSpanQA-E**xpanded (Li et al., 2022) are the latest datasets for MSRC created from a question answering dataset (Kwiatkowski et al., 2019). MultiSpanQA only contains multi-span questions, while MultiSpanQA-E also contains unanswerable and single-span questions. **QUOREF** (Dasigi et al., 2019) containing questions requiring coreferential reasoning has been widely used in MSRC research.

Since the official test sets of these datasets are not public, we took their official dev set as our test set and randomly held out an in-house dev set from the official training set. Some statistics about the datasets are shown in Table 1.

4.2 Participating Methods

We experimented with two variants of our method: **SpanQualifier** representing the full version and **SpanQualifier$_{\text{vanilla}}$** representing the vanilla version without span interaction in Section 3.2.

We compared with two pointer-based methods for MSRC. **MTMSN** (Hu et al., 2019) predicts the number of answer spans and repeatedly extracts non-overlapping spans until reaching the predicted number. **MUSST** (Yang et al., 2021) iteratively extracts spans until meeting a special stop symbol.

We compared with four tagger-based methods for MSRC. **Tagger$_{\text{vanilla}}$** (Li et al., 2022) uses a standard BIO tagger. **Tagger$_{\text{multi-task}}$** (Li et al., 2022) enhances Tagger$_{\text{vanilla}}$ with answer structure and answer number prediction. **Tagger$_{\text{crf}}$** extends Tagger$_{\text{vanilla}}$ with a CRF layer (Lafferty et al., 2001) to capture tag dependencies. **TASE** (Segal et al., 2020) employs a multi-head mechanism to separately handle single-span and multi-span questions. We further extended it to separately handle unanswerable questions in MultiSpanQA-E.

	MultiSpanQA				MultiSpanQA-E				QUOREF			
	EM		PM		EM		PM		EM		PM	
	dev	test	dev	test	dev	test	dev	test	dev	test	dev	test
BERT												
MTMSN	46.67$^{\bullet\blacktriangle}$	43.55$^{\bullet\blacktriangle}$	72.17$^{\bullet\blacktriangle}$	70.44$^{\bullet\blacktriangle}$	49.96$^{\bullet\blacktriangle}$	50.60$^{\bullet\blacktriangle}$	70.61$^{\bullet\blacktriangle}$	70.92 $^{\blacktriangle}$	62.26$^{\circ\blacktriangle}$	62.23 $^{\blacktriangle}$	73.81 $^{\blacktriangle}$	73.62 $^{\blacktriangle}$
MUSST	51.35$^{\bullet\blacktriangle}$	48.83$^{\bullet\blacktriangle}$	74.61$^{\bullet\blacktriangle}$	72.36$^{\bullet\blacktriangle}$	54.26$^{\bullet\blacktriangle}$	54.65$^{\bullet\blacktriangle}$	70.81$^{\bullet\blacktriangle}$	70.60$^{\bullet\blacktriangle}$	33.35$^{\bullet\blacktriangle}$	31.91$^{\bullet\blacktriangle}$	55.21$^{\bullet\blacktriangle}$	54.23$^{\bullet\blacktriangle}$
Tagger$_{\text{vanilla}}$	60.36$^{\bullet\blacktriangle}$	57.89$^{\bullet\blacktriangle}$	73.76$^{\bullet\blacktriangle}$	72.60$^{\bullet\blacktriangle}$	60.39$^{\bullet\blacktriangle}$	59.81$^{\bullet\blacktriangle}$	72.38 $^{\triangle}$	72.33 $^{\blacktriangle}$	56.33$^{\bullet\blacktriangle}$	55.67$^{\bullet\blacktriangle}$	68.32$^{\bullet\blacktriangle}$	67.47$^{\bullet\blacktriangle}$
Tagger$_{\text{multi-task}}$	51.63$^{\bullet\blacktriangle}$	51.72$^{\bullet\blacktriangle}$	67.97$^{\bullet\blacktriangle}$	69.06$^{\bullet\blacktriangle}$	54.93$^{\bullet\blacktriangle}$	56.53$^{\bullet\blacktriangle}$	68.54$^{\bullet\blacktriangle}$	70.36$^{\bullet\blacktriangle}$	39.55$^{\bullet\blacktriangle}$	39.00$^{\bullet\blacktriangle}$	51.30$^{\bullet\blacktriangle}$	49.75$^{\bullet\blacktriangle}$
Tagger$_{\text{crf}}$	61.28$^{\bullet\blacktriangle}$	58.94$^{\bullet\blacktriangle}$	74.55$^{\bullet\blacktriangle}$	73.43$^{\bullet\blacktriangle}$	57.00$^{\bullet\blacktriangle}$	57.23$^{\bullet\blacktriangle}$	66.94$^{\bullet\blacktriangle}$	67.18$^{\bullet\blacktriangle}$	56.73$^{\bullet\blacktriangle}$	55.09$^{\bullet\blacktriangle}$	68.24$^{\bullet\blacktriangle}$	66.89$^{\bullet\blacktriangle}$
TASE	57.41$^{\bullet\blacktriangle}$	54.99$^{\bullet\blacktriangle}$	73.17$^{\bullet\blacktriangle}$	72.59$^{\bullet\blacktriangle}$	51.01$^{\bullet\blacktriangle}$	52.47$^{\bullet\blacktriangle}$	65.38$^{\bullet\blacktriangle}$	66.99$^{\bullet\blacktriangle}$	58.58$^{\bullet\blacktriangle}$	56.98$^{\bullet\blacktriangle}$	69.28$^{\bullet\blacktriangle}$	67.50$^{\bullet\blacktriangle}$
SpanQualifier$_{\text{vanilla}}$	66.38	64.92	78.61	77.69	63.25	64.24	73.19	73.80	63.92	62.31	73.33	71.69
SpanQualifier	**69.94**	**68.41**	**81.26**	**80.61**	**65.47**	**66.08**	**74.09**	**75.11**	**67.19**	**65.65**	**75.70**	**74.11**
ALBERT												
MTMSN	40.85$^{\bullet\blacktriangle}$	38.12$^{\bullet\blacktriangle}$	63.17$^{\bullet\blacktriangle}$	61.14$^{\bullet\blacktriangle}$	41.74$^{\bullet\blacktriangle}$	43.90$^{\bullet\blacktriangle}$	56.28$^{\bullet\blacktriangle}$	58.58$^{\bullet\blacktriangle}$	45.78$^{\bullet\blacktriangle}$	44.35$^{\bullet\blacktriangle}$	57.91$^{\bullet\blacktriangle}$	56.64$^{\bullet\blacktriangle}$
MUSST	54.13$^{\bullet\blacktriangle}$	51.69$^{\bullet\blacktriangle}$	77.45$^{\bullet\blacktriangle}$	75.56$^{\bullet\blacktriangle}$	57.27$^{\bullet\blacktriangle}$	57.34$^{\bullet\blacktriangle}$	73.95$^{\bullet\blacktriangle}$	74.52$^{\bullet\blacktriangle}$	41.73$^{\bullet\blacktriangle}$	39.27$^{\bullet\blacktriangle}$	65.13$^{\bullet\blacktriangle}$	63.48$^{\bullet\blacktriangle}$
Tagger$_{\text{vanilla}}$	67.06$^{\bullet\blacktriangle}$	64.78$^{\bullet\blacktriangle}$	80.07$^{\bullet\blacktriangle}$	79.32$^{\bullet\blacktriangle}$	61.73$^{\bullet\blacktriangle}$	61.36$^{\bullet\blacktriangle}$	72.77$^{\bullet\blacktriangle}$	72.73$^{\bullet\blacktriangle}$	70.78$^{\bullet\blacktriangle}$	68.97$^{\bullet\blacktriangle}$	79.68$^{\bullet\blacktriangle}$	77.68$^{\bullet\blacktriangle}$
Tagger$_{\text{multi-task}}$	49.78$^{\bullet\blacktriangle}$	48.75$^{\bullet\blacktriangle}$	67.99$^{\bullet\blacktriangle}$	67.34$^{\bullet\blacktriangle}$	51.65$^{\bullet\blacktriangle}$	52.15$^{\bullet\blacktriangle}$	65.89$^{\bullet\blacktriangle}$	66.38$^{\bullet\blacktriangle}$	50.95$^{\bullet\blacktriangle}$	48.91$^{\bullet\blacktriangle}$	61.77$^{\bullet\blacktriangle}$	59.04$^{\bullet\blacktriangle}$
Tagger$_{\text{crf}}$	68.11$^{\bullet\blacktriangle}$	65.43$^{\bullet\blacktriangle}$	81.03$^{\bullet\blacktriangle}$	80.00$^{\bullet\blacktriangle}$	61.99$^{\bullet\blacktriangle}$	61.75$^{\bullet\blacktriangle}$	72.46$^{\bullet\blacktriangle}$	72.86$^{\bullet\blacktriangle}$	70.63$^{\bullet\blacktriangle}$	69.03$^{\bullet\blacktriangle}$	79.35$^{\bullet\blacktriangle}$	77.75$^{\bullet\blacktriangle}$
TASE	65.55$^{\bullet\blacktriangle}$	62.06$^{\bullet\blacktriangle}$	80.85$^{\bullet\blacktriangle}$	79.34$^{\bullet\blacktriangle}$	57.52$^{\bullet\blacktriangle}$	56.95$^{\bullet\blacktriangle}$	70.95$^{\bullet\blacktriangle}$	71.11$^{\bullet\blacktriangle}$	71.90$^{\bullet\blacktriangle}$	70.31$^{\bullet\blacktriangle}$	81.39$^{\bullet\blacktriangle}$	79.90$^{\bullet\blacktriangle}$
SpanQualifier$_{\text{vanilla}}$	68.92	67.23	82.42	81.76	65.68	66.07	75.85	76.53	73.90	71.77	81.95	79.92
SpanQualifier	**72.29**	**70.20**	**83.62**	**82.68**	**67.24**	**66.99**	**76.79**	**77.01**	**75.18**	**72.84**	**82.56**	**80.68**

Table 2: Comparison with baselines. We mark the results of baselines that are significantly lower than the results of SpanQualifier$_{\text{vanilla}}$ under $p < 0.01$ ($^{\bullet}$) or $p < 0.05$ ($^{\circ}$), and of SpanQualifier under $p < 0.01$ ($^{\blacktriangle}$) or $p < 0.05$ ($^{\triangle}$).

4.3 Evaluation Metrics

We followed Li et al. (2022) to use two metrics. Exact Match F1 (**EM**) measures the percentage of predicted spans that exactly match any gold-standard answer span. Partial Match F1 (**PM**) generalizes EM by using the length of the longest common substring to score each predicted span based on its nearest gold-standard answer span.

4.4 Implementation Details

We used two PLMs of different sizes: BERT-base-uncased (Devlin et al., 2019) and ALBERT-base-v2 (Lan et al., 2020). We implemented our models with PyTorch 1.10 and HuggingFace Transformers 4.12.5. The dimension of the hidden layer of MLP was 768. We implemented CNN using kernel size $= 5 \times 5$, stride $= 1$, and in-channels $=$ out-channels $= 64$. We used the Adam optimizer and set warmup fraction $= 0.1$, weight decay $= 0.01$, maximum length $= 512$, and epoch $= 30$. For each model, we searched for the best learning rate from $\{1e-5, 3e-5\}$, and batch size from $\{24, 32\}$. We used three seeds $\{0, 1, 2\}$ and took the mean results. All the experiments were performed on RTX 3090 (24G).

5 Experimental Results

5.1 Comparison with Baselines

In Table 2, SpanQualifer achieves the best results in all the settings. It significantly outperforms all the baselines under $p < 0.01$ except in one setting under $p < 0.05$. Satisfyingly, SpanQualifier$_{\text{vanilla}}$ also exceeds most baselines, demonstrating the effectiveness of our vanilla span-centric scheme.

	MultiSpanQA-E		QUOREF	
	EM	PM	EM	PM
BERT				
MTMSN	35.86	64.85	57.69	75.34
MUSST	47.20	67.23	29.79	57.15
Tagger$_{\text{vanilla}}$	59.61	73.52	61.66	74.43
Tagger$_{\text{multi-task}}$	54.64	70.58	42.11	56.02
Tagger$_{\text{crf}}$	56.99	68.51	60.85	73.81
TASE	49.41	65.43	61.47	74.17
SpanQualifier$_{\text{vanilla}}$	63.53	74.35	64.57	74.75
SpanQualifier	**65.59**	**75.98**	**68.82**	**77.98**
ALBERT				
MTMSN	27.58	47.48	36.48	54.20
MUSST	50.40	71.60	33.12	64.17
Tagger$_{\text{vanilla}}$	61.58	74.92	71.44	82.55
Tagger$_{\text{multi-task}}$	48.23	65.59	49.99	65.16
Tagger$_{\text{crf}}$	62.11	75.03	71.70	**83.15**
TASE	53.37	69.92	70.55	81.80
SpanQualifier$_{\text{vanilla}}$	64.67	76.29	**72.38**	81.67
SpanQualifier	**65.81**	**76.91**	72.31	81.83

Table 3: Comparison with baselines on multi-span questions in the test sets of MultiSpanQA-E and QUOREF.

By excluding single-span and unanswerable questions from MultiSpanQA-E and QUOREF, in Table 3, the gap between SpanQualifier and pointer-based methods becomes larger on multi-span questions. Indeed, by contrast with pointer-based methods, SpanQualifier and tagger-based methods achieve better results on multi-span questions than on other questions, especially on QUOREF, indicating their better suitability for the MSRC task.

5.2 Ablation Study: Span Representation

To analyze the effectiveness of our span-centric scheme, we implemented TokenQualifier which is

	MultiSpanQA		MultiSpanQA-E		QUOREF	
	EM	PM	EM	PM	EM	PM
BERT						
SpanQualifier$_{\text{vanilla}}$	**64.92**	**77.69**	**64.24**	**73.80**	**62.31**	**71.69**
TokenQualifier	61.35	75.52	59.81	72.33	60.79	70.25
ALBERT						
SpanQualifier$_{\text{vanilla}}$	**67.23**	**81.76**	**66.07**	**76.53**	**71.77**	**79.92**
TokenQualifier	66.28	80.88	63.52	75.67	70.58	78.89

Table 4: Ablation study of span representation on the test sets of three datasets.

	MultiSpanQA		MultiSpanQA-E		QUOREF	
	EM	PM	EM	PM	EM	PM
BERT						
SpanQualifier	**68.41**	**80.61**	**66.08**	75.11	**65.65**	**74.11**
w/o intra-span	68.20	79.84	65.00	73.69	65.49	74.01
w/o inter-span	64.76	77.65	65.61	75.10	62.94	72.27
w/o $\mathcal{L}_{\text{att}}$	67.15	78.96	64.12	72.83	65.09	73.66
standard attention	67.89	79.86	65.88	**75.16**	65.15	73.97
ALBERT						
SpanQualifier	**70.20**	**82.68**	66.99	**77.01**	**72.84**	**80.68**
w/o intra-span	69.14	81.73	**67.11**	76.81	72.60	80.30
w/o inter-span	68.28	81.88	65.30	76.14	71.51	79.74
w/o $\mathcal{L}_{\text{att}}$	68.87	81.56	66.08	75.80	71.89	80.05
Standard Attention	68.12	81.13	65.75	76.72	72.12	80.01

Table 5: Ablation study of span interaction on the test sets of three datasets.

a variant of SpanQualifier$_{\text{vanilla}}$ by removing boundary and span representations and directly using token representations for prediction. In Table 4, EM and PM drop by 0.95–4.43 and 0.86–2.17, respectively, demonstrating the usefulness of our boundary and span representations.

5.3 Ablation Study: Span Interaction

Our span interaction consists of an intra-span layer with a supervised attention mechanism and an inter-span layer. To analyze their effectiveness, we conducted an ablation study by separately removing each component. We also replaced our attention mechanism for computing $\mathbf{G}^{\text{s}}$ and $\mathbf{G}^{\text{e}}$ in Equation (6) and Equation (7), respectively, with a standard attention mechanism where the key and value are $\texttt{MLP}(\hat{\mathbf{B}}^{\text{s}})$ and $\texttt{MLP}(\hat{\mathbf{B}}^{\text{e}})$, respectively.

In Table 5, the full version of SpanQualifier outperforms its variants in almost all the settings, demonstrating the usefulness of the removed/replaced components. In particular, on MultiSpanQA and QUOREF, inter-span interaction appears more influential, while on MultiSpanQA-E, attention supervision is more helpful.

5.4 Official Leaderboards

We submitted the results predicted by SpanQualifier on each official test set to the leaderboard: to MultiSpanQA and MultiSpanQA-E[1] on 10/11/2022, and to QUOREF[2] on 10/12/2022. For a fair comparison, we used the same PLMs as other methods on the leaderboard: BERT Base (Devlin et al., 2019) and RoBERTa Large (Liu et al., 2019) on MultiSpanQA and MultiSpanQA-E, and RoBERTa Large and CorefRoBERTa Large (Ye et al., 2020) on QUOREF.

	MultiSpanQA		MultiSpanQA-E	
	EM	PM	EM	PM
SpanQualifier (RoBERTa Large)	**72.75**	**85.41**	**70.85**	**81.64**
Tagger$_{\text{vanilla}}$ (RoBERTa Large)	69.19	83.86	69.05	78.97
Tagger+LIQUID (RoBERTa Base)	67.40	81.16	–	–
SpanQualifier (BERT Base)	64.87	78.79	64.39	74.22
Tagger$_{\text{multi-task}}$ (BERT Base)	59.28	76.26	42.26	70.47
Tagger$_{\text{vanilla}}$ (BERT Base)	56.45	75.22	41.38	70.10
Single-span (BERT Base)	14.41	67.56	12.26	67.73

Table 6: Official leaderboards of MultiSpanQA and MultiSpanQA-E.

	QUOREF	
	Global EM	Global F1
SpanQualifier (CorefRoBERTa Large)	**81.36**	**87.24**
SpanQualifier (RoBERTa Large)	81.24	87.05
TASE (CorefRoBERTa Large)	80.61	86.70
TASE$_{\text{CoNLL-joint-qgen}}$	80.45	86.46
TASE (RoBERTa Large)	79.66	86.13
CorefAdditive	79.11	85.86
CorefRoBERTa Large	75.80	82.81

Table 7: Official leaderboard of QUOREF.

Table 6 compares SpanQualifier with other top-ranked single-model methods on the leaderboards of MultiSpanQA and MultiSpanQA-E at the time of our submission. SpanQualifier outperforms all the methods using the same PLM, including **Tagger+LIQUID** (Lee et al., 2023) whose implementation is not public, and **Single-span** (Li et al., 2022) which is a pointer-based method selecting multiple spans by tuning a threshold on the dev set.

Table 7 compares SpanQualifier with other top-ranked single-model methods on the leaderboard of QUOREF at the time of our submission. Note that this leaderboard uses Global Exact Match (**Global EM**) (Rajpurkar et al., 2016) and **Global F1** (Dua et al., 2019) as evaluation metrics. SpanQualifier outperforms all the methods, including **TASE$_{\text{CoNLL-joint-qgen}}$** which jointly trains TASE on QUOREF and CoNLL (Sang and Meulder, 2003), **CorefAdditive** (Zhang and Zhao, 2022) and **CorefRoBERTa Large** (Ye et al., 2020) which are pointer-based methods predicting the number of answer spans as in MTMSN.

[1] https://multi-span.github.io/

[2] https://leaderboard.allenai.org/quoref/submissions/public

Question	Context	Prediction
Which two hormones are known for their roles in the bluefight-or-flight response?	His theory states that ... preparing the animal for fighting or fleeindull. More specifically, the adrenal medulla produces a hormonal cascade that results in the secretion of catecholamines, especially norepinephrine and epinephrine. The hormones estrogen, testosterone, and cortisol, as well as ... also affect how organisms react to stress ...	{norepinephrine, epinephrine, estrogen, testosterone, cortisol}
Where does the red hair gene come from?	Red hair ... occurs more frequently in people of northern or western European ancestry, and less frequently in other populations. Red hair appears most commonly in people with two copies of a recessive allele on chromosome 16 which produces an altered version of the MC1R protein ...	{northern or western European ancestry, other populations}

Table 8: Answer spans predicted by SpanQualifier on MultiSpanQA. Gold-standard answer spans are underlined.

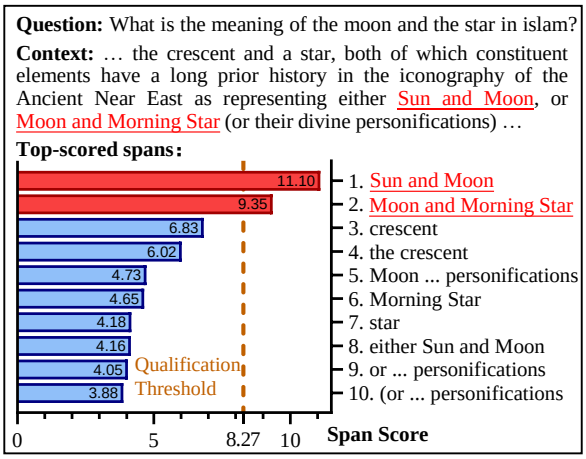

Figure 4: Top spans scored by SpanQualifier on MultiSpanQA. Gold-standard answer spans are underlined.

5.5 Case Study

Figure 4 illustrates the top-scored spans predicted by SpanQualifier for a question in MultiSpanQA. The top-two spans with scores higher than the predicted qualification threshold exactly match the gold standard. By contrast, the pointer-based MUSST predicted a long span containing 52 words, and $\text{Tagger}_{\text{crf}}$ predicted three single-word spans. While their spans partially match the gold standard, their inexactness reveals the limitation of token-centric methods in MSRC, particularly the suboptimality of converting token-level predictions to multiple answer spans, which has been addressed by our span-centric scheme.

5.6 Error Analysis

We manually analyzed errors made by SpanQualifier on MultiSpanQA and identified two common types of errors in MSRC: incorrect homogeneous answers and missing heterogeneous answers.

For the first type of error, as illustrated in the upper part of Table 8, the gold-standard answer spans are surrounded by many other spans of the same type, which misled our model to pay attention to their dependencies and extract incorrect answers such as “estrogen”, “testosterone”, and “cortisol”.

For the second type of error, as illustrated in the lower part of Table 8, the question has multiple heterogeneous answers of distinct types, including population, chromosome, and protein. Their dependencies are difficult to be captured so that our model failed to find some of them.

	BERT	ALBERT
MTMSN	115M	17M
MUSST	109M	12M
$\text{Tagger}_{\text{vanilla}}$	109M	12M
$\text{Tagger}_{\text{multi-task}}$	117M	20M
$\text{Tagger}_{\text{crf}}$	109M	12M
TASE	111M	13M
$\text{SpanQualifier}_{\text{vanilla}}$	112M	15M
SpanQualifier	115M	18M

Table 9: Model size (number of parameters).

5.7 Model Size and Inference Time

Table 9 compares model sizes. Despite its superior performance, SpanQualifier has a number of parameters comparable to those of the baselines.

BERT-based $\text{SpanQualifier}_{\text{vanilla}}$ and SpanQualifier use an average of 29.69ms and 33.72ms to answer a question, respectively. ALBERT-based $\text{SpanQualifier}_{\text{vanilla}}$ and SpanQualifier use an average of 36.87ms and 42.15ms, respectively. The inference time of our models is reasonably fast.

6 Conclusion

In this paper, we explore the effectiveness of span-centric scheme for MSRC and propose a novel extraction model SpanQualifier. Experiments show that our span-centric scheme outperforms existing pointer-based and tagger-based methods which follow a token-centric scheme and, furthermore, with span interaction our model achieves state-of-the-art results on three MSRC datasets. In the future, we plan to add more expressive interaction methods to the span representation module and to soft answer labels to accommodate overlapping spans.

Limitations

Our model was specifically designed for MSRC. Some modules (e.g., inter-span interaction) may not be useful for conventional single-span reading comprehension. Moreover, considering the potentially different data distributions between single-span and multi-span questions, our model trained on a hybrid of single-span and multi-span questions (e.g., on QUOREF in our experiments) may not outperform dedicatedly trained models.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

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

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1870–1879. Association for Computational Linguistics.

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

Peng Cui, Dongyao Hu, and Le Hu. 2021. Listreader: Extracting list-form answers for opinion questions. *CoRR*, abs/2110.11692.

Pradeep Dasigi, Nelson F. Liu, Ana Marasovic, Noah A. Smith, and Matt Gardner. 2019. Quoref: A reading comprehension dataset with questions requiring coreferential 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 2019, Hong Kong, China, November 3-7, 2019*, pages 5924–5931. Association for Computational Linguistics.

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

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

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 2019, Hong Kong, China, November 3-7, 2019*, pages 1596–1606. Association for Computational Linguistics.

Yiming Ju, Weikang Wang, Yuanzhe Zhang, Suncong Zheng, Kang Liu, and Jun Zhao. 2022. CMQA: A dataset of conditional question answering with multiple-span answers. In *Proceedings of the 29th International Conference on Computational Linguistics, COLING 2022, Gyeongju, Republic of Korea, October 12-17, 2022*, pages 1697–1707. International Committee on Computational Linguistics.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, pages 282–289. Morgan Kaufmann.

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.

Kenton Lee, Tom Kwiatkowski, Ankur P. Parikh, and Dipanjan Das. 2016. Learning recurrent span representations for extractive question answering. *CoRR*, abs/1611.01436.

Seongyun Lee, Hyunjae Kim, and Jaewoo Kang. 2023. Liquid: A framework for list question answering dataset generation. In *The Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023*.

Haonan Li, Martin Tomko, Maria Vasardani, and Timothy Baldwin. 2022. Multispanqa: A dataset for multi-span question answering. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 1250–1260. Association for Computational Linguistics.

Dayiheng Liu, Yeyun Gong, Jie Fu, Yu Yan, Jiusheng Chen, Daxin Jiang, Jiancheng Lv, and Nan Duan. 2020. Rikinet: Reading wikipedia pages for natural question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 6762–6771. Association for Computational Linguistics.

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. *CoRR*, abs/1907.11692.

Yi Luan, Luheng He, Mari Ostendorf, and Hannaneh Hajishirzi. 2018. Multi-task identification of entities, relations, and coreference for scientific knowledge graph construction. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3219–3232. Association for Computational Linguistics.

Yi Luan, Dave Wadden, Luheng He, Amy Shah, Mari Ostendorf, and Hannaneh Hajishirzi. 2019. A general framework for information extraction using dynamic span graphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 3036–3046. Association for Computational Linguistics.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 784–789. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2383–2392. The Association for Computational Linguistics.

Azriel Rosenfeld and Mark Thurston. 1971. Edge and curve detection for visual scene analysis. *IEEE Trans. Computers*, 20(5):562–569.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning, CoNLL 2003, Held in cooperation with HLT-NAACL 2003, Edmonton, Canada, May 31 - June 1, 2003*, pages 142–147. ACL.

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 2020, Online, November 16-20, 2020*, pages 3074–3080. Association for Computational Linguistics.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Jianlin Su, Ahmed Murtadha, Shengfeng Pan, Jing Hou, Jun Sun, Wanwei Huang, Bo Wen, and Yunfeng Liu. 2022. Global pointer: Novel efficient span-based approach for named entity recognition. *CoRR*, abs/2208.03054.

Chuanqi Tan, Wei Qiu, Mosha Chen, Rui Wang, and Fei Huang. 2020. Boundary enhanced neural span classification for nested named entity recognition. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 9016–9023. AAAI Press.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. *Advances in neural information processing systems*, 28.

David Wadden, Ulme Wennberg, Yi Luan, and Hannaneh Hajishirzi. 2019. Entity, relation, and event extraction with contextualized span representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 5783–5788. Association for Computational Linguistics.

Shuohang Wang and Jing Jiang. 2017. Machine comprehension using match-lstm and answer pointer. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Junjie Yang, Zhuosheng Zhang, and Hai Zhao. 2021. Multi-span style extraction for generative reading comprehension. In *Proceedings of the Workshop on Scientific Document Understanding co-located with 35th AAAI Conference on Artificial Inteligence, SDU@AAAI 2021, Virtual Event, February 9, 2021*, volume 2831 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013. Answer extraction as sequence tagging with tree edit distance. In *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 858–867. The Association for Computational Linguistics.

Deming Ye, Yankai Lin, Jiaju Du, Zhenghao Liu, Peng Li, Maosong Sun, and Zhiyuan Liu. 2020. Coreferential reasoning learning for language representation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 7170–7186. Association for Computational Linguistics.

Wonjin Yoon, Richard Jackson, Aron Lagerberg, and Jaewoo Kang. 2022. Sequence tagging for biomedical extractive question answering. *Bioinform.*, 38(15):3794–3801.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Juntao Yu, Bernd Bohnet, and Massimo Poesio. 2020. Named entity recognition as dependency parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 6470–6476. Association for Computational Linguistics.

Zheng Yuan, Chuanqi Tan, Songfang Huang, and Fei Huang. 2022. Fusing heterogeneous factors with triaffine mechanism for nested named entity recognition. In *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 3174–3186. Association for Computational Linguistics.

Zhuosheng Zhang and Hai Zhao. 2022. Tracing origins: Coreference-aware machine reading comprehension. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 1281–1292. Association for Computational Linguistics.

Ming Zhu, Aman Ahuja, Da-Cheng Juan, Wei Wei, and Chandan K. Reddy. 2020. Question answering with long multiple-span answers. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 3840–3849. Association for Computational Linguistics.