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

arXiv:1710.10723v2 [cs.CL] 7 Nov 2017

Abstract

We consider the problem of adapting neural paragraph-level question answering models to the case where entire documents are given as input. Our proposed solution trains models to produce well calibrated confidence scores for their results on individual paragraphs. We sample multiple paragraphs from the documents during training, and use a shared-normalization training objective that encourages the model to produce globally correct output. We combine this method with a state-of-the-art pipeline for training models on document QA data. Experiments demonstrate strong performance on several document QA datasets. Overall, we are able to achieve a score of 71.3 F1 on the web portion of TriviaQA, a large improvement from the 56.7 F1 of the previous best system.

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., 2017b; Tan et al., 2017) suggests neural models have the potential to be a key part of a solution to this problem. Training and testing neural models that take entire documents as input is extremely computationally expensive, 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 returns the answer with the highest confidence (Chen et al., 2017). 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 propose a TF-IDF heuristic to select which paragraphs to train and test on. Since annotating entire documents is very 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).

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

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

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 web (Joshi et al., 2017), a dataset of questions paired with web documents that contain the answer. We achieve 71.3 F1 on the test set, a 15 point absolute gain over prior work. We additionally perform an ablation study on our pipelined method, and we show the effectiveness of our multi-paragraph methods on TriviaQA unfiltered and a modified version of SQuAD (Rajpurkar et al., 2016) where only the correct document, not the correct paragraph, is known. We also build a demonstration of our method by combining our model with a re-implementation of the retrieval mechanism used in TriviaQA to build a prototype end-to-end general question answering system [1]. We release our code [2] to facilitate future work in this field.

2 Pipelined Method

In this section we propose an approach to training pipelined question answering systems, where a single paragraph is heuristically extracted from the context document(s) and passed to a paragraph-level QA model. We suggest using a TF-IDF based paragraph selection method and argue that a summed objective function should be used to handle noisy supervision. We also propose a refined model that incorporates some recent modeling ideas for reading comprehension systems.

2.1 Paragraph Selection

Our paragraph selection method chooses the paragraph that has the smallest TF-IDF cosine distance with the question. Document frequencies are computed using just the paragraphs within the relevant documents, not the entire corpus. The advantage of this approach is that if a question word is prevalent in the context, for example if the word "tiger" is prevalent in the document(s) for the question "What is the largest living subspecies of the tiger?", greater weight will be given to question words that are less common, such as "largest" or "sub-species". Relative to selecting the first paragraph in the document, this improves the chance of the selected paragraph containing the correct answer from 83.1% to 85.1% on TriviaQA web. We also expect this approach to do a better job of selecting paragraphs that relate to the question since it is explicitly selecting paragraphs that contain question words.

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 defied orders and refused to evacuate...

Figure 1: Noisy supervision causes many spans of text that contain the answer, but are not situated in a context that relates to the question, to be labelled as correct answer spans (highlighted in red). This risks distracting the model from learning from more relevant spans (highlighted in 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 sum of the probabilities of all answer spans. 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. Thus the objective for the span start boundaries becomes:

$$-\log\left(\frac{\sum_{k\in A} e^{s_k}}{\sum_{i=1}^{n} e^{s_i}}\right)$$

where A is the set of tokens that start an answer span, n is the number of context tokens, and s_i is a scalar score computed by the model for span i. This optimizes the negative log-likelihood of selecting any correct start token. This objective is agnostic to how the model distributes probability

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

mass across the possible answer spans, thus the model can "choose" to focus on only the more relevant spans.

2.3 Model

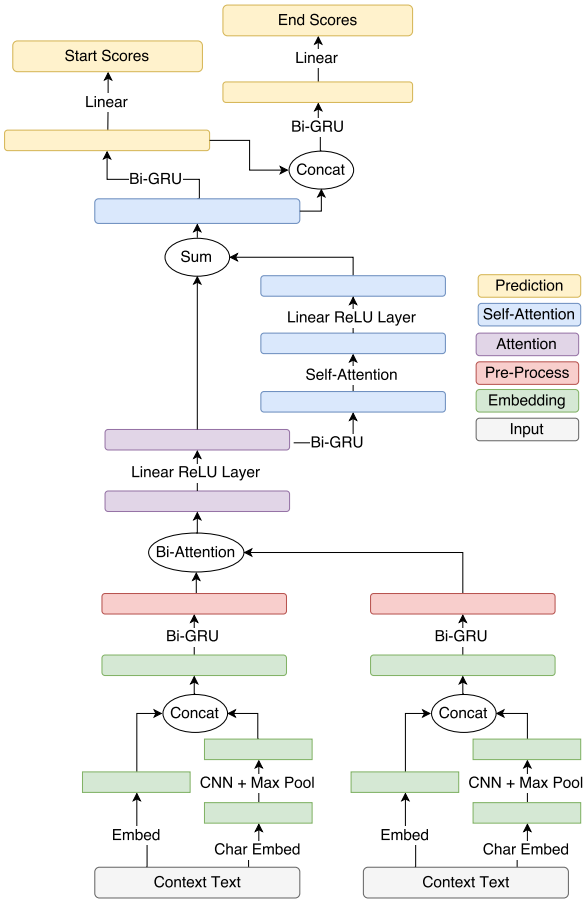

Figure 2: High level outline of our model.

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

Embedding: We embed words using pre-trained word vectors. We also embed the characters in each word into size 20 vectors which are learned, and run a convolution neural network followed by max-pooling to get character-derived embeddings for each word. The character-level and word-level embeddings are then concatenated and passed to the next layer. We do not update the word embeddings during training.

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

Attention: The bi-directional attention mechanism from the Bi-Directional Attention Flow (BiDAF) model (Seo et al., 2016) is used to build a query-aware context representation. Let $\mathbf{h_i}$ be the vector for context word i, $\mathbf{q_j}$ be the vector for 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:

$$p_{ij} = \frac{e^{a_{ij}}}{\sum_{j=1}^{n_q} e^{a_{ij}}}$$

$$\mathbf{c_i} = \sum_{j=1}^{n_q} \mathbf{q_j} p_{ij}$$

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

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

$$p_i = \frac{e^{m_i}}{\sum_{i=1}^{n_c} e^{m_i}}$$

$$\mathbf{q_c} = \sum_{i=1}^{n_c} \mathbf{h_i} p_i$$

The final vector computed 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. This layer is applied residually, so this output is additionally summed with the input.

Prediction: In the last layer of our model a bi-directional GRU is applied, followed by a linear layer that computes answer start scores for each token. The hidden states of that layer are concatenated with the input and fed into a second bi-directional GRU and linear layer to predict answer end scores. The softmax operation is applied to the start and end scores to produce start and end probabilities, and we optimize the negative log-likelihood of selecting correct start and end tokens.

Dropout: We also employ variational dropout, where a randomly selected set of hidden units

are set to zero across all time steps during training (Gal and Ghahramani, 2016). We dropout the input to all the GRUs, including the word embeddings, as well as 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. (2017).

Applying this approach without altering how the model is trained is, however, a gamble; the training objective does not require these confidence scores to be comparable between paragraphs. Our experiments in Section 5 show that in practice these models can be very poor at providing good confidence scores. Table 1 shows some qualitative examples of this phenomenon.

We hypothesize that there are two key reasons a model's confidence scores might not be well calibrated. 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 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, in Table 1 we observe that the model will assign high confidence values to spans that strongly match the category of the answer, even if the question words do not match the context. This might work passably well if an answer is present, but can lead to highly over-confident extractions in other cases. Similar kinds of errors have 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 follow 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 all paragraphs are processed independently as usual. However, a modified objective function is used where the normalization factor in the softmax operation is shared between all paragraphs from the same context. Therefore, the probability that token a from paragraph p starts an answer span is computed as:

$$\frac{e^{s_{ap}}}{\sum_{j \in P} \sum_{i=1}^{n_j} e^{s_{ij}}}$$

where P is the set of paragraphs that are from the same context as p, and s_{ij} is the score given to token i from paragraph j. We train on this objective by including multiple paragraphs from the same context in each mini-batch.

This is similar to simply feeding the model multiple paragraphs from each context concatenated together, except that each paragraph is processed independently until the normalization step. 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 the other paragraphs 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. Our motive is to test whether simply exposing the model to more text will teach the model to be more adept at ignoring irrelevant text.

3.3 No-Answer Option

We also experiment with allowing the model to select a special "no-answer" option for each paragraph. First, note that the independent-bounds objective can be re-written 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

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?	...plant species is the highest on Earth with 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)
What do the auricles do?	...many species of lobates have four auricles, gelatinous projections edged with cilia that **produce water currents that help direct microscopic prey toward the mouth**...	The Cestida are ribbon - shaped planktonic animals, with the mouth and aboral **organ aligned in the middle of opposite edges of the ribbon**

Table 1: Examples from SQuAD where a paragraph-level 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, the model still 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 the correct answer span.

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 compute a soft attention over the span start scores, $p_i = \frac{e^{s_i}}{\sum_{j=1}^{n} e^{s_j}}$, and then take the weighted sum of the hidden states from the GRU used to generate those scores, $\mathbf{h_i}$, giving $\mathbf{v_1} = \sum_{i=1}^{n} \mathbf{h_i} p_i$. We compute a second vector, $\mathbf{v_2}$ in the same way using the end scores. Finally, a step of learned attention is performed on the output of the Self-Attention layer that computes:

$$a_i = \mathbf{w} \cdot \mathbf{h_i}$$
$$p_i = \frac{e^{a_i}}{\sum_{j=1}^{n} e^{a_j}}$$
$$\mathbf{v_3} = \sum_{i=1}^{n} \mathbf{h_i} p_i$$

where $\mathbf{w}$ is a learned weight vector and $\mathbf{h_i}$ is the vector for token i.

We concatenate these three vectors and use them as input to 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 in the context 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 will be comparable between different paragraphs.

4 Experimental Setup

4.1 Datasets

We evaluate our approach on three 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 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 full 530k question-document training pairs. We also observed that the metrics for TriviaQA are computed after applying a small

amount of text normalization (stripping punctuation, removing articles, ect.) to both the ground truth text and the predicted text. As a result, some spans of text that would have been considered an exact match after normalization were not marked as answer spans during preprocessing, which only detected exact string matches. We fix this issue by labeling all spans of text that would have been considered an exact match by the official evaluation script as an answer span.

In TriviaQA, documents often contain many small paragraphs, so we merge paragraphs together as needed to get paragraphs of up to a target size. We use a maximum size of 400 unless stated otherwise. Paragraph separator tokens with learned embeddings are added between merged paragraphs to preserve formatting information.

4.3 Sampling

Our confidence-based approaches are all trained by sampling paragraphs, including paragraphs that do not contain an answer, during training. For SQuAD and TriviaQA web we take the top four paragraphs ranked by TF-IDF score for each question-document pair. We then sample two different paragraphs from this set each epoch. Since we observe that the higher-ranked paragraphs are much 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.

For TriviaQA unfiltered, where we have multiple documents for each question, we find it beneficial to use a more sophisticated paragraph ranking function. In particular, we use a linear function with five features: the TF-IDF cosine distance, whether the paragraph was the first in its document, how many tokens occur before it, and the number of case insensitive and case sensitive matches with question words. The function is trained on the distantly supervised objective of selecting paragraphs that contain at least one answer span. We select the top 16 paragraphs for each question and sample pairs of paragraphs as before.

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 find 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.

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. We significantly improve upon the baseline by combining the preprocessing procedures, TF-IDF paragraph selection, the sum objective, and our model design.

5 Results

5.1 TriviaQA Web

First, we do an ablation study on TriviaQA web to show the effects of our proposed methods for our pipeline model. We start with an implementation of the baseline from (Joshi et al., 2017). Their system selects paragraphs by taking the first 400 tokens of each document, uses BiDAF (Seo et al., 2016) as the paragraph model, and selects a random answer span from each paragraph each epoch to be used in BiDAF's cross entropy loss function during training. Paragraphs of size 800 are used at test time. As shown in Table 2, our implementation of this approach 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; even without changing the model we achieve state-of-the-art results. Using our refined model increases the gain by another 4 points.

Next we show the results of our confidence-based approaches. In this setting we group each document's text into paragraphs of at most 400 tokens and rank them using our TF-IDF heuristic. Then we measure the performance of our proposed

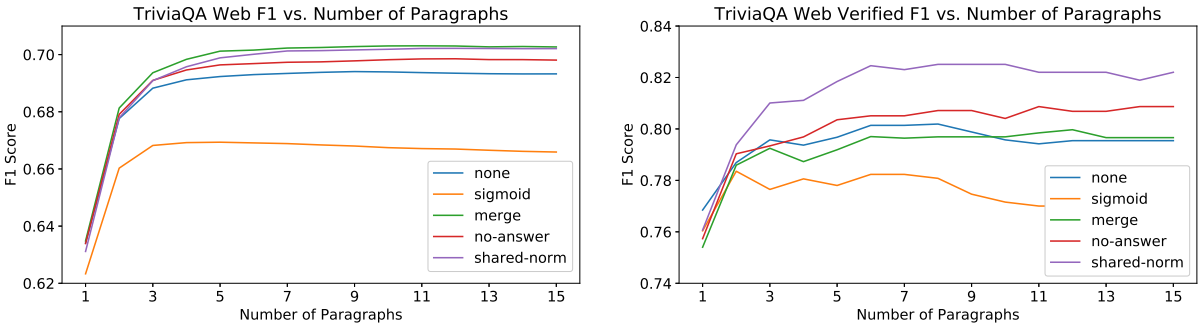

Figure 3: Results on TriviaQA web (left) and verified TriviaQA web (right) when applying our models to multiple paragraphs from each document. The shared-norm, merge, and no-answer training methods improve the model's ability to utilize more text, with the shared-norm method being significantly ahead of the others on the verified set and tied with the merge approach on the general set.

Model	All		Verified	
	EM	F1	EM	F1
baseline (Joshi et al., 2017)	40.74	47.06	49.54	55.80
MEMEN* (Pan et al., 2017)	43.16	46.90	49.28	55.83
Mnemonic Reader (Hu et al., 2017)	46.94	52.85	54.45	59.46
Reading Twice for NLU (Weissenborn et al., 2017a)	50.56	56.73	63.20	67.97
S-Norm (ours)	66.37	71.32	79.97	83.70

*Results on the dev set

Table 3: Published TriviaQA results. We advance the state of the art by about 15 points both test sets.

approaches as the model is used to independently process an increasing number of these paragraphs and the model's most confident answer is returned. We additionally measure performance on the verified portion of TriviaQA, a small subset of the question-document pairs in TriviaQA web where humans have manually verified that the document contains sufficient context to answer the question. The results are shown in Figure 3.

On these datasets even the model trained without any of the proposed training methods ("none") improves as it is allowed to use more text, 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 approach are even better. On the verified set, the shared-norm approach is solidly ahead of the other options. This suggests the shared-norm model is better at extracting answers when it is clearly stated in the text, but worse at guessing the answer in other cases.

We use the shared-norm approach for evaluation on the TriviaQA test set. We found that increasing the paragraph size to 800 at test time, and re-training the model on paragraphs of size 600, was slightly beneficial, allowing our model to reach 66.04 EM and 70.98 F1 on the dev set. We submitted this model to be evaluated on the TriviaQA test set and achieved 66.37 EM and 71.32 F1, firmly ahead of prior work, as shown in Table 3. Note that human annotators have estimated that only 75.4% of the question-document pairs contain sufficient evidence to answer the question (Joshi et al., 2017), which suggests we are approaching the upper bound for this task. However, the score of 83.7 F1 on the verified set suggests that there is still room for improvement.

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 attempting to answer a question using a document retrieval system. We show the same graph as before for this dataset in Figure 4. On this dataset it is more important to train the model to produce well calibrated confidence scores. Note the base 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.

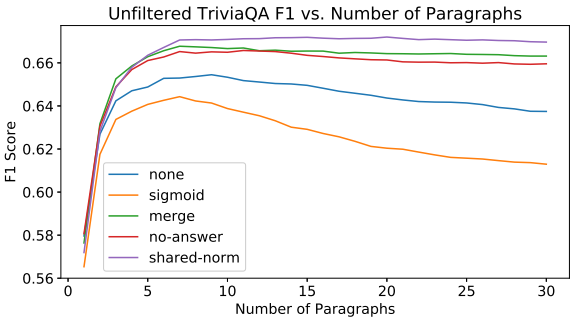

Figure 4: Results for our confidence methods on TriviaQA unfiltered. Here we see a more dramatic difference between these models. The shared-norm approach is the strongest, while the base model starts to lose performance as more paragraphs are used.

	Dev		Test	
Model	EM	F1	EM	F1
none	71.60	80.78	72.14	81.05
sigmoid	70.28	79.05	-	-
merge	71.20	80.26	-	-
no-answer	71.51	80.71	-	-
shared-norm	71.16	80.23	-	-

Table 4: Results on the standard SQuAD dataset. The test scores place our model as 8th on the SQuAD leader board among non-ensemble models[3]. Training with the proposed multi-paragraph approaches only leads to a marginal drop in performance in this setting.

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?".

[3]as of 10/23/2017

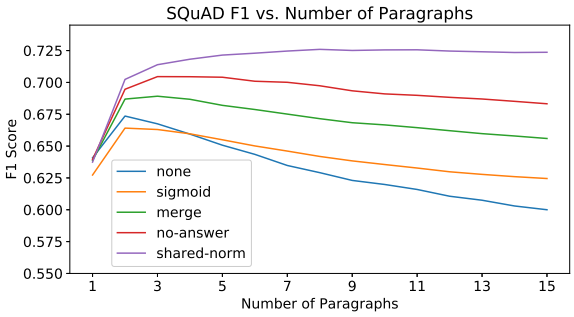

Figure 5: Results for our confidence methods on document-level SQuAD. The base model does poorly in this case, rapidly losing performance once more than two paragraphs are used. While all our approaches had some benefit, the shared-norm model is the strongest, and is the only one to not lose performance as large numbers of paragraphs are used.

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. The many document-dependent questions stem from the fact that questions are frequently about the subject of the document, so the article's title is often sufficient to resolve co-references or ambiguities that appear in the question. 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.

The performance of our models given the correct paragraph (i.e., in the standard SQuAD setting), is shown in Table 4. Our paragraph-level model is competitive on this task, and our variations to handle the multi-paragraph setting only cause a minor loss of performance.

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. (2017). We re-evaluate our model using the documents used by Chen et al. (2017), 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. (2017).

5.4 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. In general, we found the shared-norm approach to be the most effective way to resolve this problem. The no-answer and merge approaches were moderately effective, but we note that they do not resolve the scaling problem inherent to the softmax objective we discussed in Section 3, which might be why 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.

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 and TriviaQA.

Neural Reading Comprehension. Neural reading comprehension systems typically use some form of attention (Wang and Jiang, 2016), although alternative architectures exist (Chen et al., 2017; 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., 2017b; 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. (2017), 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.

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 by a large stride. 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*. pages 1156–1165.

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. 2017. Reading Wikipedia to Answer Open-Domain Questions. *arXiv preprint arXiv:1704.00051* .

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* .

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.

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* .

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017b. 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, pages 189–198.

Dirk Weissenborn, Tom Koisk, 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* .