Read + Verify: Machine Reading Comprehension with Unanswerable Questions

Minghao Hu,[*1] **Furu Wei,**[2] **Yuxing Peng,**[1] **Zhen Huang,**[1] **Nan Yang,**[2] **Dongsheng Li**[1]
[1]College of Computer, National University of Defense Technology
[2]Microsoft Research Asia
{huminghao09,pengyuxing,huangzhen,dsli}@nudt.edu.cn
{fuwei,nanya}@microsoft.com

Abstract

Machine reading comprehension with unanswerable questions aims to abstain from answering when no answer can be inferred. In addition to extract answers, previous works usually predict an additional "no-answer" probability to detect unanswerable cases. However, they fail to validate the answerability of the question by verifying the legitimacy of the predicted answer. To address this problem, we propose a novel read-then-verify system, which not only utilizes a neural reader to extract candidate answers and produce no-answer probabilities, but also leverages an answer verifier to decide whether the predicted answer is entailed by the input snippets. Moreover, we introduce two auxiliary losses to help the reader better handle answer extraction as well as no-answer detection, and investigate three different architectures for the answer verifier. Our experiments on the SQuAD 2.0 dataset show that our system obtains a score of 74.2 F1 on test set, achieving state-of-the-art results at the time of submission (Aug. 28th, 2018).

Introduction

The ability to comprehend text and answer questions is crucial for natural language processing. Due to the creation of various large-scale datasets (Hermann et al. 2015; Nguyen et al. 2016; Joshi et al. 2017; Kočiskỳ et al. 2018), remarkable advancements have been made in the task of machine reading comprehension. Nevertheless, one important hypothesis behind current approaches is that there always exists a correct answer in the context passage. Therefore, the models only need to choose a most plausible text span based on the question, instead of checking if there exists an answer in the first place. Recently, a new version of Stanford Question Answering Dataset (SQuAD), namely SQuAD 2.0 (Rajpurkar, Jia, and Liang 2018), has been proposed to test the ability of answering answerable questions as well as detecting unanswerable cases. To deal with unanswerable cases, systems must learn to identify a wide range of linguistic phenomena such as negation, antonymy and entity changes between the passage and the question.

Previous works (Levy et al. 2017; Clark and Gardner 2018; Kundu and Ng 2018) all apply a shared-normalization operation between a "no-answer" score and answer span scores, so as to produce a probability that a question is unanswerable as well as output a candidate answer. However, they have not considered further validating the answerability of the question by verifying the legitimacy of the predicted answer. Here, *answerability* denotes whether the question has an answer, and *legitimacy* means whether the extracted text can be supported by the passage and the question. Human, on the contrary, tends to first find a plausible answer given a question, and then checks if there exists any contradictory semantics.

To address the above issue, we propose a *read-then-verify* system that aims to be robust to unanswerable questions in this paper. As shown in Figure 1, our system consists of two

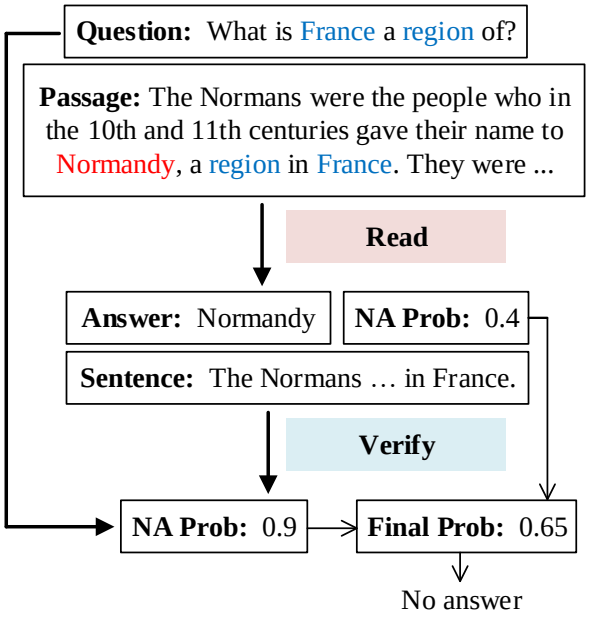

Figure 1: An overview of our approach. The reader first extracts a candidate answer and produces a no-answer probability (NA Prob). The answer verifier then checks whether the extracted answer is legitimate or not. Finally, the system aggregates previous results and outputs the final prediction.

[*]Contribution during internship at Microsoft Research Asia.

components: (1) a *no-answer reader* for extracting candidate answers and detecting unanswerable questions, and (2) an *answer verifier* for deciding whether or not the extracted candidate is legitimate. The key contributions of our work are three-fold.

First, we augment existing readers with two *auxiliary losses*, to better handle answer extraction and no-answer detection respectively. Since the downstream verifying stage always requires a candidate answer, the reader must be able to extract plausible answers for all questions. However, previous approaches are not trained to find potential candidates for unanswerable questions. We solve this problem by introducing an independent span loss that aims to concentrate on the answer extraction task regardless of the answerability of the question. In order to not conflict with no-answer detection, we leverage a *multi-head pointer network* to generate two pairs of span scores, where one pair is normalized with the no-answer score and the other is used for our auxiliary loss. Besides, we present another independent no-answer loss to further alleviate the confliction, by focusing on the no-answer detection task without considering the shared normalization of answer extraction.

Second, in addition to the standard reading phase, we introduce an additional answer verifying phase, which aims at finding local entailment that supports the answer by comparing the answer sentence with the question. This is based on the observation that the core phenomenon of unanswerable questions usually occurs between a few passage words and question words. Take Figure 1 for example, after comparing the passage snippet "*Normandy, a region in France*" with the question, we can easily determine that no answer exists since the question asks for an *impossible condition*[1]. This observation is even more obvious when *antonym* or *mutual exclusion* occurs, such as the question asks for "*the decline of rainforests*" but the passage mentions that "*the rainforests spread out*". Inspired by recent advances in natural language inference (NLI) (Bowman et al. 2015), we investigate three different architectures for the answer verifying task. The first one is a sequential model that takes two sentences as a long sequence, while the second one attempts to capture interactions between two sentences. The last one is a hybrid model that combines the above two models to test if the performance can be further improved.

Lastly, we evaluate our system on the SQuAD 2.0 dataset (Rajpurkar, Jia, and Liang 2018), a reading comprehension benchmark augmented with unanswerable questions. Our best reader achieves a F1 score of 73.7 and 69.1 on the development set, with or without ELMo embeddings (Peters et al. 2018). When combined with the answer verifier, the whole system improves to 74.8 F1 and 71.5 F1 respectively. Moreover, the best system obtains a score of 74.2 F1 on test set, achieving state-of-the-art results at the time of submission (Aug. 28th, 2018).

[1]Impossible condition means that the question asks for something that is not satisfied by anything in the given passage.

Background

Existing reading comprehension models focus on answering questions where a correct answer is guaranteed to exist. However, they are not able to identify unanswerable questions but tend to return an unreliable text span. Consequently, we first give a brief introduction on the unanswerable reading comprehension task, and then investigate current solutions.

Task Description

Given a context passage and a question, the machine needs to not only find answers to answerable questions but also detect unanswerable cases. The passage and the question are described as sequences of word tokens, denoted as $P = \{x_i^p\}_{i=1}^{l_p}$ and $Q = \{x_j^q\}_{j=1}^{l_q}$ respectively, where l_p is the passage length and l_q is the question length. Our goal is to predict an answer A, which is constrained as a segment of text in the passage: $A = \{x_i^p\}_{i=l_a}^{l_b}$, or return an empty string if there is no answer, where l_a and l_b indicate the answer boundary.

No-Answer Reader

To predict an answer span, current approaches first embed and encode both of passage and question into two series of fix-sized vectors. Then they leverage various attention mechanisms, such as bi-attention (Seo et al. 2017) or reattention (Hu et al. 2018a), to build interdependent representations for passage and question, which are denoted as $U = \{u_i\}_{i=1}^{l_p}$ and $V = \{v_j\}_{j=1}^{l_q}$ respectively. Finally, they summarize the question representation into a dense vector t, and utilize the pointer network (Vinyals, Fortunato, and Jaitly 2015) to produce two scores over passage words that indicate the answer boundary (Wang et al. 2017):

$$o_j = w_v^{\mathrm{T}} v_j \ , \ t = \sum_{j=1}^{l_q} \frac{e^{o_j}}{\sum_{k=1}^{l_q} e^{o_k}} v_j$$

$$\alpha, \beta = \mathrm{pointer\_network}(U, t)$$

where α and β are the *span scores* for answer start and end bounds.

In order to additionally detect if the question is unanswerable, previous approaches (Levy et al. 2017; Clark and Gardner 2018; Kundu and Ng 2018) attempt to predict a special *no-answer score* z in addition to the distribution over answer spans. Concretely, a shared softmax function can be applied to normalize both of no-answer score and span scores, yielding a joint no-answer objective defined as:

$$\mathcal{L}_{joint} = -\log \left(\frac{(1-\delta)e^z + \delta e^{\alpha_a \beta_b}}{e^z + \sum_{i=1}^{l_p} \sum_{j=1}^{l_p} e^{\alpha_i \beta_j}} \right)$$

where a and b are the ground-truth start and end positions, and δ is 1 if the question is answerable and 0 otherwise. At test time, a question is detected as being unanswerable once the normalized no-answer score exceeds some threshold.

Approach

In this section we describe our proposed read-then-verify system. The system first leverages a neural reader to extract a candidate answer and detect if the question is unanswerable. It then utilizes an answer verifier to further check the legitimacy of the predicted answer. We enhance the reader with two novel auxiliary losses, and investigate three different architectures for the answer verifier.

Reader with Auxiliary Losses

Although previous no-answer readers are capable of jointly learning answer extraction and no-answer detection, there exists two problems for each individual task. For the answer extraction, previous readers are not trained to find candidate answers for unanswerable questions. In our system, however, the reader is required to extract a plausible answer that is fed to the downstream verifying stage for all questions. As for no-answer detection, a *confliction* could be triggered due to the shared normalization between span scores and no-answer score. Since the sum of these normalized scores is always 1, an over-confident span probability would cause an unconfident no-answer probability, and vice versa. Therefore, inaccurate confidence on answer span, which has been observed by Clark et al. (2018), could lead to imprecise prediction on no-answer score. To address the above issues, we propose two auxiliary losses to optimize and enhance each task independently without interfering with each other.

Independent Span Loss This loss is designed to concentrate on answer extraction. In this task, the model is asked to extract candidate answers for all possible questions. Therefore, besides answerable questions, we also include unanswerable cases as positive examples, and consider the *plausible answer* as gold answer[2]. In order to not conflict with no-answer detection, we propose to use a multi-head pointer network to additionally produce another pair of span scores $\tilde{\alpha}$ and $\tilde{\beta}$:

$$\tilde{o}_j = \tilde{w}_v^{\mathrm{T}} v_j \ , \ \tilde{t} = \sum_{j=1}^{l_q} \frac{e^{\tilde{o}_j}}{\sum_{k=1}^{l_q} e^{\tilde{o}_k}} v_j$$

$$\tilde{\alpha}, \tilde{\beta} = \mathrm{pointer\_network}(U, \tilde{t})$$

where multiple heads share the same network architecture but with different parameters.

Then, we define an independent span loss as:

$$\mathcal{L}_{indep-I} = -\log \left(\frac{e^{\tilde{\alpha}_{\tilde{a}} \tilde{\beta}_{\tilde{b}}}}{\sum_{i=1}^{l_p} \sum_{j=1}^{l_p} e^{\tilde{\alpha}_i \tilde{\beta}_j}} \right)$$

where $\tilde{a}$ and $\tilde{b}$ are the augmented ground-truth answer boundaries. The final span probability is obtained using a simple mean pooling over the two pairs of softmax-normalized span scores.

[2]In SQuAD 2.0, the plausible answer is annotated by human for every unanswerable question. A pre-trained reader can also be used to extract plausible answers if no annotation is provided.

Independent No-Answer Loss Despite a multi-head pointer network being used to prevent the confliction problem, no-answer detection can still be weakened since the no-answer score z is normalized with span scores. Therefore, we consider exclusively encouraging the prediction on no-answer detection. This is achieved by introducing an independent no-answer loss as:

$$\mathcal{L}_{indep-II} = -(1-\delta) \log \sigma(z) - \delta \log(1 - \sigma(z))$$

where σ is the sigmoid activation function. Through this loss, we expect the model to produce a more confident prediction on no-answer score z without considering the shared-normalization operation.

Finally, we combine the above losses as follows:

$$\mathcal{L} = \mathcal{L}_{joint} + \gamma \mathcal{L}_{indep-I} + \lambda \mathcal{L}_{indep-II}$$

where γ and λ are two hyper-parameters that control the weight of two auxiliary losses.

Answer Verifier

After the answer is extracted, an answer verifier is used to compare the answer sentence with the question, so as to recognize local textual entailment that supports the answer. Here, we define the *answer sentence* as the context sentence that contains either gold answers or plausible answers. We explore three different architectures, as shown in Figure 2: (1) a sequential model that takes the inputs as a long sequence, (2) an interactive model that encodes two sentences interdependently, and (3) a hybrid model that takes both of the two approaches into account.

Model-I: Sequential Architecture In Model-I, we convert the answer sentence and the question along with the extracted answer into an ordered input sequence. Then we adapt the recently proposed Generative Pre-trained Transformer (OpenAI GPT) (Radford et al. 2018) to perform the task. The model is a multi-layer Transformer decoder (Liu et al. 2018a), which is first trained with a language modeling objective on a large unlabeled text corpus and then finetuned on the specific target task.

Specifically, given an answer sentence S, a question Q and an extracted answer A, we concatenate the two sentences with the answer while adding a delimiter token in between to get $[S; Q; \$; A]$. We then embed the sequence with its word embedding as well as position embedding. Multiple transformer blocks are used to encode the sequence embeddings as follows:

$$h_0 = W_e[X] + W_p$$

$$h_i = \mathrm{transformer\_block}(h_{i-1}), \forall i \in [1, n]$$

where X denotes the sequence's indexes in the vocab, W_e is the token embedding matrix, W_p is the position embedding matrix, and n is the number of transformer blocks. Each block consists of a masked multi-head self-attention layer (Vaswani et al. 2017) and a position-wise feed-forward layer. Residual connection and layer normalization are used after each layer.

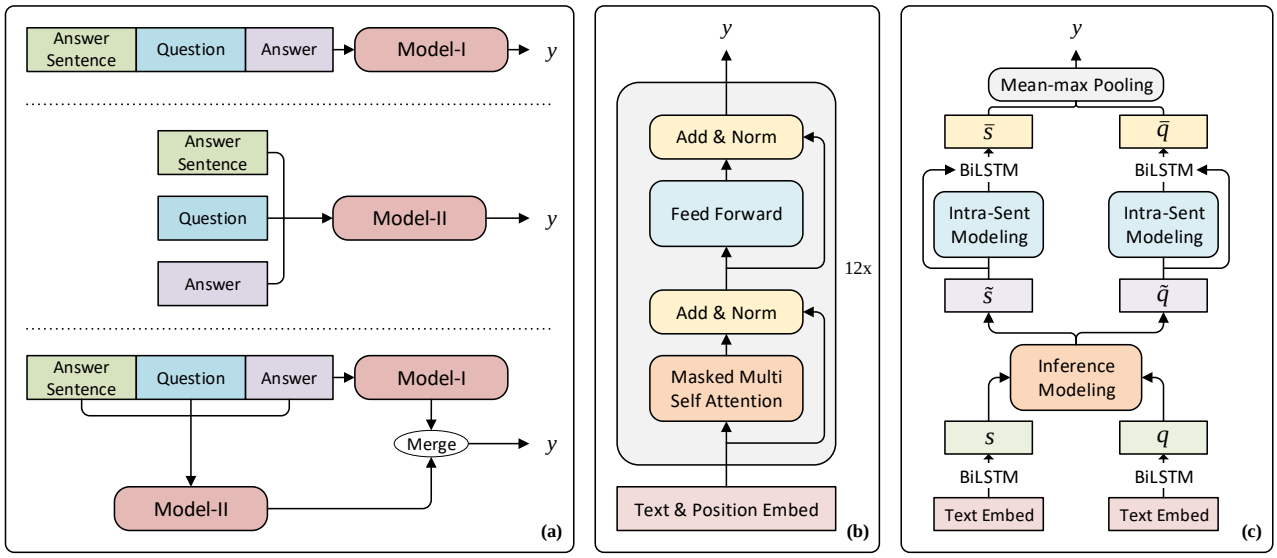

Figure 2: An overview of answer verifiers. (a) Input structures for running three different models. (b) Generative Pre-trained Transformer proposed by Radford et al. (2018). Here, "Masked Multi Self Attention" refers to multi-head self-attention function (Vaswani et al. 2017) that only attends to previous tokens. "Add & Norm" indicates residual connection and layer normalization. (c) Our proposed token-wise interaction model, which is designed to compare two sentences and aggregate the results for verifying the answer.

The last token's activation $h_n^{l_m}$ is then fed into a linear projection layer followed by a $\mathrm{softmax}$ function to output the no-answer probability y:

$$p(y|X) = \mathrm{softmax}(h_n^{l_m} W_y)$$

A standard cross-entropy objective is used to minimize the negative log-likelihood:

$$\mathcal{L}(\theta) = -\sum_{(X,y)} \log p(y|X)$$

Model-II: Interactive Architecture In Model-II, we consider an interactive architecture that aims to capture the interactions between two sentences, so as to recognize their local entailment relationships for verifying the answer. This model consists of the following layers:

Encoding: We embed words using the GloVe embedding (Pennington, Socher, and Manning 2014), and also embed characters of each word with trainable vectors. We run a bidirectional LSTM (BiLSTM) (Hochreiter and Schmidhuber 1997) to encode the characters and concatenate two last hidden states to get character-level embeddings. In addition, we use a binary feature to indicate if a word is part of the answer. All embeddings along with the feature are then concatenated and encoded by a weight-shared BiLSTM, yielding two series of contextual representations:

$$s_i = \mathrm{BiLSTM}([\mathrm{word}_i^s; \mathrm{char}_i^s; \mathrm{fea}_i^s]), \forall i \in [1, l_s]$$
$$q_j = \mathrm{BiLSTM}([\mathrm{word}_j^q; \mathrm{char}_j^q; \mathrm{fea}_j^q]), \forall j \in [1, l_q]$$

where l_s is the length of answer sentence, and $[\cdot;\cdot]$ denotes concatenation.

Inference Modeling: An inference modeling layer is used to capture the interactions between two sentences and produce two inference-aware sentence representations. We first compute the dot products of all tuples $< s_i, q_j >$ as attention weights, and then normalize these weights so as to obtain attended vectors as follows:

$$a_{ij} = s_i^\mathsf{T} q_j, \forall i \in [1, l_s], \forall j \in [1, l_q]$$
$$b_i = \sum_{j=1}^{l_q} \frac{e^{a_{ij}}}{\sum_{k=1}^{l_q} e^{a_{ik}}} q_j \ , \ c_j = \sum_{i=1}^{l_s} \frac{e^{a_{ij}}}{\sum_{k=1}^{l_s} e^{a_{kj}}} s_i$$

Here, b_i refers to the attended vector from question Q for the i-th word in answer sentence S, and vice versa for c_j.

Next, in order to separately compare the aligned pairs $\{(s_i, b_i)\}_{i=1}^{l_s}$ and $\{(q_j, c_j)\}_{j=1}^{l_q}$ for finding local inference information, we use a weight-shared function F to model these aligned pairs as:

$$\tilde{s}_i = F(s_i, b_i) \ , \ \tilde{q}_j = F(q_j, c_j)$$

F can have various forms, such as BiLSTM, multilayer perceptron, and so on. Here we use a heuristic function $o = F(x, y)$ proposed by Hu et al. (2018a), which demonstrates good performances compared to other options:

$$r = \mathrm{gelu}\left(W_r[x; y; x \circ y; x - y]\right)$$
$$g = \sigma\left(W_g[x; y; x \circ y; x - y]\right)$$
$$o = g \circ r + (1 - g) \circ x$$

where gelu is the Gaussian Error Linear Unit (Hendrycks and Gimpel 2016), $\circ$ is element-wise multiplication, and the bias term is omitted.

Intra-Sentence Modeling: Next we apply an intra-sentence modeling layer to capture self correlations inside each sentence. The input are inference-aware vectors $\tilde{s}_i$ and $\tilde{q}_j$, which are first passed through another BiLSTM layer for encoding. We then use the same attention mechanism described above, only now between each sentence and itself, and we set $a_{ij} = -inf$ if $i = j$ to ensure that the word is not aligned with itself. Another function F is used to produce self-aware vectors $\hat{s}_i$ and $\hat{q}_j$ respectively.

Prediction: Before the final prediction, we apply a concatenated residual connection and model the sentences with a BiLSTM as:

$$\bar{s}_i = \text{BiLSTM}([\tilde{s}_i; \hat{s}_i]) \ , \ \bar{q}_j = \text{BiLSTM}([\tilde{q}_j; \hat{q}_j])$$

A mean-max pooling operation is then applied to summarize the final representation of two sentences, namely $\bar{s}_i$ and $\bar{q}_j$. All summarized vectors are then concatenated and fed into a feed-forward classifier that consists of a projection sublayer with gelu activation and a softmax output sublayer, yielding the no-answer probability. As before, we optimize the negative log-likelihood objective function.

Model-III: Hybrid Architecture To explore how the features extracted by Model-I and Model-II can be integrated to obtain better representation capacities, we investigate the combination of the above two models, namely Model-III. We merge the output vectors of two models into a single joint representation. An unified feed-forward classifier is then applied to output the no-answer probability. Such design allows us to test whether the performance can benefit from the integration of two different architectures. In practice we use a simple concatenation to merge the two sources of information.

Experimental Setup

Dataset

We evaluate our approach on the SQuAD 2.0 dataset (Rajpurkar, Jia, and Liang 2018). SQuAD 2.0 is a new machine reading comprehension benchmark that aims to test the models whether they have truely understood the questions by knowing what they don't know. It combines answerable questions from the previous SQuAD 1.1 dataset (Rajpurkar et al. 2016) with 53,775 unanswerable questions about the same passages. Crowdsourcing workers craft these questions with a plausible answer in mind, and make sure that they are relevant to the corresponding passages.

Training and Inference

Our no-answer reader is trained on context passages, while the answer verifier is trained on oracle answer sentences. Model-I follows a procedure of unsupervised pre-training and supervised fine-tuning. That is, the model is first optimized with a language modeling objective on a large unlabeled text corpus to initialize its parameters. Then it adapts the parameters to the answer verifying task with our supervised objective. For Model-II, we directly train it with the supervised loss. Model-III, however, consists of two different architectures that require different training procedures. Therefore, we initialize Model-III with the pre-trained parameters from both of Model-I and Model-II, and then fine-tune the whole model until convergence.

At test time, the reader first predicts a candidate answer as well as a passage-level no-answer probability. The answer verifier then validates the extracted answer along with its sentence and outputs a sentence-level probability. Following the official evaluation setting, a question is detected to be unanswerable once the joint no-answer probability, which is computed as the mean of the above two probabilities, exceeds some threshold. We tune this threshold to maximize F1 score on the development set, and report both of EM (Exact Match) and F1 metrics. We also evaluate the performance on no-answer detection with an accuracy metric (ACC), where its threshold is set as 0.5 by default.

Implementation

We use the Reinforced Mnemonic Reader (RMR) (Hu et al. 2018a), one of the state-of-the-art reading comprehension models on the SQuAD 1.1 dataset, as our base reader. The reader is configurated with its default setting, and trained with the no-answer objective with our auxiliary losses. ELMo (Embeddings from Language Models) (Peters et al. 2018) is exclusively listed in our experimental configuration. We run a grid search on γ and λ among [0.1, 0.3, 0.5, 0.7, 1, 2]. Based on the performance on development set, we set γ as 0.3 and λ to be 1. As for answer verifiers, we use the original configuration from Radford et al. (2018) for Model-I. For Model-II, the Adam optimizer (Kingma and Ba 2014) with a learning rate of 0.0008 is used, the hidden size is set as 300, and a dropout (Srivastava et al. 2014) of 0.3 is applied for preventing overfitting. The batch size is 48 for the reader, 64 for Model-II, and 32 for Model-I as well as Model-III. We use the GloVe (Pennington, Socher, and Manning 2014) 100D embeddings for the reader, and 300D embeddings for Model-II and Model-III. We utilize the *nltk* tokenizer[3] to preprocess passages and questions, as well as split sentences. The passages and the sentences are truncated to not exceed 300 words and 150 words respectively.

Evaluation

Main Results

We first submit our approach on the hidden test set of SQuAD 2.0 for evaluation, which is shown in Table 1. We use Model-III as the default answer verifier, and only report the best result. As we can see, our system obtains state-of-the-art results by achieving an EM score of 71.7 and a F1 score of 74.2 on the test set. Notice that SLQA+ has reached a comparable result compared to our approach. We argue that its promising result is largely due to its superior performance compared to our base reader[4].

Ablation Study

Next, we do an ablation study on the SQuAD 2.0 development set to show the effects of our proposed methods for

[3] https://www.nltk.org/

[4] SLQA+ achieves 87.0 F1 on the SQuAD 1.1 test set, while RMR reaches 86.6.

Model	Dev		Test	
	EM	F1	EM	F1
BNA[1]	59.8	62.6	59.2	62.1
DocQA[2]	61.9	64.8	59.3	62.3
DocQA + ELMo	65.1	67.6	63.4	66.3
ARRR†	-	-	68.6	71.1
VS^3−Net†	-	-	68.4	71.3
SAN[3]	-	-	68.6	71.4
FusionNet++(ensemble)[4]	-	-	70.3	72.6
SLQA+[5]	-	-	71.5	**74.4**
RMR + ELMo + Verifier	**72.3**	**74.8**	**71.7**	74.2
Human	86.3	89.0	86.9	89.5

Table 1: Comparison of different approaches on the SQuAD 2.0 test set, extracted on Aug 28, 2018: Levy et al. (2017)[1], Clark et al. (2018)[2], Liu et al. (2018b)[3], Huang et al. (2018)[4] and Wang et al. (2018)[5]. † indicates unpublished works.

Configuration	HasAns		All		NoAns
	EM	F1	EM	F1	ACC
RMR	72.6	81.6	66.9	69.1	73.1
- indep-I	71.3	80.4	66.0	68.6	72.8
- indep-II	72.4	81.4	64.0	66.1	69.8
- both	71.9	80.9	65.2	67.5	71.4
RMR + ELMo	79.4	86.8	71.4	73.7	77.0
- indep-I	78.9	86.5	71.2	73.5	76.7
- indep-II	79.5	86.6	69.4	71.4	75.1
- both	78.7	86.2	70.0	71.9	75.3

Table 2: Comparison of readers with different auxiliary losses.

each individual component. Table 2 first shows the ablation results of different auxiliary losses on the reader. Removing the independent span loss (indep-I) results in a performance drop for all answerable questions (HasAns), indicating that this loss helps the model in better identifying the answer boundary. Ablating independent no-answer loss (indep-II), on the other hand, causes little influence on HasAns, but leads to a severe decline on no-answer accuracy (NoAns ACC). This suggests that a confliction between answer extraction and no-answer detection indeed happens. Finally, deleting both of two losses causes a degradation of more than 1.5 points on the overall performance in terms of F1, with or without ELMo embeddings.

Table 3 details the results of various architectures for the answer verifier. Model-III outperforms all of other competitors, achieving a no-answer accuracy of 76.2. This illustrates that the combination of two different architectures can bring in further improvement. Adding ELMo embeddings, however, does not boost the performance. We hythosize that the bytepair encoding (Sennrich, Haddow, and Birch 2016) from Model-I and the word/character embeddings from Model-II have provided enough representation capacities.

Configuration	NoAns ACC
Model-I	74.5
Model-II	74.6
Model-II + ELMo	75.3
Model-III	**76.2**
Model-III + ELMo	76.1

Table 3: Comparison of different architectures for the answer verifier.

Configuration	All		NoAns
	EM	F1	ACC
RMR	66.9	69.1	73.1
+ Model-I	68.3	71.1	76.2
+ Model-II	68.1	70.8	75.6
+ Model-II + ELMo	68.2	70.9	75.9
+ Model-III	**68.5**	**71.5**	**77.1**
+ Model-III + ELMo	68.5	71.2	76.5
RMR + ELMo	71.4	73.7	77.0
+ Model-I	71.8	74.4	77.3
+ Model-II	71.8	74.2	78.1
+ Model-II + ELMo	72.0	74.3	78.2
+ Model-III	**72.3**	**74.8**	**78.6**
+ Model-III + ELMo	71.8	74.3	78.3

Table 4: Comparison of readers with different answer verifiers.

After doing separate ablations on each component, we then compare the performance of the whole system, as shown in Table 4. The combination of base reader with any answer verifier can always result in considerable performance gains, and combining the reader with Model-III obtains the best result. We find that the improvement on no-answer accuracy is significant. This metric raises from 73.1 to 77.1 after adding Model-III to RMR, increasing by 4 absolute points. Similar observation can be found when ELMo embeddings are used, demonstrating that the gains are consistent and stable.

In order to investigate how the readers affect the overall performance, we fix the answer verifier as Model-III and use DocQA (Clark and Gardner 2018) as the base reader instead of RMR, as shown in Table 5. We find that the absolute improvements are even larger: the no-answer accuracy roughly increases by 6 points when adding Model-III to DocQA (from 69.1 to 75.2), and 5.5 points when adding Model-III to DocQA + ELMo (from 70.6 to 76.1).

Finally, we plot the precision-recall curves of F1 score on the development set in Figure 3. We observe that RMR + ELMo + Verifier achieves the best precision when the recall is less than 80. After the recall exceeds 80, the precision of RMR + ELMo becomes slightly better. Ablating two auxiliary losses, however, leads to an overall degradation on the curve, but it still outperforms the baseline by a large margin.

Configuration	All		NoAns
	EM	F1	ACC
DocQA	61.9	64.8	69.1
+ Model-III	**66.5**	**69.2**	**75.2**
DocQA + ELMo	65.1	67.6	70.6
+ Model-III	**68.0**	**70.7**	**76.1**

Table 5: Comparison of different readers with fixed answer verifier.

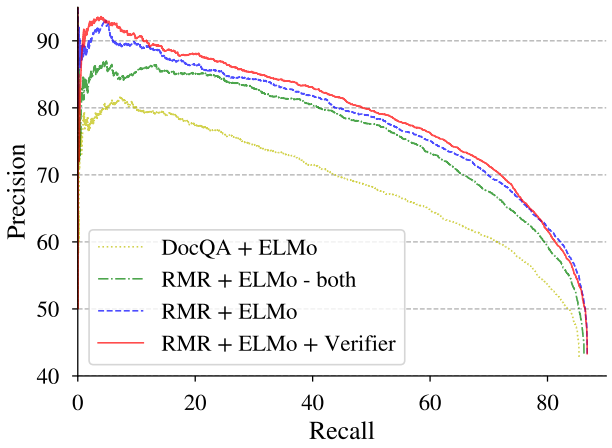

Figure 3: Precision-Recall curves of F1 score.

Error Analysis

To perform error analysis, we first categorize all examples on the development set into 5 classes:

- *Case1*: the question is *answerable*, the no-answer probability is *less* than the threshold, and the answer is *correct*.
- *Case2*: the question is *unanswerable*, and the no-answer probability is *larger* than the threshold.
- *Case3*: almost the same as case1, except that the predicted answer is *wrong*.
- *Case4*: the question is *unanswerable*, but the no-answer probability is *less* than the threshold.
- *Case5*: the question is *answerable*, but the no-answer probability is *larger* than the threshold.

We then show the percentage of each category in Table 6. As we can see, the base reader trained with auxiliary losses is notably better at *case2* and *case4* compared to the baseline, implying that our proposed losses help the model mainly improve upon unanswerable cases. After adding the answer verifier, we observe that although the system's performance on unanswerable cases slightly decreases, the results on *case1* and *case5* have been improved. This demonstrates that the answer verifier does well on detecting answerable question rather than unanswerable one. Besides, we find that the error of answer extraction is relatively small (6.5% for Case3 in RMR + ELMo + Verifier). However, the classification error on no-answer detection is much larger. More than 20% of examples are misclassified even with our best system (10.3% for Case4 and 10.9% for Case5 in RMR + ELMo + Verifier). Therefore, we argue that the main performance bottleneck lies in no-answer detection instead of answer extraction.

Next, to understand the challenges our approach faces, we manually investigate 50 incorrectly predicted unanswerable examples (based on F1) that are randomly sampled from the development set. Following the types of negative examples defined by Rajpurkar et al. (2018), we categorize the sampled examples and show them in Table 7. As we can see, our system is good at recognize *negation* and *antonym*. The frequency of negation decreases from 9% to 0% and only 4 antonym examples are predicted wrongly. We think that this is because the two types are relatively easier to identify. Both of negation and antonym only require to detect one single word in the question, such as "*never*" or "*not*" for negation and "*increase*" to "*decrease*" for antonym. However, *impossible condition* and *other neutral* types roughly acount for 46% of the error set, indicating that our system performs less effectively on these more difficult cases.

Related Work

Reading Comprehension Datasets. Various large-scale reading comprehension datasets, such as cloze-style test (Hermann et al. 2015), answer extraction benchmark (Rajpurkar et al. 2016; Joshi et al. 2017) and answer generation benchmark (Nguyen et al. 2016; Kočiskỳ et al. 2018), have been proposed. However, these datasets still guarantee that the given context must contain an answer. Recently, some works construct negative examples by retrieving passages for existing questions based on Lucene (Tan et al. 2018) and TF-IDF (Clark and Gardner 2018), or using crowdworkers to craft unanswerable questions (Rajpurkar, Jia, and Liang 2018). Compared to automatically retrieved negative examples, human-annotated examples are more difficult to detect for two reasons: (1) the questions are relevant to the passage and (2) the passage contains a plausible answer to the question. Therefore, we choose to work on the SQuAD 2.0 dataset in this paper.

Neural Networks for Reading Comprehension. Neural reading models typically leverage various attention mechanisms to build interdependent representations of passage and question, and sequentially predict the answer boundary (Seo et al. 2017; Hu et al. 2018a; Wang et al. 2017; Yu et al. 2018; Hu et al. 2018b). However, these approaches are not designed to handle no-answer cases. To address this problem, previous works (Levy et al. 2017; Clark and Gardner 2018; Kundu and Ng 2018) predict a no-answer probability in addition to the distribution over answer spans, so as to jointly learn no-answer detection as well as answer extraction. Our no-answer reader extends existing approaches by introducing two auxiliary losses that enhance these two tasks independently.

Recognizing Textual Entailment. Recognizing textual entailment (RTE) (Dagan et al. 2010; Marelli et al. 2014), or known as natural language inference (NLI) (Bowman et al. 2015), requires systems to understand entailment, contra-

Configuration	Case1 ✓	Case2 ✓	Case3 ✗	Case4 ✗	Case5 ✗
RMR - both	27.8%	37.3%	6.5%	12.7%	15.7%
RMR	27%	**39.9%**	**5.9%**	**10.2%**	17%
RMR + Verifier	**30.3%**	38.2%	8.4%	11.8%	**11.3%**
RMR + ELMo - both	31.5%	38.3%	5.6%	11.8%	12.8%
RMR + ELMo	31.2%	**40.2%**	**5.5%**	**9.9%**	13.2%
RMR + ELMo + Verifier	**32.5%**	39.8%	6.5%	10.3%	**10.9%**

Table 6: Percentage of five categories. Correct predictions are denoted with ✓, while wrong cases are marked with ✗.

Phenomenon	Percentage	
	All	Error
Negation	9%	0%
Antonym	20%	8%
Entity Swap	21%	24%
Mutual Exclusion	15%	16%
Impossible Condition	4%	14%
Other Neutral	24%	32%
Answerable	7%	6%

Table 7: Linguistic phenomena exhibited by all negative examples (statistics from Rajpurkar et al. (2018)) and sampled error cases of RMR + ELMo + Verifier.

diction or semantic neutrality between two sentences. This task is strongly related to no-answer detection, where the machine needs to understand if the passage and the question supports the answer. To recognize entailment, various branches of works have been proposed, including encoding-based approach (Bowman et al. 2016; Mou et al. 2015), interaction-based approach (Parikh et al. 2016; Chen et al. 2016) and sequence-based approach (Radford et al. 2018). In this paper we investigate the last two branches and further propose a hybrid architecture that combines both of them properly.

Answer Validation. Early answer validation task (Magnini et al. 2002) aims at ranking multiple candidate answers to return a most reliable one. Later, the answer validation exercise (Rodrigo, Peñas, and Verdejo 2008) has been proposed to decide whether an answer is correct or not according to a given supporting text and a question, but the dataset is too small for neural network-based approaches. Recently, Tan et al. (2018) propose to validate the candidate answer for detecting unanswerable questions, by comparing the question with the passage. Our answer verifier, on the contrary, denoises the passage by comparing questions with answer sentences, so as to focus on finding local entailment that supports the answer.

Conclusion

We proposed a read-then-verify system that is able to abstain from answering when a question has no answer given the passage. We first introduce two auxiliary losses to help the reader concentrate on answer extraction and no-answer detection respectively, and then utilize an answer verifier to validate the legitimacy of the predicted answer, in which three different architectures are investigated. Our system has achieved state-of-the-art results on the SQuAD 2.0 dataset at the time of submission (Aug. 28th, 2018). Looking forward, we plan to design new structures for answer verifiers to handle questions with more complicated inferences.

Acknowledgments

We would like to thank Pranav Rajpurkar and Robin Jia for their helps with SQuAD 2.0 submissions. This work is supported by the Major State Research Development Program (2016YFB0201305).

References

Bowman, S. R.; Angeli, G.; Potts, C.; and Manning, C. D. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of EMNLP*.

Bowman, S. R.; Gauthier, J.; Rastogi, A.; Gupta, R.; Manning, C. D.; and Potts, C. 2016. A fast unified model for parsing and sentence understanding. *arXiv preprint arXiv:1603.06021*.

Chen, Q.; Zhu, X.; Ling, Z.; Wei, S.; Jiang, H.; and Inkpen, D. 2016. Enhanced lstm for natural language inference. *arXiv preprint arXiv:1609.06038*.

Clark, C., and Gardner, M. 2018. Simple and effective multi-paragraph reading comprehension. In *Proceedings of ACL*.

Dagan, I.; Dolan, B.; Magnini, B.; and Roth, D. 2010. Recognizing textual entailment: rational, evaluation and approaches. *Natural Language Engineering* 16(1):105–105.

Hendrycks, D., and Gimpel, K. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *arXiv preprint arXiv:1606.08415*.

Hermann, K. M.; Kocisky, T.; Grefenstette, E.; Espeholt, L.; Kay, W.; Suleyman, M.; and Blunsom, P. 2015. Teaching machines to read and comprehend. In *Proceedings of NIPS*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Hu, M.; Peng, Y.; Huang, Z.; Qiu, X.; Wei, F.; and Zhou, M. 2018a. Reinforced mnemonic reader for machine reading comprehension. In *Proceedings of IJCAI*.

Hu, M.; Peng, Y.; Wei, F.; Huang, Z.; DongshengLi; Yang, N.; and Zhou, M. 2018b. Attention-guided answer distilla-

tion for machine reading comprehension. In *Proceedings of EMNLP*.

Huang, H.-Y.; Zhu, C.; Shen, Y.; and Chen, W. 2018. Fusionnet: fusing via fully-aware attention with application to machine comprehension. In *Proceedings of ICLR*.

Joshi, M.; Choi, E.; Weld, D. S.; and Zettlemoyer, L. 2017. Triviaqa: a large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of ACL*.

Kingma, D. P., and Ba, L. J. 2014. Adam: A method for stochastic optimization. In *CoRR, abs/1412.6980*.

Kočiskỳ, T.; Schwarz, J.; Blunsom, P.; Dyer, C.; Hermann, K. M.; Melis, G.; and Grefenstette, E. 2018. The narrativeqa reading comprehension challenge. *Transactions of ACL* 6:317–328.

Kundu, S., and Ng, H. T. 2018. A nil-aware answer extraction framework for question answering. In *Proceedings of EMNLP*, 4243–4252.

Levy, O.; Seo, M.; Choi, E.; and Zettlemoyer, L. 2017. Zero-shot relation extraction via reading comprehension. *arXiv preprint arXiv:1706.04115*.

Liu, P. J.; Saleh, M.; Pot, E.; Goodrich, B.; Sepassi, R.; Kaiser, L.; and Shazeer, N. 2018a. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*.

Liu, X.; Shen, Y.; Duh, K.; and Gao, J. 2018b. Stochastic answer networks for machine reading comprehension. In *Proceedings of ACL*.

Magnini, B.; Negri, M.; Prevete, R.; and Tanev, H. 2002. Is it the right answer? exploiting web redundancy for answer validation. In *Proceedings of ACL*.

Marelli, M.; Menini, S.; Baroni, M.; Bentivogli, L.; Bernardi, R.; Zamparelli, R.; et al. 2014. A sick cure for the evaluation of compositional distributional semantic models. In *LREC*, 216–223.

Mou, L.; Men, R.; Li, G.; Xu, Y.; Zhang, L.; Yan, R.; and Jin, Z. 2015. Natural language inference by tree-based convolution and heuristic matching. *arXiv preprint arXiv:1512.08422*.

Nguyen, T.; Rosenberg, M.; Song, X.; Gao, J.; Tiwary, S.; Majumder, R.; and Deng, L. 2016. Ms marco: a human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Parikh, A. P.; Täckström, O.; Das, D.; and Uszkoreit, J. 2016. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*.

Peters, M. E.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; and Zettlemoyer, L. 2018. Deep contextualized word prepresentations. In *Proceedings of NAACL*.

Radford, A.; Narasimhan, K.; Salimans, T.; and Sutskever, I. 2018. Improving language understanding by generative pre-training.

Rajpurkar, P.; Zhang, J.; Lopyrev, K.; and Liang, P. 2016. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of EMNLP*.

Rajpurkar, P.; Jia, R.; and Liang, P. 2018. Know what you don't know: unanswerable questions for squad. In *Proceedings of ACL*.

Rodrigo, Á.; Peñas, A.; and Verdejo, F. 2008. Overview of the answer validation exercise 2008. In *Workshop of CLEF*, 296–313. Springer.

Sennrich, R.; Haddow, B.; and Birch, A. 2016. Neural machine translation of rare words with subword units. In *Proceedings of ACL*.

Seo, M.; Kembhavi, A.; Farhadi, A.; and Hajishirzi, H. 2017. Bidirectional attention flow for machine comprehension. In *Proceedings of ICLR*.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR* 1929–1958.

Tan, C.; Wei, F.; Zhou, Q.; Yang, N.; Lv, W.; and Zhou, M. 2018. I know there is no answer: modeling answer validation for machine reading comprehension. In *Proceedings of NLPCC*, 85–97. Springer.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Proceedings of NIPS*, 5998–6008.

Vinyals, O.; Fortunato, M.; and Jaitly, N. 2015. Pointer networks. In *Proceedings of NIPS*.

Wang, W.; Yang, N.; Wei, F.; Chang, B.; and Zhou, M. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of ACL*.

Wang, W.; Yan, M.; and Wu, C. 2018. Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. In *Proceedings of ACL*.

Yu, A. W.; Dohan, D.; Luong, M.-T.; Zhao, R.; Chen, K.; Norouzi, M.; and Le, Q. V. 2018. Qanet: combining local convolution with global self-attention for reading comprehension. In *Proceedings of ICLR*.