Retrieve, Read, Rerank: Towards End-to-End Multi-Document Reading Comprehension

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

Abstract

This paper considers the reading comprehension task in which multiple documents are given as input. Prior work has shown that a pipeline of retriever, reader, and reranker can improve the overall performance. However, the pipeline system is inefficient since the input is re-encoded within each module, and is unable to leverage upstream components to help downstream training. In this work, we present RE^{3}QA, a unified question answering model that combines context retrieving, reading comprehension, and answer reranking to predict the final answer. Unlike previous pipelined approaches, RE^{3}QA shares contextualized text representation across different components, and is carefully designed to use high-quality upstream outputs (e.g., retrieved context or candidate answers) for directly supervising downstream modules (e.g., the reader or the reranker). As a result, the whole network can be trained end-to-end to avoid the context inconsistency problem. Experiments show that our model outperforms the pipelined baseline and achieves state-of-the-art results on two versions of TriviaQA and two variants of SQuAD.

1 Introduction

Teaching machines to read and comprehend text is a long-term goal of natural language processing. Despite recent success in leveraging reading comprehension (RC) models to answer questions given a related paragraph (Wang et al., 2017; Hu et al., 2018; Yu et al., 2018), extracting answers from documents or even a large corpus of text (e.g., Wikipedia or the whole web) remains to be an open challenge. This paper considers the multi-document RC task (Joshi et al., 2017), where the system needs to, given a question, identify the answer from multiple evidence documents. Unlike single-pargraph settings (Rajpurkar et al., 2016), this task typically involves a *retriever* for selecting few relevant document content (Chen et al., 2017), a *reader* for extracting answers from the retrieved context (Clark and Gardner, 2018), and even a *reranker* for rescoring multiple candidate answers (Bogdanova and Foster, 2016).

Previous approaches such as DS-QA (Lin et al., 2018) and R^3 (Wang et al., 2018a) consist of separate retriever and reader models that are jointly trained. Wang et al. (2018d) further propose to rerank multiple candidates for verifying the final answer. Wang et al. (2018b) investigate the full retrieve-read-rerank process by constructing a pipeline system that combines an information retrieval (IR) engine, a neural reader, and two kinds of answer rerankers. Nevertheless, the pipeline system requires re-encoding inputs for each subtask, which is inefficient for large RC tasks. Moreover, as each model is trained independently, high-quality upstream outputs can not benefit downstream modules. For example, as the training proceeds, a neural retriever is able to provide more relevant context than an IR engine (Htut et al., 2018). However, the reader is still trained on the initial context retrieved using IR techniques. As a result, the reader could face a *context inconsistency* problem once the neural retriever is used. Similar observation has been made by Wang et al. (2018c), where integrating both the reader and the reranker into a unified network is more benefical than a pipeline (see Table 1 for more details).

In this paper, we propose RE^{3}QA, a neural question answering model that conducts the full **re**trieve-**re**ad-**re**rank process for multi-document RC tasks. Unlike previous pipelined approaches that contain separate models, we integrate an early-stopped retriever, a distantly-supervised reader, and a span-level answer reranker into a unified network. Specifically, we encode segments of text with pre-trained Transformer blocks (Devlin

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 2285–2295
Florence, Italy, July 28 - August 2, 2019.

Model	Retrieve	Read	Rerank	Architecture
DS-QA (Lin et al., 2018)	✓	✓	✗	Pipeline
R^3 (Wang et al., 2018a)	✓	✓	✗	Pipeline*
Extract-Select (Wang et al., 2018d)	✗	✓	✓	Pipeline*
V-Net (Wang et al., 2018c)	✗	✓	✓	Unified
Re-Ranker (Wang et al., 2018b)	✓	✓	✓	Pipeline
RE^{3}QA	✓	✓	✓	Unified

Table 1: Comparison of RE^{3}QA with existing approaches. Our approach performs the full retrieve-read-rerank process with a unified network instead of a pipeline of separate models. *: R^3 and Extract-Select jointly train two models with reinforcement learning.

et al., 2018), where earlier blocks are used to predict retrieving scores and later blocks are fed with few top-ranked segments to produce multiple candidate answers. Redundant candidates are pruned and the rest are reranked using their span representations extracted from the shared contextualized representation. The final answer is chosen according to three factors: the retrieving, reading, and reranking scores. The whole network is trained end-to-end so that the context inconsistency problem can be alleviated. Besides, we can avoid re-encoding input segments by sharing contextualized representations across different components, thus achieving better efficiency.

We evaluate our approach on four datasets. On TriviaQA-Wikipedia and TriviaQA-unfiltered datasets (Joshi et al., 2017), we achieve 75.2 F1 and 71.2 F1 respectively, outperforming previous best approaches. On SQuAD-document and SQuAD-open datasets, both of which are modified versions of SQuAD (Rajpurkar et al., 2016), we obtain 14.8 and 4.1 absolute gains on F1 score over prior state-of-the-art results. Moreover, our approach surpasses the pipelined baseline with faster inference speed on both TriviaQA-Wikipedia and SQuAD-document. Source code is released for future research exploration[1].

2 Related Work

Recently, several large datasets have been proposed to facilitate the research in document-level reading comprehension (RC) (Clark and Gardner, 2018) or even open-domain question answering (Chen et al., 2017). TriviaQA (Joshi et al., 2017) is a challenging dataset containing over 650K question-answer-document triples, in which the document are either Wikipedia articles or web pages. Quasar-T (Dhingra et al., 2017) and SearchQA (Dunn et al., 2017), however, pair each question-answer pair with a set of web page snippets that are more analogous to paragraphs. Since this paper considers the multi-document RC task, we therefore choose to work on TriviaQA and two variants of SQuAD (Rajpurkar et al., 2016).

To tackle this task, previous approaches typically first retrieve relevant document content and then extract answers from the retrieved context. Choi et al. (2017) construct a coarse-to-fine framework that answers the question from a retrieved document summary. Wang et al. (2018a) jointly train a ranker and a reader with reinforcement learning (Sutton and Barto, 2011). Lin et al. (2018) propose a pipeline system consisting of a paragraph selector and a paragraph reader. Yang et al. (2019) combine BERT with an IR toolkit for open-domain question answering.

However, Jia and Liang (2017) show that the RC models are easily fooled by adversarial examples. By only extracting an answer without verifying it, the models may predict a wrong answer and are unable to recover from such mistakes (Hu et al., 2019). In response, Wang et al. (2018d) present an extract-then-select framework that involves candidate extraction and answer selection. Wang et al. (2018c) introduce a unified network for cross-passage answer verification. Wang et al. (2018b) explore two kinds of answer rerankers in an existing retrieve-read pipeline system. There are some other works that handle this task in different perspectives, such as using hierarchical answer span representations (Pang et al., 2019), modeling the interaction between the retriever and the reader (Das et al., 2019), and so on.

Our model differs from these approaches in several ways: (a) we integrate the retriever, reader,

[1] https://github.com/huminghao16/RE3QA

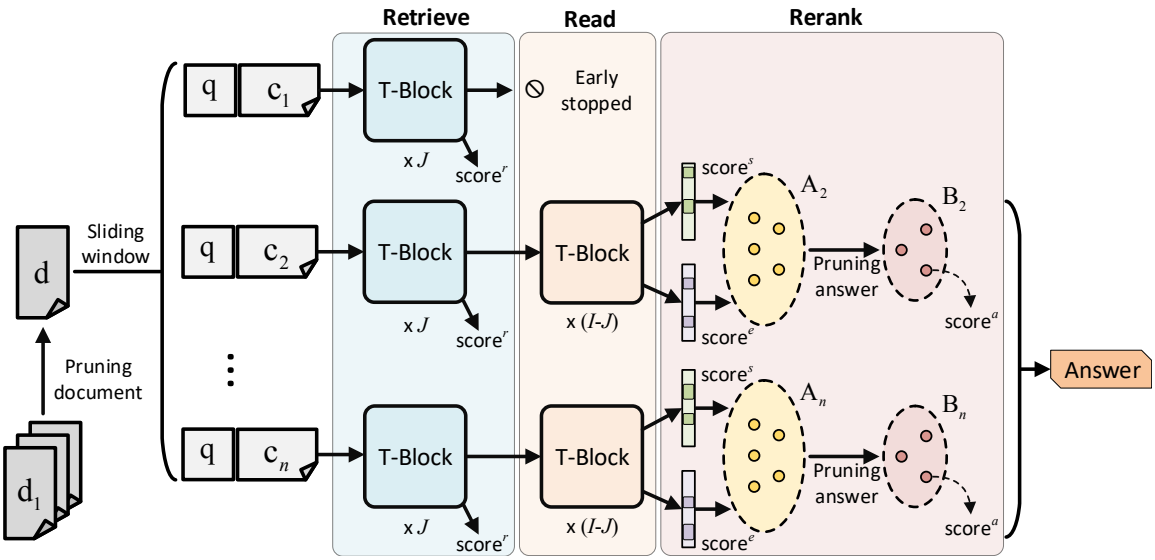

Figure 1: RE^{3}QA architecture. The input documents are pruned and splitted into multiple segments of text, which are then fed into the model[2]. Few top-ranked segments are retrieved and the rest are early stopped. Multiple candidate answers are proposed for each segment, which are later pruned and reranked. RE^{3}QA has three outputs per candidate answer: the retrieving, reading, and reranking scores. The network is trained end-to-end with a multi-task objective. "T-Block" refers to pre-trained Transformer block (Devlin et al., 2018).

and reranker components into a unified network instead of a pipeline of separate models, (b) we share contextualized representation across different components while pipelined approaches re-encode inputs for each model, and (c) we propose an end-to-end training strategy so that the context inconsistency problem can be alleviated.

A cascaded approach is recently proposed by Yan et al. (2019), which also combines several components such as the retriever and the reader while sharing several sets of parameters. Our approach is different in that we ignore the document retrieval step since a minimal context phenomenon has been observed by Min et al. (2018), and we additionally consider answer reranking.

3 RE^{3}QA

Figure 1 gives an overview of our multi-document reading comprehension approach. Formally, given a question and a set of documents, we first filter out irrelevant document content to narrow the search space (§3.1). We then split the remaining context into multiple overlapping, fixed-length text segments. Next, we encode these segments along with the question using pre-trained Transformer blocks (Devlin et al., 2018) (§3.2). To maintain efficiency, the model computes a retrieving score based on shallow contextual representations with early summarization, and only returns a few top-ranked segments (§3.3). It then continues encoding these retrieved segments and outputs multiple candidate answers under the distant supervision setting (§3.4). Finally, redundant candidates are pruned and the rest are reranked using their span representations (§3.5). The final answer is chosen according to the retrieving, reading, and reranking scores. Our model is trained end-to-end[3] by back-propagation (§3.6).

3.1 Document Pruning

The input to our model is a question $\mathbf{q}$ and a set of documents $\mathbf{D} = \{\mathbf{d}_1, ..., \mathbf{d}_{N_D}\}$. Since the documents could be retrieved by a search engine (e.g., up to 50 webpages in the unfiltered version of TriviaQA (Joshi et al., 2017)) or Wikipedia articles could contain hundreds of paragraphs, we therefore first discard irrelevant document content at paragraph level. Following Clark and Gardner (2018), we select the top-K paragraphs that have smallest TF-IDF cosine distances with each question. These paragraphs are then sorted according to their positions in the documents and concatenated to form a new pruned document $\mathbf{d}$. As a result, a large amount of unrelated text can be filtered out while a high recall is guaranteed. For example, nearly 95% of context are discarded while

[3]Note that "end-to-end training" only involves retrieving, reading, and reranking, but not the very first pruning step.

the chance of selected paragraphs containing correct answers is 84.3% in TriviaQA-unfiltered.

3.2 Segment Encoding

Typically, existing approaches either read the retrieved document at the paragraph level (Clark and Gardner, 2018) or at the sentence level (Min et al., 2018). Instead, following Hewlett et al. (2017), we slide a window of length l with a stride r over the pruned document $\mathbf{d}$ and produce a set of text segments $\mathbf{C} = \{\mathbf{c}_1, ..., \mathbf{c}_n\}$, where $n = \left\lceil \frac{L_d - l}{r} \right\rceil + 1$, and L_d is the document length. Next, we encode these segments along with the question using pre-trained Transformer blocks (Devlin et al., 2018), which is a highly parallel encoding scheme instead of recurrent approaches such as LSTMs.

The input to the network is a sequence of tokens $\mathbf{x} = (x_1, ..., x_{L_x})$ with length L_x. It is obtained by concatenating the question, segment, and several delimiters as $[\texttt{[CLS]}; \mathbf{q}; \texttt{[SEP]}; \mathbf{c}; \texttt{[SEP]}]$, where `[CLS]` is a classification token and `[SEP]` is another token for differentiating sentences. We refer to this sequence as "segment" in the rest of this paper. For each token x_i in $\mathbf{x}$, its input representation is the element-wise addition of word, type, and position embeddings. Then, we can obtain the input embeddings $\mathbf{h}^0 \in \mathbb{R}^{L_x \times D_h}$, where D_h is hidden size.

Next, a series of I pre-trained Transformer blocks are used to project the input embeddings into a sequence of contextualized vectors as:

$$\mathbf{h}^i = \text{TransformerBlock}(\mathbf{h}^{i-1}), \forall i \in [1, I]$$

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

3.3 Early-Stopped Retriever

While we find the above parallel encoding scheme very appealing, there is a crucial computational inefficiency if all segments are fully encoded. For example, the average number of segments per instance in TriviaQA-unfiltered is 20 even after pruning, while the total number of Transformer blocks is 12 or 24. Therefore, we propose to rank all segments using early-summarized hidden representations as a mechanism for efficiently retrieving few top-ranked segments.

Specifically, let $\mathbf{h}^J$ denote the hidden states in the J-th block, where $J < I$. We compute a $\mathbf{score}^r \in \mathbb{R}^2$ by summarizing $\mathbf{h}^J$ into a fix-sized vector with a weighted self aligning layer followed by multi-layer perceptrons as:

$$\mu = \text{softmax}(\mathbf{w}_\mu \mathbf{h}^J)$$
$$\mathbf{score}^r = \mathbf{w}_r \tanh(\mathbf{W}_r \sum\nolimits_{i=1}^{L_x} \mu_i \mathbf{h}_i^J)$$

where $\mathbf{w}_\mu$, $\mathbf{w}_r$, $\mathbf{W}_r$ are parameters to be learned.

After obtaining the scores of all segments, we pass the top-N ranked segments per instance to the subsequent blocks, and discard the rest. Here, N is relatively small so that the model can focus on reading the most revelant context.

To train the retrieving component, we normalize $\mathbf{score}^r$ and define the objective function as:

$$\mathcal{L}_I = -\sum\nolimits_{i=1}^{2} \mathbf{y}_i^r \log(\text{softmax}(\mathbf{score}^r)_i) \quad (1)$$

where $\mathbf{y}^r$ is an one-hot label indicating whether current segment contains at least one exactly-matched ground truth answer text or not.

3.4 Distantly-Supervised Reader

Given the retrieved segments, the reading component aims to propose multiple candidate answers per segment. This is achieved by first element-wisely projecting the final hidden states $\mathbf{h}^I$ into two sets of scores as follows:

$$\mathbf{score}^s = \mathbf{w}_s \mathbf{h}^I \ , \ \mathbf{score}^e = \mathbf{w}_e \mathbf{h}^I$$

where $\mathbf{score}^s \in \mathbb{R}^{L_x}$ and $\mathbf{score}^e \in \mathbb{R}^{L_x}$ are the scores for the start and end positions of answer spans, and $\mathbf{w}_s$, $\mathbf{w}_e$ are trainable parameter vectors.

Next, let α_i and β_i denote the start and end indices of candidate answer $\mathbf{a}_i$. We compute a reading score, $\mathbf{s}_i = \mathbf{score}^s_{\alpha_i} + \mathbf{score}^e_{\beta_i}$, and then propose top-$M$ candidates according to the descending order of the scores, yielding a set of preliminary candidate answers $\mathbf{A} = \{\mathbf{a}_1, ..., \mathbf{a}_M\}$ along with their scores $\mathbf{S} = \{\mathbf{s}_1, ..., \mathbf{s}_M\}$.

Following previous work (Clark and Gardner, 2018), we label all text spans within a segment that match the gold answer as being correct, thus yielding two label vectors $\mathbf{y}^s \in \mathbb{R}^{L_x}$ and $\mathbf{y}^e \in \mathbb{R}^{L_x}$. Since there is a chance that the segment does not contain any answer string, we then label the first element in both $\mathbf{y}^s$ and $\mathbf{y}^e$ as 1, and set the rest as 0. Finally, we define the objective function as:

$$\mathcal{L}_{II} = -\sum\nolimits_{i=1}^{L_x} \mathbf{y}_i^s \log(\text{softmax}(\mathbf{score}^s)_i) - \sum\nolimits_{j=1}^{L_x} \mathbf{y}_j^e \log(\text{softmax}(\mathbf{score}^e)_j) \quad (2)$$

3.5 Answer Reranker

The answer reranker aims to rerank the candidate answers proposed by the previous reader. We first introduce a span-level non-maximum suppression algorithm to prune redundant candidate spans, and then predict the reranking scores for remaining candidates using their span representations.

Span-level non-maximum suppression So far, the reader has proposed multiple candidate spans. However, since there is no constraint to predict an unique span for an answer string, multiple candidates may refer to the same text. As a result, other than the first correct span, all other spans on the same text would be false positives. Figure 2 shows a qualitative example of this phenomenon.

> **Question:** In the late 60s Owen Finlay MacLaren pioneered what useful item for parents of small chldren?
> **Answer:** baby buggy
> **Candidates:** baby buggy, collapsible baby buggy, buggy, folding buggy, folding chair ...

Figure 2: An example from TriviaQA shows that multiple candidate answers refer to the same text.

Inspired by the non-maximum suppression (NMS) algorithm (Rosenfeld and Thurston, 1971) that is used to prune redundant bounding boxes in object detection (Ren et al., 2015), we present a span-level NMS (Algorithm 1) to alleviate the problem. Specifically, span-level NMS starts with a set of candidate answers $\mathbf{A}$ with scores $\mathbf{S}$. After selecting the answer $\mathbf{a}_i$ that possesses the maximum score, we remove it from the set $\mathbf{A}$ and add it to $\mathbf{B}$. We also delete any answer $\mathbf{a}_j$ in $\mathbf{A}$ that is overlapped with $\mathbf{a}_i$. We define that two candidates overlap with each other if they share at least one boundary position[4]. This process is repeated for remaining answers in $\mathbf{A}$, until $\mathbf{A}$ is empty or the size of $\mathbf{B}$ reaches a maximum threshold.

Candidate answer reranking Given the candidate answer $\mathbf{a}_i$ in $\mathbf{B}$, we compute a reranking score based on its span representation, where the representation is a weighted self-aligned vector bounded by the span boundary of the answer, similar to Lee et al. (2017); He et al. (2018):

$$\eta = \mathrm{softmax}(\mathbf{w}_\eta \mathbf{h}^I_{\alpha_i:\beta_i})$$

$$\mathbf{score}^a_i = \mathbf{w}_a \tanh(\mathbf{W}_a \sum\nolimits_{j=\alpha_i}^{\beta_i} \eta_{j-\alpha_i+1} \mathbf{h}^I_j)$$

[4]We also experimented with the span-level F1 function, but found no performance improment.

Algorithm 1 Span-level NMS

Input: $\mathbf{A} = \{\mathbf{a}_i\}_{i=1}^M$; $\mathbf{S} = \{\mathbf{s}_i\}_{i=1}^M$; M^*
$\mathbf{A}$ is the set of preliminary candidate answers
$\mathbf{S}$ is the corresponding confidence scores
M^* denotes the maximum size threshold

```
Initialize B = {}
while A ≠ {} and size(B) < M* do
    i = arg max S
    B = B ∪ {a_i}; A = A − {a_i}; S = S − {s_i}
    for a_j in A do
        if overlap(a_i, a_j) then
            A = A − {a_j}; S = S − {s_j}
return B
```

Here, $\mathbf{score}^a \in \mathbb{R}^{M^*}$, and $\mathbf{h}^I_{\alpha_i:\beta_i}$ is a shorthand for stacking a list of vectors $\mathbf{h}^I_j$ ($\alpha_i \le j \le \beta_i$).

To train the reranker, we construct two kinds of labels for each candidate $\mathbf{a}_i$. First, we define a hard label $\mathbf{y}^{hard}_i$ as the maximum exact match score between $\mathbf{a}_i$ and ground truth answers. Second, we also utilize a soft label $\mathbf{y}^{soft}_i$, which is computed as the maximum F1 score between $\mathbf{a}_i$ and gold answers, so that the partially correct prediction can still have a supervised signal. The above labels are annotated for each candidate in $\mathbf{B}$, yielding $\mathbf{y}^{hard} \in \mathbb{R}^{M^*}$ and $\mathbf{y}^{soft} \in \mathbb{R}^{M^*}$. If there is no correct prediction in $\mathbf{B}$ (all elements of $\mathbf{y}^{hard}$ are 0), then we replace the least confident candidate with a gold answer. Finally, we define the following reranking objective:

$$\mathcal{L}_{III} = -\sum\nolimits_{i=1}^{M^*} \mathbf{y}^{hard}_i \log(\mathrm{softmax}(\mathbf{score}^a)_i) + \sum\nolimits_{i=1}^{M^*} ||\mathbf{y}^{soft}_i - \frac{\mathbf{score}^a_i}{\sum_{j=1}^{M^*} \mathbf{score}^a_j}||^2 \quad (3)$$

3.6 Training and Inference

Rather than separately training each component, we propose an end-to-end training strategy so that downstream components (e.g., the reader) can benefit from the high-quality upstream outputs (e.g., the retrieved segments) during training.

Specifically, we take a multi-task learning approach (Caruna, 1993; Ruder, 2017), sharing the parameters of earlier blocks with a joint objective function defined as:

$$\mathcal{J} = \mathcal{L}_I + \mathcal{L}_{II} + \mathcal{L}_{III}$$

Algorithm 2 details the training process. Before each epoch, we compute $\mathbf{score}^r$ for all segments in the training set $\mathcal{X}$. Then, we retrieve top-N segments per instance and construct a new training set $\tilde{\mathcal{X}}$, which only contains retrieved segments. For

Dataset	#Ins	#Doc	#Seg	#Tok	#Tok*	K	N	Recall
TriviaQA-Wikipedia	7,993	1.8	17	10,256	2,103	14	8	94.8%
TriviaQA-unfiltered	11,313	11.7	20	52,635	2,542	14	8	84.3%
SQuAD-document	10,570	1	35	5,287	3,666	30	8	99.0%
SQuAD-open	10,570	5	42	38,159	5,103	30	8	64.9%

Table 2: Dataset statistics. '#Ins' denotes the number of instances, while '#Doc', '#Seg', '#Tok', and '#Tok*' refer to the average number of documents, segments, and tokens before/after pruning, respectively. K and N are the number of retrieved paragraphs and segments. All statistics are calculated on the development set.

each instance, if all of its top-ranked segments are negative examples, then we replace the least confident one with a gold segment. During each epoch, we sample two sets of mini-batch from both the $\mathcal{X}$ and the $\tilde{\mathcal{X}}$, where the first batch is used to calculate $\mathcal{L}_I$ and the other one for computing $\mathcal{L}_{II}$ and $\mathcal{L}_{III}$. Note that the contextualized vectors $\mathbf{h}^I$ are shared across the reader and the reranker to avoid repeated computations. The batch size of $\mathcal{X}$ is dynamically decided so that both of $\mathcal{X}$ and $\tilde{\mathcal{X}}$ can be traversed with the same number of steps.

During inference, we take the retrieving, reading, and reranking scores into account. We compare the scores across all segments from the same instance, and choose the final answer according to the weighted addition of these three scores.

Algorithm 2 End-to-end training of RE^{3}QA

Input: $\mathcal{X} = \{\mathbf{X}_i\}_{i=1}^{t}$, where $\mathbf{X}_i = \{\mathbf{x}_i^j\}_{j=1}^{n}$; $\mathbf{M}_\Theta$; k
$\mathcal{X}$ is the dataset containing t instances
$\mathbf{X}^i$ is i-th instance containing n segments
$\mathbf{M}_\Theta$ denotes the model with parameters Θ
k is the maximum number of epoch

Initialize Θ from pre-trained parameters
for epoch in $1, ..., k$ **do**
  Compute $\mathbf{score}^r$ for all $\mathbf{x}$ in $\mathcal{X}$
  Retrieve top-N segments per instance
  Construct a new $\tilde{\mathcal{X}}$ that includes retrieved $\mathbf{x}$
  for $\texttt{batch}_{\mathcal{X}}$, $\texttt{batch}_{\tilde{\mathcal{X}}}$ in $\mathcal{X}$, $\tilde{\mathcal{X}}$ **do**
    Compute $\mathcal{L}_I$ using $\texttt{batch}_{\mathcal{X}}$ by Eq. 1
    Compute $\mathcal{L}_{II}$ using $\texttt{batch}_{\tilde{\mathcal{X}}}$ by Eq. 2
    Reuse $\mathbf{h}^I$ to compute $\mathcal{L}_{III}$ by Eq. 3
    Update $\mathbf{M}_\Theta$ with graident $\nabla(\mathcal{J})$

4 Experimental Setup

Datasets We experiment on four datasets: (a) TriviaQA-Wikipedia (Joshi et al., 2017), a dataset of 77K trivia questions where each question is paired with one or multiple Wikipedia articles. (b) TriviaQA-unfiltered is a open-domain dataset that contains 99K question-answer tuples. The evidence documents are constructed by completing a web search given the question. (c) SQuAD-document, a variant of SQuAD dataset (Rajpurkar et al., 2016) that pairs each question with a full Wikipedia article instead of a specific paragraph. (d) SQuAD-open (Chen et al., 2017) is the open domain version of SQuAD where the evidence corpus is the entire Wikipedia domain. For fair comparision to other methods, we retrieve top-5 articles as our input documents. The detailed statistics of these datasets are shown in Table 2.

Data preprocessing Following Clark and Gardner (2018), we merge small paragraphs into a single paragraph of up to a threshold length in TriviaQA and SQuAD-open. The threshold is set as 200 by default. We manually tune the number of retrieved paragraphs K for each dataset, and set the number of retrieved segments N as 8. Following Devlin et al. (2018), we set the window length l as $384 - L_q - 3$ so that L_x is 384 and set the stride r as 128, where L_q is the question length. We also calculate the answer recall after document pruning, which indicates the performance upper bound.

Model settings We initialize our model using two publicly available uncased versions of BERT[5]: $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$, and refer readers to Devlin et al. (2018) for details on model sizes. We use Adam optimizer with a learning rate of 3e-5 and warmup over the first 10% steps to fine-tune the network for 2 epochs. The batch size is 32 and a dropout probability of 0.1 is used. The number of blocks J used for early-stopped retriever is 3 for base model and 6 for large model by default. The number of proposed answers M is 20, while the threshold of NMS M^* is 5. During inference, we tune the weights for retrieving, reading, and reranking, and set them as 1.4, 1, 1.4.

Evaluation metrics We use mean average precision (MAP) and top-N to evaluate the retriev-

[5]https://github.com/google-research/bert

Model	Full		Verified	
	EM	F1	EM	F1
Baseline[1]	40.3	45.9	44.9	50.7
M-Reader[2]	46.9	52.9	54.5	59.5
Re-Ranker[3]	50.2	55.5	58.7	63.2
DrQA[4]	52.6	58.2	57.4	62.6
S-Norm[5]	64.0	68.9	68.0	72.9
MemoReader[6]	64.4	69.6	70.2	75.5
Reading Twice[7]	64.6	69.9	72.8	77.4
SLQA[8]	66.6	71.4	74.8	78.7
CAPE†	67.3	72.4	75.7	79.3
RE^{3}QA$_{\text{BASE}}$	68.4	72.6	76.7	79.9
RE^{3}QA$_{\text{LARGE}}$	**71.0**	**75.2**	**80.3**	**83.0**

Table 3: Results on the TriviaQA-Wikipedia test set: Joshi et al. (2017)[1], Hu et al. (2018)[2], Wang et al. (2018b)[3], Chen et al. (2017)[4], Clark and Gardner (2018)[5], Back et al. (2018)[6], Weissenborn et al. (2017)[7], and Yan et al. (2019)[8]. † indicates unpublished works.

Model	EM	F1
S-Norm (Clark and Gardner, 2018)	64.08	72.37
RE^{3}QA$_{\text{BASE}}$	77.90	84.81
RE^{3}QA$_{\text{LARGE}}$	**80.71**	**87.20**

Table 4: Results on the SQuAD-document dev set.

Model	TriviaQA-unfiltered		SQuAD-open	
	EM	F1	EM	F1
DrQA[1]	32.3	38.3	27.1	-
R3[2]	47.3	53.7	29.1	37.5
DS-QA[3]	48.7	56.3	28.7	36.6
Re-Ranker[4]	50.6	57.3	-	-
MINIMAL[5]	-	-	34.7	42.5
Multi-Step[6]	51.9	61.7	31.9	39.2
S-Norm[7]	61.3	67.2	-	-
HAS-QA[8]	63.6	68.9	-	-
BERTserini[9]	-	-	38.6	46.1
RE^{3}QA$_{\text{BASE}}$	64.1	69.8	40.1	48.4
RE^{3}QA$_{\text{LARGE}}$	**65.5**	**71.2**	**41.9**	**50.2**

Table 5: Results on TriviaQA-unfiltered test set and SQuAD-open dev set: Chen et al. (2017)[1], Wang et al. (2018a)[2], Lin et al. (2018)[3], Wang et al. (2018b)[4], Min et al. (2018)[5], Das et al. (2019)[6], Clark and Gardner (2018)[7], Pang et al. (2019)[8] and Yang et al. (2019)[9].

Model	TriviaQA-Wikipedia		SQuAD-document	
	F1	Speed	F1	Speed
RE^{3}QA	72.68	4.62	84.81	3.76
BERT$_{\text{PIPE}}$	71.13	2.05	83.65	1.78
BERT$_{\text{PIPE}}$*	71.59	2.08	84.04	1.82

Table 6: Comparison between our approach and the pipelined method. "Speed" denotes the number of instances processed per second during inference.

ing component. As for evaluating the performance of reading and reranking, we measure the exact match (EM) accuracy and F1 score calculated between the final prediction and gold answers.

Baselines We construct two pipelined baselines (denoted as BERT$_{\text{PIPE}}$ and BERT$_{\text{PIPE}}$*) to investigate the context inconsistency problem. Both systems contain exactly the same components (e.g., retriever, reader, and reranker) as ours, except that they are trained separately. For BERT$_{\text{PIPE}}$, the reader is trained on the context retrieved by an IR engine. As for BERT$_{\text{PIPE}}$*, the reading context is obtained using the trained neural retriever.

5 Evaluation

5.1 Main Results

Table 3 summarizes the results on the test set of TriviaQA-Wikipedia dataset. As we can see, our best model achieves 71.0 EM and 75.2 F1, firmly outperforming previous methods. Besides, Joshi et al. (2017) show that the evidence documents contain answers for only 79.7% of questions in the Wikipedia domain, suggesting that we are approaching the ceiling performance of this task. However, the score of 80.3 EM on the verified set implies that there is still room for improvement.

We also report the performance on document-level SQuAD in Table 4 to assess our approach in single-document setting. We find our approach adapts well: the best model achieves 87.2 F1. Note that the BERT$_{\text{LARGE}}$ model has obtained 90.9 F1 on the original SQuAD dataset (single-paragraph setting), which is only 3.7% ahead of us.

Finally, to validate our approach in open-domain scenarios, we run experiments on the TriviaQA-unfiltered and SQuAD-open datasets, as shown in Table 5. Again, RE^{3}QA surpasses prior works by an evident margin: our best model achieves 71.2 F1 on TriviaQA-unfiltered, and outperforms a BERT baseline by 4 F1 on SQuAD-open, indicating that our approach is effective for the challenging multi-document RC task.

5.2 Model Analysis

In this section, we analyze our approach by answering the following questions[6]: (a) Is end-to-

[6]The BERT$_{\text{BASE}}$ model is used by default in this section.

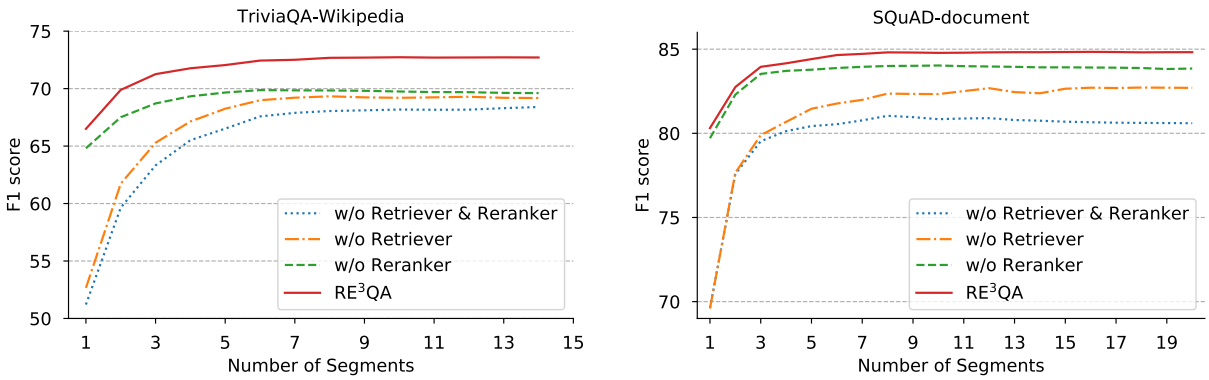

Figure 3: F1 score on TriviaQA-Wikipedia and SQuAD-document w.r.t different number of retrieved segments.

J	TriviaQA-Wikipedia					SQuAD-document				
	MAP	Top-3	Top-5	F1	Speed	MAP	Top-3	Top-5	F1	Speed
1	67.4	81.5	87.3	69.2	5.9	39.2	47.5	66.8	54.4	5.6
2	75.3	87.4	91.1	71.7	5.1	80.3	89.4	94.0	83.4	4.7
3	77.8	88.8	91.8	72.7	4.6	88.7	94.5	96.8	84.8	3.8
4	80.0	89.2	92.1	71.6	4.2	90.2	95.0	97.2	84.3	3.0
5	80.6	89.6	92.3	71.7	3.5	91.0	95.6	97.6	84.3	2.3

Table 7: Retrieving performance with different number of blocks J used for the early-stopped retriever.

end network superior to the pipeline system? (b) How does each component contribute to the performance? (c) Is early-stopped retriever sufficient for returning high-quality segments? (d) How does the reranking loss affect the answer reranker?

Comparison with pipelined method First, we compare our approach with the pipelined baselines on TriviaQA-Wikipedia and SQuAD-document development sets in Table 6. Our approach outperforms $\text{BERT}_{\text{PIPE}}$ by 1.6/1.2 F1 on two datasets respectively, and is also 2.3/2.1 times faster during inference. Moreover, RE^3QA also beats the $\text{BERT}_{\text{PIPE}}$* baseline by 1.1/0.8 F1, even as the parameters of retriever and reader are trained sequentially in $\text{BERT}_{\text{PIPE}}$*. The above results confirm that the end-to-end training can indeed mitigate the context inconsistency problem, perhaps due to multi-task learning and parameter sharing. Our approach can also obtain inference speedups because of the fact that it avoids re-encoding inputs by sharing contextualized representations.

Ablation study To show the effect of each individual component, we plot the F1 curve with respect to different number of retrieved segments in Figure 3. We notice that all curves become stable as more text are used, implying that our approach is robust across different amounts of context. Next, to evaluate the reranker, we only consider the retrieving and reading scores, and the performance decreases by 2.8/0.8 F1 on two datasets after the reranker is removed. To ablate the retriever, we select segments based on the TF-IDF distance instead. The results show that the F1 score reduces by about 3.3 and 2.5 points on two datasets after the ablation. Removing both the retriever and the reranker performs the worst, which only achieves 68.1/81.0 F1 on two datasets at peak. The above results suggest that combining retriever, reader, and reranker is crucial for achieving promising performance.

Effect of early-stopped retriever We assess whether the early-stopped retriever is sufficient for the segment retrieving task. Table 7 details the retrieving and reading results with different number of blocks J being used. As we can see, the model performs worst but maintains a high speed when J is 1. As J becomes larger, the retrieving metrices such as MAP, Top-3 and Top-5 significantly increase on both datasets. On the other hand, the speed continues to decline since more computations have been done during retrieving. A J of 6 eventually leads to an out-of-memory issue on both datasets. As for the F1 score, the model

Model	TriviaQA-Wikipedia EM	TriviaQA-Wikipedia F1	SQuAD-document EM	SQuAD-document F1
RE^{3}QA	68.51	72.68	77.90	84.81
w/o NMS	68.29	72.33	77.67	84.36
w/o $\mathbf{y}^{hard}$	67.36	71.87	77.26	84.17
w/o $\mathbf{y}^{soft}$	67.76	72.29	77.04	84.05

Table 8: Reranking performance with different ablations. $\mathbf{y}^{hard}$ and $\mathbf{y}^{soft}$ refer to the two labels used to train the reranker.

achieves the best result when J reaches 3, and starts to degrade as J continues rising. We experiment with the RE^{3}QA$_{\text{LARGE}}$ model and observe similar results, where the best J is 6. A likely reason for this observation may be that sharing high-level features with the retriever could disturb the reading prediction. Therefore, the above results demonstrate that an early-stopped retriever with a relatively small J is able to reach a good trade-off between efficiency and effectiveness.

Effect of answer reranker Finally, we run our model under different reranking ablations and report the results in Table 8. As we can see, removing the non-maximum suppression (NMS) algorithm has a negative impact on the performance, suggesting it is necessary to prune highly-overlapped candidate answers before reranking. Ablating the hard label leads to a drop of 0.81 and 0.64 F1 scores on two datasets respectively, while the F1 drops by 0.39 and 0.76 points after removing the soft label. This implies that the hard label has a larger impact than the soft label on the TriviaQA dataset, but vice versa on SQuAD.

6 Conclusion

We present RE^{3}QA, a unified network that answers questions from multiple documents by conducting the retrieve-read-rerank process. We design three components for each subtask and show that an end-to-end training strategy can bring in additional benefits. RE^{3}QA outperforms the pipelined baseline with faster inference speed and achieves state-of-the-art results on four challenging reading comprehension datasets. Future work will concentrate on designing a fast neural pruner to replace the IR-based pruning component, developing better end-to-end training strategies, and adapting our approach to other datasets such as Natural Questions (Kwiatkowski et al., 2019).

Acknowledgments

We would like to thank Mandar Joshi for his help with TriviaQA submissions. We also thank anonymous reviewers for their thoughtful comments and helpful suggestions. This work was supported by the National Key Research and Development Program of China (2018YFB0204300).

References

Seohyun Back, Seunghak Yu, Sathish Reddy Indurthi, Jihie Kim, and Jaegul Choo. 2018. Memoreader: Large-scale reading comprehension through neural memory controller. In *Proceedings of EMNLP*.

Dasha Bogdanova and Jennifer Foster. 2016. This is how we do it: Answer reranking for open-domain how questions with paragraph vectors and minimal feature engineering. In *Proceedings of NAACL*.

Rich Caruna. 1993. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of ICML*.

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

Eunsol Choi, Daniel Hewlett, Jakob Uszkoreit, Illia Polosukhin, Alexandre Lacoste, and Jonathan Berant. 2017. Coarse-to-fine question answering for long documents. In *Proceedings of ACL*.

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

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. 2019. Multi-step retriever-reader interaction for scalable open-domain question answering. In *Proceedings of ICLR*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Bhuwan Dhingra, Kathryn Mazaitis, and William W Cohen. 2017. Quasar: Datasets for question answering by search and reading. *arXiv preprint arXiv:1707.03904*.

Matthew Dunn, Levent Sagun, Mike Higgins, V Ugur Guney, Volkan Cirik, and Kyunghyun Cho. 2017. Searchqa: A new q&a dataset augmented with context from a search engine. *arXiv preprint arXiv:1704.05179*.

Luheng He, Kenton Lee, Omer Levy, and Luke Zettlemoyer. 2018. Jointly predicting predicates and arguments in neural semantic role labeling. *arXiv preprint arXiv:1805.04787*.

Daniel Hewlett, Llion Jones, Alexandre Lacoste, et al. 2017. Accurate supervised and semi-supervised machine reading for long documents. In *Proceedings of EMNLP*.

Phu Mon Htut, Samuel R Bowman, and Kyunghyun Cho. 2018. Training a ranking function for open-domain question answering. In *Proceedings of NAACL*.

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*.

Minghao Hu, Yuxing Peng, Zhen Huang, Nan Yang, Ming Zhou, et al. 2019. Read+ verify: Machine reading comprehension with unanswerable questions. In *Proceedings of AAAI*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of EMNLP*.

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*.

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

Kenton Lee, Luheng He, Mike Lewis, and Luke Zettlemoyer. 2017. End-to-end neural coreference resolution. *arXiv preprint arXiv:1707.07045*.

Yankai Lin, Haozhe Ji, Zhiyuan Liu, and Maosong Sun. 2018. Denoising distantly supervised open-domain question answering. In *Proceedings of ACL*.

Sewon Min, Victor Zhong, Richard Socher, and Caiming Xiong. 2018. Efficient and robust question answering from minimal context over documents. In *Proceedings of ACL*.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Lixin Su, and Xueqi Cheng. 2019. Has-qa: Hierarchical answer spans model for open-domain question answering. In *Proceedings of AAAI*.

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

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proceedings of NIPS*.

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

Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Richard S Sutton and Andrew G Barto. 2011. Reinforcement learning: An introduction.

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, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. 2018a. R3: Reinforced ranker-reader for open-domain question answering. In *Proceedings of AAAI*.

Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. 2018b. 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. 2018c. 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. 2018d. Joint training of candidate extraction and answer selection for reading comprehension. In *Proceedings of ACL*.

Dirk Weissenborn, Tomáš Kočiskỳ, and Chris Dyer. 2017. Dynamic integration of background knowledge in neural nlu systems. *arXiv preprint arXiv:1706.02596*.

Ming Yan, Jiangnan Xia, Chen Wu, Bin Bi, Zhongzhou Zhao, Ji Zhang, Luo Si, Rui Wang, Wei Wang, and Haiqing Chen. 2019. A deep cascade model for multi-document reading comprehension. In *Proceedings of AAAI*.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. End-to-end open-domain question answering with bertserini. *arXiv preprint arXiv:1902.01718*.

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 *Proceedings of ICLR*.

Question: Which organisation was founded in Ontario, Canada in 1897 by Adelaide Hoodless?	**Scores**		
Candidate Answers:	**Retrieving**	**Reading**	**Reranking**
[1] Women's Institute	**0.517**	11.226	2.093
[2] Young Women's Christian Association	0.231	11.263	**2.299**
[3] Federated Women's Institutes of Canada	0.426	**11.267**	1.742
[4] Victorian Order of Nurses	0.360	11.139	1.837
[5] National Council of Women	0.291	8.966	1.02
.....		.....	

Table 9: A sampled case (ID: sfq_21220) from the TriviaQA-Wikipedia dev set shows that although candidate [2] and candidate [3] get higher reranking and reading scores, the candidate [1] is preferred by the retrieving component and is therefore chosen as the final answer. The ground truth answer is "*Women's Institute*".

Question: Hong Kong is one of two 'special administrative regions' of China; what is the other?	**Scores**		
Candidate Answers:	**Retrieving**	**Reading**	**Reranking**
[1] Macau	0.195	11.067	**2.502**
[2] Kowloon	**0.346**	**11.175**	1.795
[3] Kowloon, and the new territories	0.346	7.941	0
[4] Macau, China	0.323	7.812	0
[5] Taiwan	0.224	5.926	0.028
.....		.....	

Table 10: A sampled case (ID: sfq_10640) from the TriviaQA-Wikipedia dev set shows that although the candidate [2] gets higher retrieving and reading scores, the candidate [1] is chosen as the final answer since it has the highest reranking score. The ground truth answer is "*Macau*".

A Case Study

To demonstrate how each component takes effect when predicting the final answer, we conduct some qualitative case studies sampled from the RE^{3}QA$_{\text{LARGE}}$ model on the TriviaQA-Wikipedia development set. For each question, we list top-5 candidate answers along with their retrieving, reading, and reranking scores.

As shown in Table 9, we first notice that the top-ranked predictions have highly-relevant semantics and share the same linguistic pattern. As a result, the top-4 candidates contain very similar reading scores from 11.1 to 11.3, which matches the observations of Clark and Gardner (2018). A likely reason of this phenomenon is that reading comprehension models are easily fooled by confusing distractors (also referred as adversarial examples mentioned by Jia and Liang (2017)). Under such circumstance, it is crucial to perform additional answer verifications to identify the final answer. In this example, we can see that the retriever becomes the key factor when the reader and reranker are distracted by confusing candidates (e.g., the second and third predictions). By taking the weighted sum of the three scores, our model eventually predicts the correct answer since the first prediction has the largest retrieving score.

Similar observations can be made in Table 10. On the one hand, despite the confusing candidate "*Kowloon*" has the highest retrieving and reading scores, the reranker assigns a larger confidence on the candidate "*Macau*". As a result, "*Macau*" is chosen as the final answer. On the other hand, we find that the reranking scores of some candidates (e.g., the third and fourth predictions) are zero. This is due to the span-level non-maximum suppression algorithm, where redundant spans such as "*Macau, China*" will be pruned before the reranking step. Therefore, the final weighted-sum scores of these candidates will be significantly lower than the top predictions, which is beneficial for filtering distractors out.