Simple and Effective Multi-Paragraph Reading Comprehension

Christopher Clark[*]
University of Washington
csquared@cs.washington.edu

Matt Gardner
Allen Institute for Artificial Intelligence
mattg@allenai.org

Abstract

We introduce a method of adapting neural paragraph-level question answering models to the case where entire documents are given as input. Most current question answering models cannot scale to document or multi-document input, and naively applying these models to each paragraph independently often results in them being distracted by irrelevant text. We show that it is possible to significantly improve performance by using a modified training scheme that teaches the model to ignore non-answer containing paragraphs. Our method involves sampling multiple paragraphs from each document, and using an objective function that requires the model to produce globally correct output. We additionally identify and improve upon a number of other design decisions that arise when working with document-level data. Experiments on TriviaQA and SQuAD shows our method advances the state of the art, including a 10 point gain on TriviaQA.

1 Introduction

Teaching machines to answer arbitrary user-generated questions is a long-term goal of natural language processing. For a wide range of questions, existing information retrieval methods are capable of locating documents that are likely to contain the answer. However, automatically extracting the answer from those texts remains an open challenge. The recent success of neural models at answering questions given a related paragraph (Wang et al., 2017c; Tan et al., 2017) suggests they have the potential to be a key part of a solution to this problem. Most neural models are unable to scale beyond short paragraphs, so typically this requires adapting a paragraph-level model to process document-level input.

There are two basic approaches to this task. Pipelined approaches select a single paragraph from the input documents, which is then passed to the paragraph model to extract an answer (Joshi et al., 2017; Wang et al., 2017a). Confidence based methods apply the model to multiple paragraphs and return the answer with the highest confidence (Chen et al., 2017a). Confidence methods have the advantage of being robust to errors in the (usually less sophisticated) paragraph selection step, however they require a model that can produce accurate confidence scores for each paragraph. As we shall show, naively trained models often struggle to meet this requirement.

In this paper we start by proposing an improved pipelined method which achieves state-of-the-art results. Then we introduce a method for training models to produce accurate per-paragraph confidence scores, and we show how combining this method with multiple paragraph selection further increases performance.

Our pipelined method focuses on addressing the challenges that come with training on document-level data. We use a linear classifier to select which paragraphs to train and test on. Since annotating entire documents is expensive, data of this sort is typically distantly supervised, meaning only the answer text, not the answer spans, are known. To handle the noise this creates, we use a summed objective function that marginalizes the model's output over all locations the answer text occurs. We apply this approach with a model design that integrates some recent ideas in reading comprehension models, including self-attention (Cheng et al., 2016) and bi-directional attention (Seo et al., 2016).

[*]Work completed while interning at the Allen Institute for Artificial Intelligence

Our confidence method extends this approach to better handle the multi-paragraph setting. Previous approaches trained the model on questions paired with paragraphs that are known *a priori* to contain the answer. This has several downsides: the model is not trained to produce low confidence scores for paragraphs that do not contain an answer, and the training objective does not require confidence scores to be comparable between paragraphs. We resolve these problems by sampling paragraphs from the context documents, including paragraphs that do not contain an answer, to train on. We then use a shared-normalization objective where paragraphs are processed independently, but the probability of an answer candidate is marginalized over all paragraphs sampled from the same document. This requires the model to produce globally correct output even though each paragraph is processed independently.

We evaluate our work on TriviaQA (Joshi et al., 2017) in the wiki, web, and unfiltered setting. Our model achieves a nearly 10 point lead over published prior work. We additionally perform an ablation study on our pipelined method, and we show the effectiveness of our multi-paragraph methods on a modified version of SQuAD (Rajpurkar et al., 2016) where only the correct document, not the correct paragraph, is known. Finally, we combine our model with a web search backend to build a demonstration end-to-end QA system[1], and show it performs well on questions from the TREC question answering task (Voorhees et al., 1999). We release our code[2] to facilitate future work.

2 Pipelined Method

In this section we propose a pipelined QA system, where a single paragraph is selected and passed to a paragraph-level question answering model.

2.1 Paragraph Selection

If there is a single source document, we select the paragraph with the smallest TF-IDF cosine distance with the question. Document frequencies are computed using the individual paragraphs within the document. If there are multiple input documents, we found it beneficial to use a linear classifier that uses the same TF-IDF score, whether the paragraph was the first in its document, how many tokens preceded it, and the number of question words it includes as features. The classifier is trained on the distantly supervised objective of selecting paragraphs that contain at least one answer span. On TriviaQA web, relative to truncating the document as done by prior work, this improves the chance of the selected text containing the correct answer from 83.1% to 85.1%.

2.2 Handling Noisy Labels

Question: Which British general was killed at Khartoum in 1885?
Answer: Gordon
Context: In February 1885 Gordon returned to the Sudan to evacuate Egyptian forces. Khartoum came under siege the next month and rebels broke into the city, killing Gordon and the other defenders. The British public reacted to his death by acclaiming 'Gordon of Khartoum', a saint. However, historians have suggested that Gordon...

Figure 1: Noisy supervision can cause many spans of text that contain the answer, but are not situated in a context that relates to the question (red), to distract the model from learning from more relevant spans (green).

In a distantly supervised setup we label all text spans that match the answer text as being correct. This can lead to training the model to select unwanted answer spans. Figure 1 contains an example. To handle this difficulty, we use a summed objective function similar to the one from Kadlec et al. (2016), that optimizes the negative log-likelihood of selecting any correct answer span. The models we consider here work by independently predicting the start and end token of the answer span, so we take this approach for both predictions. For example, the objective for predicting the answer start token becomes $-\log\left(\sum_{a\in A} p_a\right)$ where A is the set of tokens that start an answer and p_i is the answer-start probability predicted by the model for token i. This objective has the advantage of being agnostic to how the model distributes probability mass across the possible answer spans, allowing the model to focus on only the most relevant spans.

2.3 Model

We use a model with the following layers (shown in Figure 2):

Embedding: We embed words using pre-trained word vectors. We concatenate these with character-derived word embeddings, which are

[1]https://documentqa.allenai.org
[2]https://github.com/allenai/document-qa

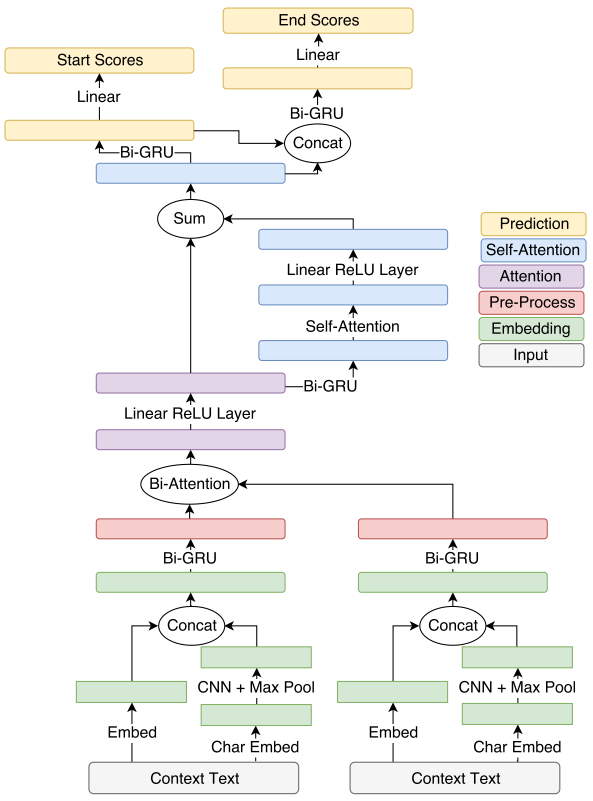

Figure 2: High level outline of our model.

produced by embedding characters using a learned embedding matrix and then applying a convolutional neural network and max-pooling.

Pre-Process: A shared bi-directional GRU (Cho et al., 2014) is used to process the question and passage embeddings.

Attention: The attention mechanism from the Bi-Directional Attention Flow (BiDAF) model (Seo et al., 2016) is used to build a query-aware context representation. Let $\mathbf{h_i}$ and $\mathbf{q_j}$ be the vector for context word i and question word j, and n_q and n_c be the lengths of the question and context respectively. We compute attention between context word i and question word j as:

$$a_{ij} = \mathbf{w_1} \cdot \mathbf{h_i} + \mathbf{w_2} \cdot \mathbf{q_j} + \mathbf{w_3} \cdot (\mathbf{h_i} \odot \mathbf{q_j})$$

where $\mathbf{w_1}$, $\mathbf{w_2}$, and $\mathbf{w_3}$ are learned vectors and $\odot$ is element-wise multiplication. We then compute an attended vector $\mathbf{c_i}$ for each context token as:

$$p_{ij} = \frac{e^{a_{ij}}}{\sum_{j=1}^{n_q} e^{a_{ij}}} \qquad \mathbf{c_i} = \sum_{j=1}^{n_q} \mathbf{q_j} p_{ij}$$

We also compute a query-to-context vector $\mathbf{q_c}$:

$$m_i = \max_{1 \le j \le n_q} a_{ij}$$

$$p_i = \frac{e^{m_i}}{\sum_{i=1}^{n_c} e^{m_i}} \qquad \mathbf{q_c} = \sum_{i=1}^{n_c} \mathbf{h_i} p_i$$

The final vector for each token is built by concatenating $\mathbf{h_i}$, $\mathbf{c_i}$, $\mathbf{h_i} \odot \mathbf{c_i}$, and $\mathbf{q_c} \odot \mathbf{c_i}$. In our model we subsequently pass the result through a linear layer with ReLU activations.

Self-Attention: Next we use a layer of residual self-attention. The input is passed through another bi-directional GRU. Then we apply the same attention mechanism, only now between the passage and itself. In this case we do not use query-to-context attention and we set $a_{ij} = -inf$ if $i = j$.

As before, we pass the concatenated output through a linear layer with ReLU activations. The result is then summed with the original input.

Prediction: In the last layer of our model a bi-directional GRU is applied, followed by a linear layer to compute answer start scores for each token. The hidden states are concatenated with the input and fed into a second bi-directional GRU and linear layer to predict answer end scores. The softmax function is applied to the start and end scores to produce answer start and end probabilities.

Dropout: We apply variational dropout (Gal and Ghahramani, 2016) to the input to all the GRUs and the input to the attention mechanisms at a rate of 0.2.

3 Confidence Method

We adapt this model to the multi-paragraph setting by using the un-normalized and un-exponentiated (i.e., before the softmax operator is applied) score given to each span as a measure of the model's confidence. For the boundary-based models we use here, a span's score is the sum of the start and end score given to its start and end token. At test time we run the model on each paragraph and select the answer span with the highest confidence. This is the approach taken by Chen et al. (2017a).

Our experiments in Section 5 show that these confidence scores can be very poor if the model is only trained on answer-containing paragraphs, as done by prior work. Table 1 contains some qualitative examples of the errors that occur.

We hypothesize that there are two key sources of error. First, for models trained with the softmax objective, the pre-softmax scores for all spans can be arbitrarily increased or decreased by a constant value without changing the resulting softmax probability distribution. As a result, nothing prevents models from producing scores that are arbitrarily all larger or all smaller for one paragraph

Question	Low Confidence Correct Extraction	High Confidence Incorrect Extraction
When is the Members Debate held?	**Immediately after Decision Time** a "Members Debate" is held, which lasts for 45 minutes...	...majority of the Scottish electorate voted for it in a referendum to be held on **1 March 1979** that represented at least...
How many tree species are in the rainforest?	...one 2001 study finding a quarter square kilometer (62 acres) of Ecuadorian rainforest supports more than **1,100** tree species	The affected region was approximately **1,160,000** square miles (3,000,000 km2) of rainforest, compared to 734,000 square miles
Who was Warsz?	....In actuality, Warsz was a 12th/13th century **nobleman** who owned a village located at the modern....	One of the most famous people born in Warsaw was **Maria Sklodowska - Curie**, who achieved international...
How much did the initial LM weight in kg?	The initial LM model weighed approximately 33,300 pounds (**15,000** kg), and...	The module was 11.42 feet (3.48 m) tall, and weighed approximately 12,250 pounds (**5,560** kg)

Table 1: Examples from SQuAD where a model was less confident in a correct extraction from one paragraph (left) than in an incorrect extraction from another (right). Even if the passage has no correct answer and does not contain any question words, the model assigns high confidence to phrases that match the category the question is asking about. Because the confidence scores are not well-calibrated, this confidence is often higher than the confidence assigned to correct answer spans in different paragraphs, even when those correct spans have better contextual evidence.

than another. Second, if the model only sees paragraphs that contain answers, it might become too confident in heuristics or patterns that are only effective when it is known *a priori* that an answer exists. For example, the model might become too reliant on selecting answers that match semantic type the question is asking about, causing it be easily distracted by other entities of that type when they appear in irrelevant text. This kind of error has also been observed when distractor sentences are added to the context (Jia and Liang, 2017)

We experiment with four approaches to training models to produce comparable confidence scores, shown in the following subsections. In all cases we will sample paragraphs that do not contain an answer as additional training points.

3.1 Shared-Normalization

In this approach a modified objective function is used where span start and end scores are normalized across all paragraphs sampled from the same context. This means that paragraphs from the same context use a shared normalization factor in the final softmax operations. We train on this objective by including multiple paragraphs from the same context in each mini-batch. The key idea is that this will force the model to produce scores that are comparable between paragraphs, even though it does not have access to information about what other paragraphs are being considered.

3.2 Merge

As an alternative to the previous method, we experiment with concatenating all paragraphs sampled from the same context together during training. A paragraph separator token with a learned embedding is added before each paragraph.

3.3 No-Answer Option

We also experiment with allowing the model to select a special "no-answer" option for each paragraph. First we re-write our objective as:

$$-\log\left(\frac{e^{s_a}}{\sum_{i=1}^{n} e^{s_i}}\right) - \log\left(\frac{e^{g_b}}{\sum_{j=1}^{n} e^{g_j}}\right) =$$

$$-\log\left(\frac{e^{s_a+g_b}}{\sum_{i=1}^{n}\sum_{j=1}^{n} e^{s_i+g_j}}\right)$$

where s_j and g_j are the scores for the start and end bounds produced by the model for token j, and a and b are the correct start and end tokens. We have the model compute another score, z, to represent the weight given to a "no-answer" possibility. Our revised objective function becomes:

$$-\log\left(\frac{(1-\delta)e^{z} + \delta e^{s_a+g_b}}{e^{z} + \sum_{i=1}^{n}\sum_{j=1}^{n} e^{s_i+g_j}}\right)$$

where δ is 1 if an answer exists and 0 otherwise. If there are multiple answer spans we use the same objective, except the numerator includes the summation over all answer start and end tokens.

We compute z by adding an extra layer at the end of our model. We build input vectors by taking the summed hidden states of the RNNs used to predict the start/end token scores weighed by the start/end probabilities, and using a learned attention vector on the output of the self-attention layer.

These vectors are fed into a two layer network with an 80 dimensional hidden layer and ReLU activations that produces z as its only output.

3.4 Sigmoid

As a final baseline, we consider training models with the sigmoid loss objective function. That is, we compute a start/end probability for each token by applying the sigmoid function to the start/end scores of each token. A cross entropy loss is used on each individual probability. The intuition is that, since the scores are being evaluated independently of one another, they are more likely to be comparable between different paragraphs.

4 Experimental Setup

4.1 Datasets

We evaluate our approach on four datasets: TriviaQA unfiltered (Joshi et al., 2017), a dataset of questions from trivia databases paired with documents found by completing a web search of the questions; TriviaQA wiki, the same dataset but only including Wikipedia articles; TriviaQA web, a dataset derived from TriviaQA unfiltered by treating each question-document pair where the document contains the question answer as an individual training point; and SQuAD (Rajpurkar et al., 2016), a collection of Wikipedia articles and crowdsourced questions.

4.2 Preprocessing

We note that for TriviaQA web we do not subsample as was done by Joshi et al. (2017), instead training on the all 530k training examples. We also observe that TriviaQA documents often contain many small paragraphs, so we restructure the documents by merging consecutive paragraphs together up to a target size. We use a maximum paragraph size of 400 unless stated otherwise. Paragraph separator tokens with learned embeddings are added between merged paragraphs to preserve formatting information. We are also careful to mark all spans of text that would be considered an exact match by the official evaluation script, which includes some minor text pre-processing, as answer spans, not just spans that are an exact string match with the answer text.

4.3 Sampling

Our confidence-based approaches are trained by sampling paragraphs from the context during training. For SQuAD and TriviaQA web we take the top four paragraphs as judged by our paragraph ranking function (see Section 2.1). We sample two different paragraphs from those four each epoch to train on. Since we observe that the higher-ranked paragraphs are more likely to contain the context needed to answer the question, we sample the highest ranked paragraph that contains an answer twice as often as the others. For the merge and shared-norm approaches, we additionally require that at least one of the paragraphs contains an answer span, and both of those paragraphs are included in the same mini-batch. For TriviaQA wiki we repeat the process but use the top 8 paragraphs, and for TriviaQA unfiltered we use the top 16, because much more context is given in these settings.

Model	EM	F1
baseline (Joshi et al., 2017)	41.08	47.40
BiDAF	50.21	56.86
BiDAF + TF-IDF	53.41	59.18
BiDAF + sum	56.22	61.48
BiDAF + TF-IDF + sum	57.20	62.44
our model + TF-IDF + sum	61.10	66.04

Table 2: Results on TriviaQA web using our pipelined method.

4.4 Implementation

We train the model with the Adadelta optimizer (Zeiler, 2012) with a batch size 60 for TriviaQA and 45 for SQuAD. At test time we select the most probable answer span of length less than or equal to 8 for TriviaQA and 17 for SQuAD. The GloVe 300 dimensional word vectors released by Pennington et al. (2014) are used for word embeddings. On SQuAD, we use a dimensionality of size 100 for the GRUs and of size 200 for the linear layers employed after each attention mechanism. We found for TriviaQA, likely because there is more data, using a larger dimensionality of 140 for each GRU and 280 for the linear layers is beneficial. During training, we maintain an exponential moving average of the weights with a decay rate of 0.999. We use the weight averages at test time. We do not update the word vectors during training.

5 Results

5.1 TriviaQA Web and TriviaQA Wiki

First, we do an ablation study on TriviaQA web to show the effects of our proposed methods for our pipeline model. We start with a baseline following the one used by Joshi et al. (2017). This

Model	Web		Web Verified		Wiki		Wiki Verified	
	EM	F1	EM	F1	EM	F1	EM	F1
Baseline (Joshi et al., 2017)	40.74	47.06	49.54	55.80	40.32	45.91	44.86	50.71
Smarnet (Chen et al., 2017b)	40.87	47.09	51.11	55.98	42.41	48.84	50.51	55.90
Mnemonic Reader (Hu et al., 2017)	46.65	52.89	56.96	61.48	46.94	52.85	54.45	59.46
(Weissenborn et al., 2017a)	50.56	56.73	63.20	67.97	48.64	55.13	53.42	59.92
Neural Cascade (Swayamdipta et al., 2017)	53.75	58.57	63.20	66.88	51.59	55.95	58.90	62.53
S-Norm (ours)	66.37	71.32	79.97	83.70	63.99	68.93	67.98	72.88

Table 3: Published TriviaQA results. Our approach advances the state of the art by about 10 points on these datasets[4]

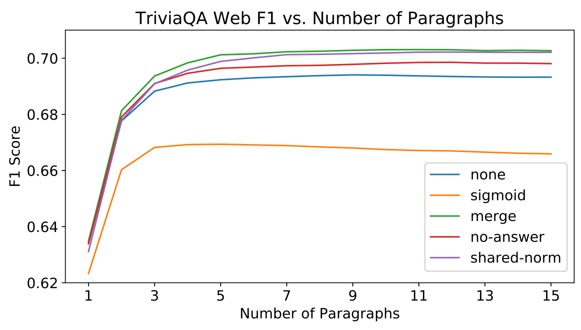

Figure 3: Results on TriviaQA web when applying our models to multiple paragraphs from each document. Most of our training methods improve the model's ability to utilize more text.

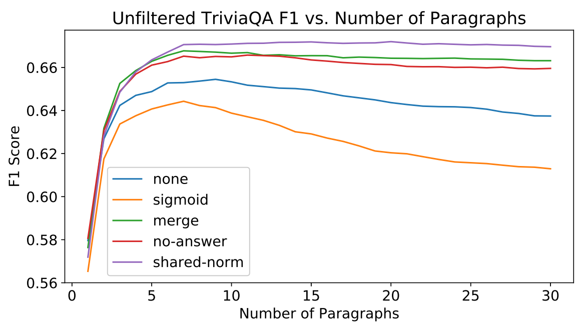

Figure 4: Results for our confidence methods on TriviaQA unfiltered. The shared-norm approach is the strongest, while the baseline model starts to lose performance as more paragraphs are used.

system uses BiDAF (Seo et al., 2016) as the paragraph model, and selects a random answer span from each paragraph each epoch to train on. The first 400 tokens of each document are used during training, and the first 800 during testing. When using the TF-IDF paragraph selection approach, we instead break the documents into paragraphs of size 400 when training and 800 when testing, and select the top-ranked paragraph to feed into the model. As shown in Table 2, our baseline outperforms the results reported by Joshi et al. (2017) significantly, likely because we are not subsampling the data. We find both TF-IDF ranking and the sum objective to be effective. Using our refined model increases the gain by another 4 points.

Next we show the results of our confidence-based approaches. For this comparison we split documents into paragraphs of at most 400 tokens, and rank them using TF-IDF cosine distance. Then we measure the performance of our proposed approaches as the model is used to independently process an increasing number of these paragraphs, and the highest confidence answer is selected as the final output. The results are shown in Figure 3.

On this dataset even the model trained without any of the proposed training methods ("none") improves as more paragraphs are used, showing it does a passable job at focusing on the correct paragraph. The no-answer option training approach lead to a significant improvement, and the shared-norm and merge approaches are even better.

We use the shared-norm approach for evaluation on the TriviaQA test sets. We found that increasing the paragraph size to 800 at test time, and to 600 during training, was slightly beneficial, allowing our model to reach 66.04 EM and 70.98 F1 on the dev set. As shown in Table 3, our model is firmly ahead of prior work on both the TriviaQA web and TriviaQA wiki test sets. Since our submission, a few additional entries have been added to the public leader for this dataset[5], although to the best of our knowledge these results have not yet been published.

5.2 TriviaQA Unfiltered

Next we apply our confidence methods to TriviaQA unfiltered. This dataset is of particular interest because the system is not told which document contains the answer, so it provides a plausible simulation of answering a question using a document

[4]Comparison made of 5/01/2018.

[5]https://competitions.codalab.org/competitions/17208

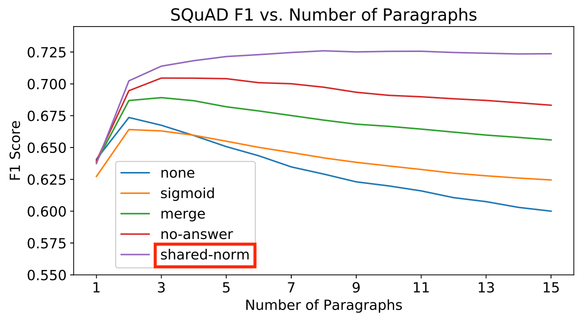

Figure 5: Results for our confidence methods on document-level SQuAD. The shared-norm model is the only model that does not lose performance when exposed to large numbers of paragraphs.

retrieval system. We show the same graph as before for this dataset in Figure 4. Our methods have an even larger impact on this dataset, probably because there are many more relevant and irrelevant paragraphs for each question, making paragraph selection more important.

Note the naively trained model starts to lose performance as more paragraphs are used, showing that errors are being caused by the model being overly confident in incorrect extractions. We achieve a score of 61.55 EM and 67.61 F1 on the dev set. This advances the only prior result reported for this dataset, 50.6 EM and 57.3 F1 from Wang et al. (2017b), by 10 points.

5.3 SQuAD

We additionally evaluate our model on SQuAD. SQuAD questions were not built to be answered independently of their context paragraph, which makes it unclear how effective of an evaluation tool they can be for document-level question answering. To assess this we manually label 500 random questions from the training set.

We categorize questions as:

1. Context-independent, meaning it can be understood independently of the paragraph.
2. Document-dependent, meaning it can be understood given the article's title. For example, "What individual is the school named after?" for the document "Harvard University".
3. Paragraph-dependent, meaning it can only be understood given its paragraph. For example, "What was the first step in the reforms?".

We find 67.4% of the questions to be context-independent, 22.6% to be document-dependent, and the remaining 10% to be paragraph-dependent. There are many document-dependent questions because questions are frequently about the subject of the document. Since a reasonably high fraction of the questions can be understood given the document they are from, and to isolate our analysis from the retrieval mechanism used, we choose to evaluate on the document-level. We build documents by concatenating all the paragraphs in SQuAD from the same article together into a single document.

Given the correct paragraph (i.e., in the standard SQuAD setting) our model reaches 72.14 EM and 81.05 F1 and can complete 26 epochs of training in less than five hours. Most of our variations to handle the multi-paragraph setting caused a minor (up to half a point) drop in performance, while the sigmoid version fell behind by a point and a half.

We graph the document-level performance in Figure 5. For SQuAD, we find it crucial to employ one of the suggested confidence training techniques. The base model starts to drop in performance once more than two paragraphs are used. However, the shared-norm approach is able to reach a peak performance of 72.37 F1 and 64.08 EM given 15 paragraphs. Given our estimate that 10% of the questions are ambiguous if the paragraph is unknown, our approach appears to have adapted to the document-level task very well.

Finally, we compare the shared-norm model with the document-level result reported by Chen et al. (2017a). We re-evaluate our model using the documents used by Chen et al. (2017a), which consist of the same Wikipedia articles SQuAD was built from, but downloaded at different dates. The advantage of this dataset is that it does not allow the model to know *a priori* which paragraphs were filtered out during the construction of SQuAD. The disadvantage is that some of the articles have been edited since the questions were written, so some questions may no longer be answerable. Our model achieves 59.14 EM and 67.34 F1 on this dataset, which significantly outperforms the 49.7 EM reported by Chen et al. (2017a).

5.4 Curated TREC

We perform one final experiment that tests our model as part of an end-to-end question answering system. For document retrieval, we re-implement the pipeline from Joshi et al. (2017). Given a question, we retrieve up to 10 web documents us-

[7]https://github.com/brmson/yodaqa/wiki/Benchmarks

Model	Accuracy
S-Norm (ours)	53.31
YodaQA with Bing (Baudiš, 2015),	37.18
YodaQA (Baudiš, 2015),	34.26
DrQA + DS (Chen et al., 2017a)	25.7

Table 4: Results on the Curated TREC corpus, YodaQA results extracted from its github page[7]

ing a Bing web search of the question, and all Wikipedia articles about entities the entity linker TAGME (Ferragina and Scaiella, 2010) identifies in the question. We then use our linear paragraph ranker to select the 16 most relevant paragraphs from all these documents, which are passed to our model to locate the final answer span. We choose to use the shared-norm model trained on the TriviaQA unfiltered dataset since it is trained using multiple web documents as input. We use the same heuristics as Joshi et al. (2017) to filter out trivia or QA websites to ensure questions cannot be trivially answered using webpages that directly address the question. A demo of the system is publicly available[8].

We find accuracy on the TriviaQA unfiltered questions remains almost unchanged (within half a percent exact match score) when using our document retrieval method instead of the given documents, showing our pipeline does a good job of producing evidence documents that are similar to the ones in the training data.

We test the system on questions from the TREC QA tasks (Voorhees et al., 1999), in particular a curated set of questions from Baudiš (2015), the same dataset used in Chen et al. (2017a). We apply our system to the 694 test questions without retraining on the train questions.

We compare against DrQA (Chen et al., 2017a) and YodaQA (Baudiš, 2015). It is important to note that these systems use different document corpora (Wikipedia for DrQA, and Wikipedia, several knowledge bases, and optionally Bing web search for YodaQA) and different training data (SQuAD and the TREC training questions for DrQA, and TREC only for YodaQA), so we cannot make assertions about the relative performance of individual components. Nevertheless, it is instructive to show how the methods we experiment with in this work can advance an end-to-end QA system.

The results are listed in Table 4. Our method outperforms prior work, breaking the 50% accuracy mark. This is a strong proof-of-concept that neural paragraph reading combined with existing document retrieval methods can advance the state-of-the-art on general question answering. It also shows that, despite the noise, the data from TriviaQA is sufficient to train models that can be effective on out-of-domain QA tasks.

[8]https://documentqa.allenai.org/

Category	proportion
Sentence reading errors	35.2
Paragraph reading errors	17.6
Document coreference errors	14.1
Part of answer extracted	7.1
Required background knowledge	5.8
Answer indirectly stated	20.2

Table 5: Error analysis on TriviaQA web.

5.5 Discussion

We found that models that have only been trained on answer-containing paragraphs can perform very poorly in the multi-paragraph setting. The results were particularly bad for SQuAD; we think this is partly because the paragraphs are shorter, so the model had less exposure to irrelevant text.

The shared-norm approach consistently outperformed the other methods, especially on SQuAD and TriviaQA unfiltered, where many paragraphs were needed to reach peak performance. Figures 3, 4, and 5 show this technique has a minimal effect on the performance when only one paragraph is used, suggesting the model's per-paragraph performance is preserved. Meanwhile, it can be seen the accuracy of the shared-norm model never drops as more paragraphs are added, showing it successfully resolves the problem of being distracted by irrelevant text.

The no-answer and merge approaches were moderately effective, we suspect because they at least expose the model to more irrelevant text. However, these methods do not address the fundamental issue of requiring confidence scores to be comparable between independent applications of the model to different paragraphs, which is why we think they lagged behind. The sigmoid objective function reduces the paragraph-level performance considerably, especially on the TriviaQA datasets. We suspect this is because it is vulnerable to label noise, as discussed in Section 2.2.

5.6 Error Analysis

We perform an error analysis by labeling 200 random TriviaQA web dev-set errors made by the shared-norm model. We found 40.5% of the errors were caused because the document did not contain sufficient evidence to answer the question, and 17% were caused by the correct answer not being contained in the answer key. The distribution of the remaining errors is shown in Table 5.

We found quite a few cases where a sentence contained the answer, but the model was unable to extract it due to complex syntactic structure or paraphrasing. Two kinds of multi-sentence reading errors were also common: cases that required connecting multiple statements made in a single paragraph, and long-range coreference cases where a sentence's subject was named in a previous paragraph. Finally, some questions required background knowledge, or required the model to extract answers that were only stated indirectly (e.g., examining a list to extract the nth element). Overall, these results suggest good avenues for improvement are to continue advancing the sentence and paragraph level reading comprehension abilities of the model, and adding a mechanism to handle document-level coreferences.

6 Related Work

Reading Comprehension Datasets. The state of the art in reading comprehension has been rapidly advanced by neural models, in no small part due to the introduction of many large datasets. The first large scale datasets for training neural reading comprehension models used a Cloze-style task, where systems must predict a held out word from a piece of text (Hermann et al., 2015; Hill et al., 2015). Additional datasets including SQuAD (Rajpurkar et al., 2016), WikiReading (Hewlett et al., 2016), MS Marco (Nguyen et al., 2016) and TriviaQA (Joshi et al., 2017) provided more realistic questions. Another dataset of trivia questions, Quasar-T (Dhingra et al., 2017), was introduced recently that uses ClueWeb09 (Callan et al., 2009) as its source for documents. In this work we choose to focus on SQuAD because it is well studied, and TriviaQA because it is more challenging and features documents and multi-document contexts (Quasar T is similar, but was released after we started work on this project).

Neural Reading Comprehension. Neural reading comprehension systems typically use some form of attention (Wang and Jiang, 2016), although alternative architectures exist (Chen et al., 2017a; Weissenborn et al., 2017b). Our model follows this approach, but includes some recent advances such as variational dropout (Gal and Ghahramani, 2016) and bi-directional attention (Seo et al., 2016). Self-attention has been used in several prior works (Cheng et al., 2016; Wang et al., 2017c; Pan et al., 2017). Our approach to allowing a reading comprehension model to produce a per-paragraph no-answer score is related to the approach used in the BiDAF-T (Min et al., 2017) model to produce per-sentence classification scores, although we use an attention-based method instead of max-pooling.

Open QA. Open question answering has been the subject of much research, especially spurred by the TREC question answering track (Voorhees et al., 1999). Knowledge bases can be used, such as in (Berant et al., 2013), although the resulting systems are limited by the quality of the knowledge base. Systems that try to answer questions using natural language resources such as YodaQA (Baudiš, 2015) typically use pipelined methods to retrieve related text, build answer candidates, and pick a final output.

Neural Open QA. Open question answering with neural models was considered by Chen et al. (2017a), where researchers trained a model on SQuAD and combined it with a retrieval engine for Wikipedia articles. Our work differs because we focus on explicitly addressing the problem of applying the model to multiple paragraphs. A pipelined approach to QA was recently proposed by Wang et al. (2017a), where a ranker model is used to select a paragraph for the reading comprehension model to process. More recent work has considered evidence aggregation techniques (Wang et al., 2017b; Swayamdipta et al., 2017). Our work shows paragraph-level models that produce well-calibrated confidence scores can effectively exploit large amounts of text without aggregation, although integrating aggregation techniques could further improve our results.

7 Conclusion

We have shown that, when using a paragraph-level QA model across multiple paragraphs, our training method of sampling non-answer-containing paragraphs while using a shared-norm objective function can be very beneficial. Combining this with our suggestions for paragraph selection, using the summed training objective, and our model design allows us to advance the state of the art on TriviaQA. As shown by our demo, this work can be directly applied to building deep-learning-powered open question answering systems.

References

Petr Baudiš. 2015. YodaQA: A Modular Question Answering System Pipeline. In *POSTER 2015-19th International Student Conference on Electrical Engineering*.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic Parsing on Freebase from Question-Answer Pairs. In *EMNLP*.

Jamie Callan, Mark Hoy, Changkuk Yoo, and Le Zhao. 2009. Clueweb09 Data Set.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017a. Reading Wikipedia to Answer Open-Domain Questions. *arXiv preprint arXiv:1704.00051*.

Zheqian Chen, Rongqin Yang, Bin Cao, Zhou Zhao, Deng Cai, and Xiaofei He. 2017b. Smarnet: Teaching Machines to Read and Comprehend Like Human. *arXiv preprint arXiv:1710.02772*.

Jianpeng Cheng, Li Dong, and Mirella Lapata. 2016. Long Short-Term Memory-Networks for Machine Reading. *arXiv preprint arXiv:1601.06733*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv preprint arXiv:1406.1078*.

Bhuwan Dhingra, Kathryn Mazaitis, and William W Cohen. 2017. Quasar: Datasets for Question Answering by Search and Reading. *arXiv preprint arXiv:1707.03904*.

Paolo Ferragina and Ugo Scaiella. 2010. TAGME: On-the-fly Annotation of Short Text Fragments (by Wikipedia Entities). In *Proceedings of the 19th ACM international conference on Information and knowledge management*.

Yarin Gal and Zoubin Ghahramani. 2016. A Theoretically Grounded Application of Dropout in Recurrent Neural Networks. In *Advances in neural information processing systems*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching Machines to Read and Comprehend. In *Advances in Neural Information Processing Systems*.

Daniel Hewlett, Alexandre Lacoste, Llion Jones, Illia Polosukhin, Andrew Fandrianto, Jay Han, Matthew Kelcey, and David Berthelot. 2016. Wikireading: A Novel Large-scale Language Understanding Task over Wikipedia. *arXiv preprint arXiv:1608.03542*.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The Goldilocks Principle: Reading Children's Books with Explicit Memory Representations. *arXiv preprint arXiv:1511.02301*.

Minghao Hu, Yuxing Peng, and Xipeng Qiu. 2017. Mnemonic Reader: Machine Comprehension with Iterative Aligning and Multi-hop Answer Pointing.

Robin Jia and Percy Liang. 2017. Adversarial Examples for Evaluating Reading Comprehension Systems. *arXiv preprint arXiv:1707.07328*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. *arXiv preprint arXiv:1705.03551*.

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text Understanding with the Attention Sum Reader Network. *arXiv preprint arXiv:1603.01547*.

Sewon Min, Minjoon Seo, and Hannaneh Hajishirzi. 2017. Question Answering through Transfer Learning from Large Fine-grained Supervision Data. *arXiv preprint arXiv:1702.02171*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A Human Generated MAchine Reading COmprehension Dataset. *arXiv preprint arXiv:1611.09268*.

Boyuan Pan, Hao Li, Zhou Zhao, Bin Cao, Deng Cai, and Xiaofei He. 2017. MEMEN: Multi-layer Embedding with Memory Networks for Machine Comprehension. *arXiv preprint arXiv:1707.09098*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. *arXiv preprint arXiv:1606.05250*.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional Attention Flow for Machine Comprehension. *CoRR*, abs/1611.01603.

Swabha Swayamdipta, Ankur P. Parikh, and Tom Kwiatkowski. 2017. Multi-Mention Learning for Reading Comprehension with Neural Cascades.

Chuanqi Tan, Furu Wei, Nan Yang, Weifeng Lv, and Ming Zhou. 2017. S-Net: From Answer Extraction to Answer Generation for Machine Reading Comprehension. *arXiv preprint arXiv:1706.04815*.

Ellen M Voorhees et al. 1999. The TREC-8 Question Answering Track Report. In *Trec*.

Shuohang Wang and Jing Jiang. 2016. Machine Comprehension Using Match-LSTM and Answer Pointer. *arXiv preprint arXiv:1608.07905*.

Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. 2017a. R: Reinforced Reader-Ranker for Open-Domain Question Answering. *arXiv preprint arXiv:1709.00023*.

Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. 2017b. Evidence Aggregation for Answer Re-Ranking in Open-Domain Question Answering.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017c. Gated Self-Matching Networks for Reading Comprehension and Question Answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1.

Dirk Weissenborn, Tomas Kocisky, and Chris Dyer. 2017a. Dynamic Integration of Background Knowledge in Neural NLU Systems. *arXiv preprint arXiv:1706.02596*.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017b. FastQA: A Simple and Efficient Neural Architecture for Question Answering. *arXiv preprint arXiv:1703.04816*.

Matthew D Zeiler. 2012. ADADELTA: an Adaptive Learning Rate Method. *arXiv preprint arXiv:1212.5701*.