Multi-paragraph Reading Comprehension with Token-level Dynamic Reader and Hybrid Verifier

1st Yilin Dai, 2nd Qian Ji, 3rd Gongshen Liu*, 4th Bo Su*
School of Electronic Information and Electrical Engineering,
Shanghai Jiao Tong University,
Shanghai, China
{lydai1108, jeicy_good, lgshen, subo}@sjtu.edu.cn

Abstract—Multi-paragraph reading comprehension requires the model to infer answers of arbitrary user-generated questions by reasoning across-passage information. Previous work usually generates answer by directly employing a pointer network to predict the start and end position of the answer. However, span-level reading is insufficient since intermediate words may matter more. In this paper, we propose a novel unified network that includes a selector, a Token-level dynamic reader, and a Hybrid verifier (TH-Net). The core of token-level dynamic reader is a gate mechanism which dynamically selects important intermediate words according to boundary words. We decide the reader score from each token being both the boundary and the content. Moreover, we adopt a hybrid network verifier considering semantic answer-answer and entailment question-answer relationships to robust the model in case of being fooled by adversarial answers. Our experiments on SQuAD-document, SQuAD-open, and Trivia-wiki datasets show significant and consistent improvement as compared to other baselines and achieve the state-of-the-art performance on two of them.

Index Terms—reading comprehension, TH-Net, token-level dynamic reader, hybrid verifier

I. INTRODUCTION

Machine reading comprehension (MRC) models empower machines to answer user-generated questions by comprehending textual data. In real-world scenarios, passage may be extended and include both relevant and irrelevant content. Since multi-paragraph MRC task being more applicable, our method focuses on addressing the challenges coming with document-level data instead of single-paragraph data.

Existing multi-paragraph MRC models can be divided into two basic approaches: pipelined methods and unified methods. Both typically consist of a paragraph selector for choosing relevant paragraphs from document, a paragraph reader for extracting answers from chosen paragraphs, and an answer verifier for ruling out noisy answers. In pipelined approaches [1], [2], these three components are considered separate and trained independently, but high-quality upstream outputs may not necessarily benefit downstream modules. Unified methods directly apply the model to the input and return the answer with the highest score [3]. Three components share the same contextualized text representation and optimize simultaneously in a joint learning method [4], [5]. This paper adopts unified model to avoid inconsistent performance across different components. Recently, some work replaces word and character embeddings [6] with outputs from pre-trained language models (LMs) to get deeper word representations [7], [8], [9]. Our model follows this approach, but is fine-tuned in training.

Among three modules mentioned above, paragraph reader undoubtedly plays the most pivotal part because of the following reasons. Paragraph selector can be regarded as a binary classification task which can show good results by a linear network. To some extent, it exists to fulfill the document-level data input requirements [10] because ruling out irrelevant paragraphs at first can avoid problems of out-of-memory (OOM) in paragraph reading. Answer verifier also has impact on the performance because unified models without verifier [11], [12] can be easily fooled by adversarial examples [13]. However, paragraph reader remains fundamental to MRC models.

To sequence generating task which specifically selects only a member of the input sequence as the output, pointer network [14] is normally adopted to avoid generating words not known as prior. In MRC task, we usually apply it after question and context embeddings to predict the probability of each token being the start and the end of answer [15], [16]. Although many deep neural networks such as Bi-LSTM [4] and Bi-GRU [5] that can extract deep query-aware context information have brought considerable progress in the performance of reader, few has considered the direct application of pointer network.
to span-extracting task. Only boundary word scores are not enough to measure the full validity and legitimacy of predicted answers. We found that wrong answers may have the same boundary words but different intermediate words with correct ones under many circumstances, which is shown in Fig. 1(a). So we consider taking answer content into consideration in paragraph reader can raise model performance.

In answer verifier, most modules rule out noisy candidate answers which have lower correlation degree with the question. It is obvious that higher their relevance is, higher the possibility of the answer being correct. Another observation can be utilized is that many candidate answers tend to contain the same words and look similarly, as is shown in Fig. 1(b). The correct answer has evidence content that can match both relationships, while others cannot, so we think additionally mining deeper semantic information between candidate answers themselves may also help determine the final answer.

Problems mentioned above both arise from the situation when the model needs to differentiate similar candidate answers, so we propose token-level dynamic reader and hybrid verifier in vanilla unified MRC models to avoid boundary and content similarity. In token-level dynamic reader, we combine the probability of each token being the answer content with span-level prediction. In specific, we adopt a gate mechanism to automatically extract semantic information of intermediate tokens and select important ones. We then replace the representations of these tokens with the self-attention output performed over themselves. After that, another linear network is followed to predict whether each word should be contained in the answer. Finally, we generate candidate answers based on the sum of answer start, between, and end scores. To make this answer between score play an appropriate role, we introduce an auxiliary weighted between loss to help it fuse with span-level prediction. In specific, we adopt a gate mechanism to automatically extract semantic information of intermediate tokens and select important ones. We then replace the representations of these tokens with the self-attention output performed over themselves. After that, another linear network is followed to predict whether each word should be contained in the answer. Finally, we generate candidate answers based on the sum of answer start, between, and end scores. To make this answer between score play an appropriate role, we introduce an auxiliary weighted between loss to help it fuse with span-level prediction.

In answer verifying, we adopt a hybrid network which combines correlated semantic relationship between candidate answers with entailment relationship between question and answer. This mechanism also raises the performance of our verifier.

II. METHODS

The overall framework of TH-Net is demonstrated in Fig. 2 where one question and several paragraphs are given as the input and the final answer is returned. TH-Net is initialized by pre-trained LM and fine-tuned during training, with three major modules.

A. Segmentation and Encoding

This layer encodes the input sequence by several transformer blocks and computes a deep and context-aware representation for each token. Before encoding, we concatenate all the input paragraphs to a new document and split it to several segments. Specifically, we produce \( N \) paragraph segments using a sliding window of length \( l \) and stride \( r \) over the new document following [17]. Here we define the input sequence as obtained by packing each paragraph \( P_i \) and its corresponding question \( Q \), with length \( L_x = L_p + L_q + 3 \), i.e.,

\[
S_i = [<CLS>, Q, <SEP>, P_i, <SEP>]
\]

where \( L_p \) and \( L_q \) are the length of input paragraph and question. Following [18], we take token \(<CLS> \) for classification but will not be used in our paper and \(<SEP> \) separating question and paragraph. The input representation for the \( j^{th} \) token in sequence \( S_i \) is constructed as:

\[
h_{ij}^0 = s_{ij}^{tok} + s_{ij}^{pos} + s_{ij}^{seg}
\]
where \( s_{ij}^{tok}, s_{ij}^{pos}, \) and \( s_{ij}^{seg} \) are the token, position, and segment embeddings separately. In detail, tokens with the same position share the same position embedding. Besides, all the tokens in question \( Q \) and paragraph \( P_i \) share the same segment embedding respectively. The input sequence is then fed into \( L \) successive transformer encoder blocks to generate deep and context-aware representations. The output for the \( j \)th token in sequence \( S_i \) is shown in (3). For the details of transformer block, we refer readers to [19].

\[
h_{ij} = \text{TransformerBlock}(h_{ij}^{l-1}), l = 1, 2, \ldots, L \quad (3)
\]

### B. Paragraph Selector

In multi-paragraph MRC, the golden answer usually comes from one paragraph or even a small set of sentences [20], so we annotate which paragraph contains the answer in a distantly supervised setup. Here, we introduce paragraph selector to select top \( S \) paragraphs. Since reading and generating several candidate answers for each paragraph may cause OOM problems, so we use the hidden states of the first \( L' \) transformer blocks \( h_{ij}^{L'} = \{h_{ij}^{L'}\}_{j=1}^{L} \) (\( h_{ij}^{L'} \in \mathbb{R}^h \), \( h \) refers to the hidden state dimension) as the input of paragraph selector. For each input sequence, we self-align the deep semantic representation learnt afore-mentioned to obtain a weighted sequence vector \( \tilde{h}_{ij}^{L'} \) followed by a linear projection layer with activation function for a selector score \( p_i \in \mathbb{R} \) as:

\[
\mu_i = \text{softmax}(w_p^\top h_{ij}^{L'}) \quad (4)
\]

\[
\tilde{h}_{ij}^{L'} = \sum_{j=1}^{L} \mu_i h_{ij}^{L'} \quad (5)
\]

\[
p_i = w_p^\top \text{tanh}(W_p \tilde{h}_{ij}^{L'}) \quad (6)
\]

where trainable parameters \( w_p \) and \( w_r \) are vectors; \( W_p \) is a bilinear projection matrix that matches two vectors in the same space. We then normalize \( p_i \) and optimize the following objective function:

\[
\mathcal{L}_{PS} = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log PS(p_i) + (1 - y_i)(1 - \log \text{softmax}(p_i))] \quad (7)
\]

where label \( y_i = 1 \) means \( S_i \) contains one golden answer, otherwise \( y_i = 0 \). Here, \( \log \text{softmax} \) refers to \( \log \text{softmax} \) function.

### C. Token-level Dynamic Paragraph Reader

As the core module of TH-Net, this layer is designed to comprehend \( S \) paragraphs from selector and return \( M \) candidate answers for each paragraph with scores calculated on token-level instead of span-level. Different from paragraph selector, we take the output of \( L \) transformer blocks as the input since deeper neural networks with attention-mechanism are supposed to capture more complex and useful linguistic phenomena. Following previous work [15], the probability of each word being the answer start position \( r^s_i \in \mathbb{R}^L \) and end position \( r^e_i \in \mathbb{R}^L \) can be easily obtained by applying a linear projection layer with activation function after hidden states \( h_i^L \in \mathbb{R}^{L_x \times h} \) as follows:

\[
r^s_i = w^s_i h_i^L, r^e_i = w^e_i h_i^L \quad (8)
\]

where \( w_s \) and \( w_e \) are vectors to be trained.

In order to make full use of each token, we additionally introduce a token-level dynamic network to obtain answer between score indicating the probability of each word being the content of the answer, detailed in Fig. 3. Our token-level reader is dynamic because it can automatically select important tokens according to the boundary tokens. Here, we define a token important if it includes necessary information to decide a candidate answer correct or not. To distinguish different roles of boundary and intermediate tokens play, we use new hidden states \( h_i^L \) obtained by adding two linear projection layers with activation function to \( h_i^L \):

\[
\tilde{h}_{ij}^L = \text{sigmoid}(w_b^\top \text{relu}(W_b^\top h_i^L)) \quad (9)
\]

where \( w_b \) and \( W_b \) are trainable parameters.

A gate mechanism is used to select most important \( K \) words \( g_i^L \in \mathbb{R}^{K \times h} \) according to \( \tilde{h}_{ij}^L \). \( K \) changes along with the length of answer. Attended output matrix \( g_i^L \) is obtained by performing scaled dot-product self-attention mechanism over chosen important tokens and position-wise pad is used between \( \tilde{h}_{ij}^L \) and \( g_i^L \) to get the probability of each token being the content of answer \( b_i \in \mathbb{R}^{L_x} \):

\[
g_i^L = \text{softmax}(Q_g K_g^T \sqrt{h}) V_g \quad (10)
\]

\[
b_i = w_p^\top \text{pad}(\tilde{h}_{ij}^L; g_i^L) \quad (11)
\]

where query \( Q_g \in \mathbb{R}^{K \times h} \), \( K_g \in \mathbb{R}^{K \times h} \), and value \( V_g \in \mathbb{R}^{K \times h} \) are linear projections of \( g_i^L \); \( w_p \) is a trainable vector.

In a distantly supervised setup, we label all text spans that match the answer text as being correct [5], thus yielding start and end label vectors for each input sequence as \( y_i^s \in \mathbb{R}^{L_x} \) and \( y_i^e \in \mathbb{R}^{L_x} \). Besides, we also label all the words between
The sum objective function can be defined as follows:

\[ \mathcal{L} = \frac{1}{N} \sum_{m=1}^{M} \sum_{i=1}^{N} y_{ij}^m \mathbf{S}(r_{ij}^m) \]

where \( y_{ij}^m \) is the ground truth label; \( y_{ij}^m \in \mathbb{R}^M \) is the answer length dimension to be the same shape as \( \mathbf{S} \).

When predicting the final answer, we first calculate selector score for each input sequence and choose top-\( S \) answers with corresponding input sequence as attention weights and then normalize it to attended vectors as follows:

\[ \tilde{a}_{im}^2 = \text{softmax} \left( \frac{Q_a K_a^T}{\sqrt{h}} \right) V_a \]

where query \( Q_a \in \mathbb{R}^{L_m \times h} \) is linear projection of the answer representation; key \( K_a \in \mathbb{R}^{L_x \times h} \) and value \( V_a \in \mathbb{R}^{L_x \times h} \) are linear projections of the input sequence. Here, \( L_m \) refers to the maximum candidate answer length.

The verify score \( v_i \in \mathbb{R}^M \) is calculated by concatenating the output of both models followed by a linear projection layer. Before concatenating, \( \tilde{a}_{im}^2 \) will do mean function over the answer length dimension to be the same shape as \( \tilde{a}_{im}^1 \).

\[ v_i^m = W_v \tanh(W_c[\tilde{a}_{im}^1; \tilde{a}_{im}^2]) \]

where \( W_v \) and \( W_c \) are trainable parameters; \( y_i \in \mathbb{R}^M \) is the ground truth label; \([;] \) refers to matrix concatenation.

### E. Joint Training and Prediction

According to the design described above, we train these modules together as multi-task learning [4] with a joint objective function formulated as follows:

\[ \mathcal{L} = \mathcal{L}_{PS} + \mathcal{L}_{PR} + \mathcal{L}_{AV} \]

When predicting the final answer, we first calculate selector score for each input sequence and choose top-\( S \) paragraphs. Then for each paragraph, we generate \( M \) candidate answers with both boundary score and content score. Content score is calculated by using a dynamic gate mechanism which particularly considers important words in answer span. We also prune noisy candidate answers through a hybrid verifier. Therefore, the final score for \( m^{th} \) candidate answer of \( i^{th} \) input sequence can be calculated by considering selector score, reader score, and verifier score with different weights:

\[ \text{score}_i^m = \eta_1 p_i + \eta_2 (r_i^s + r_i^e + b_i)^m + \eta_3 v_i^m \]

where \( p_i \) means the score of \( i^{th} \) paragraph containing the answer; \( r_i^s + r_i^e + b_i \) refer to the answer boundary and content score respectively; \( v_i^m \) represents the confidence of \( m^{th} \) answer being correct; \( \eta_1, \eta_2, \) and \( \eta_3 \) are hyper-parameters control the weights of these three components.

### III. Experiments

#### A. Datasets

We experiment on three well-studied open extractive MRC datasets: SQuAD-document, a variant of SQuAD [23] that includes a collection of crowdsourced questions and a full Wikipedia article for each question; SQuAD-open [24], the same dataset but pair each question with the entire Wikipedia domain; Trivia-wiki [25], a dataset of questions from trivia databases associated with Wikipedia articles by completing a web search of the questions. All datasets use Exact Match (EM) accuracy and (marco-averaged) F1 score as the evaluation metrics.
### B. Experiment Setups

**Data Sampling** Before training, we first sample several paragraphs for each question by TF-IDF [5], a traditional information retrieval method measuring the distance between the question and paragraph. It is conducted between the textual metadata of question and paragraphs from the same document, including the document title and main content. Besides, other features such as the recall ratio of the question words from the paragraph are also considered to indicate the relevance. As a result, we can obtain several paragraphs relevant to the question to satisfy the multi-paragraph setting in this paper. For SQuAD-document, we use the top 4 paragraphs, and for Trivia-wiki we use the top 8 because much more instance is given. Besides, we also merge consecutive paragraphs in Trivia-wiki to a maximum of 400 words as in [5].

**Implementation Details** Our model is initialized by a publicly available uncased base version of BERT, so we set the input sequence length \( L_x \) as 384, stride \( r \) as 128. We choose the number of transformer blocks for selector \( L' \) as 3, and for reader and verifier \( L \) as 12. The number of selected paragraphs \( S \) and candidate answers \( M \) are the key factors to balance the effectiveness and efficiency tradeoff. We choose \( S=4 \) and \( M=10 \) for the good performance when evaluating on the dev set. We optimize the model by Adam optimizer for finetuning 2 epochs, with the minibatch size as 16 and the initial learning rate as 0.0005. During training, we set the weights of intermediate words \( \lambda_2 \) in (14) as 0.2 and 0.1 for SQuAD-document and Trivia-wiki respectively. During inference, we tune the weights for three major modules, and set \( \eta_1 \) as 1.4, \( \eta_2 \) as 1.2, and \( \eta_3 \) as 1.

**Comparison Setting** We start with a baseline following the model used in [15]. We also take a pipelined BERT finetuned by datasets sampled in this paper as a direct comparison. Our BERT follows exactly the same design as the original paper [18]. Besides, we further take several top-ranked systems on each dataset as additional comparisons (will be detailed in §4).

### IV. Results and Analysis

#### A. Experiment results

The results of TH-Net on SQuAD-document and SQuAD-open are summarized in Table I and Table II. We can see that by adopting the proposed method, our model achieves 77.51 EM and 84.29 F1, outperforming the previous SOTA methods. Note that the BERT\textsubscript{BASE} has obtained only 76.39 F1, which is 7.9 lower than us and validates the effectiveness of combing selector, reader, and verifier in a unified method to address multi-paragraph MRC task. TH-Net also outperforms RE\textsuperscript{3}QA which adopts a unified architecture similar to our model by 1.8 EM and 1.63 F1, proving that token-level dynamic reader and hybrid verifier have considerable impact on the performance boost. Besides, we also run experiments on SQuAD-open dataset, and TH-Net surpasses a BERT baseline by 2.34 F1 and RE\textsuperscript{3}QA by 1.27 F1. The performance boost is not so obvious as in SQuAD-document maybe because questions in open scenario situations are more independent and paying more attention to context-question relationship is not so effective.

We additionally evaluate our model on Trivia-wiki and the result is shown in Table III. As we can see, TH-Net achieves 68.55 EM and 72.77 F1, outperforming the previous methods. However, the score of 76.45 EM and 79.38 F1 on the verified version is lower than the SOTA performance of 76.7 EM and 79.9 F1 in [26], which implies our model still can be improved.

#### B. Discussion

**Ablation study** To get better insight into our model architecture, we conduct an in-depth ablation study on SQuAD-

### TABLE I  
**Performance of TH-Net on SQuAD-document dataset.** The results are reported on the dev set.

<table>
<thead>
<tr>
<th>Model</th>
<th>SQuAD-document EM</th>
<th>SQuAD-document F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline [15]</td>
<td>60.59</td>
<td>66.87</td>
</tr>
<tr>
<td>S-Norm [5]</td>
<td>64.08</td>
<td>72.37</td>
</tr>
<tr>
<td>BERT\textsubscript{BASE} [18]</td>
<td>68.32</td>
<td>76.39</td>
</tr>
<tr>
<td>RE\textsuperscript{3}QA\textsubscript{BASE} [12]</td>
<td>75.71</td>
<td>82.66</td>
</tr>
<tr>
<td>TH-Net</td>
<td><strong>77.51</strong></td>
<td><strong>84.29</strong></td>
</tr>
</tbody>
</table>

### TABLE II  
**Performance of TH-Net on SQuAD-open dataset.** The results are reported on the dev set.

<table>
<thead>
<tr>
<th>Model</th>
<th>SQuAD-open EM</th>
<th>SQuAD-open F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>MINIMAL [20]</td>
<td>34.72</td>
<td>42.53</td>
</tr>
<tr>
<td>Multi-Step [26]</td>
<td>31.95</td>
<td>39.29</td>
</tr>
<tr>
<td>BERT\textsubscript{SERN} [9]</td>
<td>38.61</td>
<td>46.15</td>
</tr>
<tr>
<td>RE\textsuperscript{3}QA\textsubscript{BASE} [12]</td>
<td>38.54</td>
<td>47.22</td>
</tr>
<tr>
<td>TH-Net</td>
<td><strong>40.18</strong></td>
<td><strong>48.49</strong></td>
</tr>
</tbody>
</table>

### TABLE III  
**Performance of TH-Net on Trivia-wiki dataset.** The results are reported on the test set.

<table>
<thead>
<tr>
<th>Model</th>
<th>Trivia-wiki Full EM</th>
<th>Trivia-wiki Full F1</th>
<th>Trivia-wiki Verified EM</th>
<th>Trivia-wiki Verified F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline [15]</td>
<td>40.32</td>
<td>45.91</td>
<td>44.86</td>
<td>50.71</td>
</tr>
<tr>
<td>Smartnet [27]</td>
<td>42.41</td>
<td>48.84</td>
<td>50.51</td>
<td>55.90</td>
</tr>
<tr>
<td>Re-Ranker [1]</td>
<td>46.94</td>
<td>52.85</td>
<td>62.83</td>
<td>70.68</td>
</tr>
<tr>
<td>S-Norm [5]</td>
<td>63.99</td>
<td>68.93</td>
<td>67.98</td>
<td>72.88</td>
</tr>
<tr>
<td>SLQA [2]</td>
<td>66.59</td>
<td>70.46</td>
<td>74.83</td>
<td>77.78</td>
</tr>
<tr>
<td>TH-Net</td>
<td><strong>68.55</strong></td>
<td><strong>72.77</strong></td>
<td><strong>76.45</strong></td>
<td><strong>79.38</strong></td>
</tr>
</tbody>
</table>

### TABLE IV  
**Comparison of TH-Net with different individual components on SQuAD-document and Trivia-wiki.**

<table>
<thead>
<tr>
<th>Model</th>
<th>SQuAD-document EM</th>
<th>SQuAD-document F1</th>
<th>Trivia-wiki EM</th>
<th>Trivia-wiki F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Complete Model</td>
<td><strong>77.51</strong></td>
<td><strong>84.29</strong></td>
<td><strong>68.55</strong></td>
<td><strong>72.77</strong></td>
</tr>
<tr>
<td>- selector</td>
<td>76.07</td>
<td>83.50</td>
<td>67.27</td>
<td>71.64</td>
</tr>
<tr>
<td>- token-level reader</td>
<td>76.49</td>
<td>83.81</td>
<td>68.26</td>
<td>72.50</td>
</tr>
<tr>
<td>- hybrid-network verifier</td>
<td>76.98</td>
<td>84.05</td>
<td>67.84</td>
<td>72.25</td>
</tr>
<tr>
<td>- model I</td>
<td>76.80</td>
<td>83.85</td>
<td>68.03</td>
<td>72.30</td>
</tr>
<tr>
<td>- model II</td>
<td>76.84</td>
<td>83.93</td>
<td>68.10</td>
<td>72.42</td>
</tr>
</tbody>
</table>
document and Trivia-wiki in Table IV. To ablate selector, we choose paragraphs based on TF-IDF scores. To evaluate token-level dynamic reader, we generate answers only based on span-level scores. The performance of hybrid verifier is shown by selecting answers only considering the selector and reader scores. Ablating selector degrades the performance by 0.27 F1, indicating that the intermediate-word mechanism can improve the performance of span-level reader. Ablating the hybrid verifier, on the other hand, causes little influence on F1 by 0.24 and 0.52 but big on EM by 1.44 and 1.28. This suggests that the hybrid network can elevate the general performance by outputting more precise answers.

Besides, we also measure the performance of our proposed approaches as the model is used to independently process an increasing number of paragraphs, which is shown in Fig. 5 and Fig. 6. We can observe that with more paragraphs to be dealt with, all curves become stable showing it does a passable job at focusing on the correct paragraph. Moreover, the token-level dynamic reader and hybrid verifier do have effect on the model performance boost. In SQuAD-document, reader plays a bigger part, while verifier becomes more important in Trivia-wiki when dealing with more paragraphs. In both datasets, the token-level dynamic reader and hybrid verifier lead to a significant improvement, and selector is even better.

**Effect of token-level dynamic reader** We assess whether our reader raise the performance with different weights of intermediate word scores being used, which is detailed in Table V. We notice that F1 score reaches the peak when the weight of between-word score is 0.2 for SQuAD-document and 0.1 for Trivia-wiki. This phenomenon is possibly caused by the fact that answer instances in Trivia-wiki dataset are much shorter and contain less words, thus making boundary words outperform intermediate words and increasing between-word weight will instead degrade the performance.

**Effect of hybrid verifier** We discuss the effect of model I and model II in hybrid verifier separately and report the result also in Table IV. As we can see, removing model I results in a worse performance drop by 0.44 F1 and 0.47 F1 compared to removing model II by 0.36 F1 and 0.35 F1 in SQuAD-document and Trivia-wiki respectively. It indicates that semantic relationship between answers matters more than entailment relationship between answer and the input sequence. This occurred probably because the reader has mastered enough semantic information of both question and paragraph, which also validates the effectiveness of our token-level dynamic reader for generating high-quality candidate answers.

**Case study** We conduct a case study to demonstrate how each module takes effect with the same example discussed in §1 and compare it with BERT. For each candidate answer, we list three scores predicted by the selector, reader, and verifier in Table VI. In specific, reader scores include scores calculated on both span-level and token-level. For the first question, top-3 ranked candidate answers all begin with “a” and end with “system”, with close selector, boundary reader, and verifier scores. It is very difficult to choose the correct answer without considering the content, but it proves that token-level reader can benefit this kind of question by concentrating on important intermediate words. Although all candidate answers have the
same boundary words, the correct one still can be chosen by determining the key intermediate word “two-phased” outperforming “complete” and “integrated”. Similarly, the second candidate answer of the second question is preferred by the verifier component, thus being returned as the final answer. It proves that our hybrid verifier can be effective when the selector and reader module make an incorrect decision among the confusing answer candidates. By taking all the four scores into consideration, our model can correctly predict the answer.

V. RELATED WORK

A. Reading Comprehension Datasets

In the last few years, the SOTA performance in MRC has been rapidly advanced, in no small part because of the creation of many datasets. Earlier cloze-style task [28] requires the system to predict a held out word from a piece of text. Other datasets including SQuAD [23], [29], TriviaQA [25], and WikiReading [30] provide problems under more realistic scenario. However, none of these datasets can fulfill the multi-paragraph requirement in this paper, so we generate examples by retrieving passages for existing questions based on TF-IDF [5]. We choose to work on SQuAD and Trivia datasets considering they are more widely studied.

B. Pre-trained Language Models

More recently, many pre-trained LMs such as BERT [18] and XLNet [31] have caused a stir in NLP. LMs pre-trained on substantial unlabeled data with deep neural networks and attention-mechanism can bring deep and complex linguistic contextual representations of text, which greatly boost the performance of more than twenty language processing tasks including MRC. In this paper, we employ a unified network with three major modules also built upon pre-trained LMs, but in fine-tuned instead of feature-based way.

C. Neural Reading Comprehension

Mainstream MRC architectures including selector, reader, and verifier are realized in pipelined or unified ways [1], [12]. Since document is much longer than question compared with paragraph, we are sure to lose many useful information if we summarize each document into a fixed-sized vector and do attention between the document and question. So it is necessary to first use a selector to extract relevant paragraph content and do comprehension after that. Recent work [32] also adopts a

---

**TABLE VI**

Scores predicted by TH-Net for examples mentioned in §1, with correct answers in bold.

<table>
<thead>
<tr>
<th>Question &amp; Candidate Answers</th>
<th>Selector</th>
<th>Reader</th>
<th>Verifier</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Q1: What system did Tesla recommend to Niagara Falls in 1893?</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A1: a two-phased system</td>
<td>Selector Scores: 0.512</td>
<td>Boundary Scores: 5.696</td>
<td>Content Scores: 5.332</td>
</tr>
<tr>
<td>A2: a completely integrated AC system</td>
<td>Selector Scores: 0.528</td>
<td>Boundary Scores: 5.914</td>
<td>Content Scores: 1.256</td>
</tr>
<tr>
<td>A3: a complete AC system</td>
<td>Selector Scores: 0.490</td>
<td>Boundary Scores: 7.961</td>
<td>Content Scores: 3.585</td>
</tr>
<tr>
<td><strong>Q2: What are the main sources of primary law?</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A1: primary law, secondary law and supplementary law</td>
<td>Selector Scores: 0.763</td>
<td>Boundary Scores: 5.342</td>
<td>Content Scores: 9.183</td>
</tr>
<tr>
<td>A3: the founding treaties</td>
<td>Selector Scores: 0.306</td>
<td>Boundary Scores: 6.106</td>
<td>Content Scores: 7.235</td>
</tr>
</tbody>
</table>

---

In this paper, we proposed TH-Net, a novel unified architecture accomplishing multi-paragraph MRC task. TH-Net includes paragraph selector, token-level dynamic reader, and hybrid verifier sharing the same context representations initialized by BERT$_{BASE}$ and fine-tuned during training. The proposed approach has the advantages of comprehending paragraphs on token-level effectively and combining semantic information between answers with entailment information between answer and the input sequence. Our method outperforms the pipelined and unified baselines on three challenging datasets: SQuAD-document, SQuAD-open, and Trivia-wiki and achieves the SOTA performance on two of them.
ACKNOWLEDGMENT

This research work has been funded by the National Natural Science Foundation of China (Grant No.61772337), the National Key Research and Development Program of China NO. 2018YFC0832004.

REFERENCES


