MULTI-MENTION LEARNING FOR READING COMPREHENSION WITH NEURAL CASCADES

Anonymous authors
Paper under double-blind review

ABSTRACT

Reading comprehension is a challenging task, especially when executed across longer or across multiple evidence documents, where the answer is likely to reoccur. Existing neural architectures typically do not scale to the entire evidence, and hence, resort to selecting a single passage in the document (either via truncation or other means), and carefully searching for the answer within that passage. However, in some cases, this strategy can be suboptimal, since by focusing on a specific passage, it becomes difficult to leverage multiple mentions of the same answer throughout the document. In this work, we take a different approach by constructing lightweight models that are combined in a cascade to find the answer. Each submodel consists only of feed-forward networks equipped with an attention mechanism, making it trivially parallelizable. We show that our approach can scale to approximately an order of magnitude larger evidence documents and can aggregate information from multiple mentions of each answer candidate across the document. Empirically, our approach achieves state-of-the-art performance on both the Wikipedia and web domains of the TriviaQA dataset, outperforming more complex, recurrent architectures.

1 INTRODUCTION

Reading comprehension, the task of answering questions based on a set of one more documents, is a key challenge in natural language understanding. While data-driven approaches for the task date back to Hirschman et al. (1999), much of the recent progress can be attributed to new large-scale datasets such as the CNN/Daily Mail Corpus (Hermann et al., 2015), the Children's Book Test Corpus (Hill et al., 2015) and the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016). These datasets have driven a large body of neural approaches (Wang & Jiang, 2016; Lee et al., 2016; Seo et al., 2016; Xiong et al., 2016, *inter alia*) that build complex deep models typically driven by long short-term memory networks (Hochreiter & Schmidhuber, 1997). These models have given impressive results on SQuAD where the document consists of a single paragraph and the correct answer span is typically only present once. However, they are computationally intensive and cannot scale to large evidence texts such as those in the recently released TriviaQA (Joshi et al., 2017) dataset, which provides evidence as *entire* webpages.

So far, progress on the TriviaQA dataset has leveraged existing approaches on the SQuAD dataset by truncating documents and focusing on the first 800 words (Joshi et al., 2017; Pan et al., 2017). This has the obvious limitation that the truncated document may not contain the evidence required to answer the question. Furthermore, in TriviaQA there is often useful evidence spread through the supporting documents. This cannot be harnessed by approaches such as Choi et al. (2017) that greedily search for the best 1-2 sentences in a document. For example, in Fig.1 the answer does not appear in the first 800 words. The passage starting at token 4089 does contain all of the information required to infer the answer but this inference requires us to resolve the two complex co-referential phrases in 'In the summer of that year they got married in a church'. Both of these are easier to resolve if we have access to the other mentions of Kramer and Pollock that describe their relationship in the 1940's and the fact that they were married in a church.

In this paper we present a novel cascaded approach to extractive question answering that can accumulate evidence from an order of magnitude more text than previous approaches, and which achieves state-of-the-art performance on all tasks and metrics in the TriviaQA evaluation. The model (§3) is

Question : Which US artist married Lee Krasner in 1945?
Answer : Jackson Pollock
Document : Wikipedia entry for *Lee Krasner* (Excerpts shown)

Start	Passage
952	She lost interest in their usage of hard-edge geometric style after her relationship with **Pollock** began.
3364	**Pollock**'s influence.
3366	Although many people believe that Krasner stopped working in the 1940s in order to nurture **Pollock**'s home life and career, she never stopped creating art.
4084	Relationship with Jackson **Pollock**
4089	Lee Krasner and Jackson **Pollock** established a relationship with one another in 1942 after they both exhibited at the McMillen Gallery. Krasner was intrigued by his work and the fact she did not know who he was since she knew many abstract painters in New York. She went to his apartment to meet him. *By 1945, they moved to The Springs on the outskirts of East Hampton. In the summer of that year, they got married in a church with two witnesses present.*
4560	While she married **Pollock** in a church, Krasner continued to identify herself as Jewish but decided to not practice the religion.

Figure 1: Example from TriviaQA in which multiple mentions contain information that is useful in inferring the answer. Only the italicized phrase completely answers the question (Kramer could have married multiple times) but contains complex coreference that is beyond the scope of current natural language processing. The last phrase is more easy to interpret but it misses the clue provided by the year 1945.

split into three levels that consist of a feed-forward network applied to an embedding of the candidate answer span in the document. The first level submodels use simple bag-of-embeddings representations of the question, span, and the words surrounding the span in the document (the context). The second level submodel uses the representation built by the first level, along with an attention mechanism (Bahdanau et al., 2014) that aligns question words with words in the sentence that contains the candidate span. Finally, for answer candidates that are mentioned multiple times in the evidence document, the third level submodel aggregates the mention level representations from the second level to build a single answer representation. At inference time, predictions are made using the output of the third level classifier only. However, in training, as opposed to using a single loss, all the classifiers are trained using the the multi-loss framework of Al-Rfou et al. (2016), with gradients flowing from down from higher to lower submodels. The motivation behind the separation into submodels and multi-loss objective is the prevention of adaptation between features (Hinton et al., 2012). This is particularly important in our case where the higher level, more complex submodels could subsume the weaker, lower level models c.f. Al-Rfou et al. (2016).

In Section §4 we show that the extension to longer documents, the inclusion of the multi-mention aggregation, and the use of a multi-loss objective are all essential in helping our model achieve state of the art performance. Since we use only feed-forward networks along with fixed length window representations of the question, answer candidate, and answer context, the vast majority of computation required by our model is trivially parallelizable, in contrast to other efforts that use recurrent models.

2 Related Approaches

Most existing approaches to reading comprehension (Wang & Jiang, 2016; Lee et al., 2016; Seo et al., 2016; Xiong et al., 2016; Wang et al., 2017, *inter alia*) involve using recurrent neural nets (LSTMs or memory nets) along with various flavors of the attention mechanism Bahdanau et al.

(2014) to align the question with the passage. In preliminary experiments in the original TriviaQA paper, Joshi et al. (2017) explored one such approach, the BiDAF architecture (Seo et al., 2016), for their dataset. However, BiDAF being designed for SQuAD, where the evidence passage is much shorter (122 tokens on an average), does not scale to the entire document (2895 tokens on an average) for TriviaQA; hence the document is truncated to the first 800 tokens. Pointer networks with multi-hop reasoning, and syntactic and NER features, have been used recently in three architectures — Smarnet (Chen et al., 2017b), Reinforced Mnemonic Reader (Hu et al., 2017) and MEMEN (Pan et al., 2017) for both SQuAD and TriviaQA. Most of the above also use document truncation.

Approaches such as Choi et al. (2017) that first select the top (1-2) sentences using a very coarse model and then run a recurrent architecture on these sentences to find the correct span. Chen et al. (2017a) propose scoring spans in each paragraph with a recurrent network architecture separately and then take taking the span with the highest score.

Our approach is different from existing question-answering architectures in the following aspects. First, instead of using one monolithic architecture, we employ a cascade of simpler models that enables us to analyze larger parts of the document. Seondly, instead of recurrent neural networks, we use only feed-forward networks to improve scalability. Further, our approach aggregates information from different mentions of the same candidate answer at the representation level. Finally, our learning problem also leverages the presence of several correct answer spans in the document.

3 MODEL

For the reading comprehension task (§3.1), we propose a cascaded model architecture arranged in three different levels (§3.2). Submodels in the lower levels (§3.3) use simple features to score candidate answer spans. Higher level submodels select the best answer candidate using more expensive attention-based features (§3.4) and by aggregating information from different occurrences of each answer candidate in the document (§3.5). The submodels score all the potential answer span candidates in the evidence document[1], each represented using simple bags-of-embeddings. Each submodel is associated with its own objective; the final objective is given by a linear combination of all the objectives (§3.6). We conclude this section with a runtime analysis of the model (§3.7).

3.1 TASK

We take as input a sequence question word embeddings $\mathbf{q} = \{\mathbf{q_1} \dots \mathbf{q_m}\}$, and document word embeddings $\mathbf{d} = \{\mathbf{d_1} \dots \mathbf{d_n}\}$, obtained from a dictionary of pre-trained word embeddings.

Each candidate answer span $\mathbf{s} = \{\mathbf{d_{s_1}} \dots \mathbf{d_{s_o}}\}$ is a collection of $o \leq l$ word embeddings where l is the maximum length of a span. The set of all candidate answer spans is $\mathrm{S} := \{\mathbf{s_i}\}_{i=1}^{nl}$ [2].

Since the same spans of text can occur multiple times in each document, we also maintain the set of unique answer candidate spans, $\mathbf{u} \in \mathrm{S}_u$, and a mapping between each span and the unique answer candidate that it corresponds to, $\mathbf{s} \twoheadrightarrow \mathbf{u}$. In TriviaQA, each answer can have multiple alternative forms, as shown in Fig.1. The set of correct answer strings is S^* and our task is to predict a single answer candidate $\hat{\mathbf{u}} \in \mathrm{S}$.

3.2 OVERVIEW OF META-ARCHITECTURE

We first describe our meta-architecture, which is a collection of simple submodels $M_k(\cdot)$ organized in a cascade. The idea of modeling separate classifiers that use complementary features comes from Al-Rfou et al. (2016) who found this gave considerable gains over combining all the features into a single model for a conversational system. As shown in Figure 2, our architecture consists of two submodels M_1, M_2 at the first level, one submodel M_3 at the second, and one submodel M_4 at the third level. Each submodel M_k returns a score, $\phi_s^{(k)}$ as well as a vector, $\mathbf{h}_s^{(k)}$ that is fed as input

[1] We truncate extremely long documents (§4), but with a truncation $10\times$ higher than prior work.

[2] Limiting spans to length l minimally effects oracle accuracy and allows the approach to scale linearly with document size.

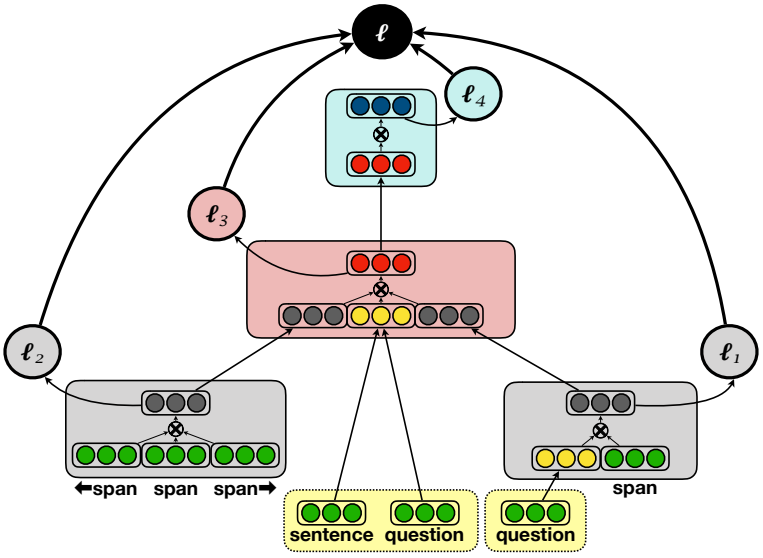

Figure 2: Cascaded model for reading comprehension. Input vectors are shown in green. Yellow rounded squares with dotted borders correspond to attention computation and attended vectors are shown in yellow. Submodels are shown in rounded squares with solid borders and the corresponding objectives are shown in color-coded circles. Level 1 submodels are in grey, level 2 in red and level 3 in blue. The **ffnn** operator is shown by the cross symbol within each submodel. The final objective, shown in the top black circle, is a linear interpolation of submodel objectives.

to the submodel at the next level.

$$\phi_s^{(1)}, \mathbf{h}_s^{(1)} := M_1(\mathbf{q}, \mathbf{d}, \mathbf{s}) \ \forall s \in \mathrm{S} \qquad \phi_s^{(2)}, \mathbf{h}_s^{(2)} := M_2(\mathbf{q}, \mathbf{d}, \mathbf{s}) \ \forall \mathbf{s} \in \mathrm{S} \qquad \textbf{LEVEL 1}$$

$$\phi_s^{(3)}, \mathbf{h}_s^{(3)} := M_3(\mathbf{q}, \mathbf{d}, \mathbf{s}, \mathbf{h}_s^{(1)}, \mathbf{h}_s^{(2)}) \ \forall \mathbf{s} \in \mathrm{S} \qquad \textbf{LEVEL 2}$$

$$\phi_u^{(4)}, \mathbf{h}_u^{(4)} := M_4(\mathbf{q}, \mathbf{d}, \mathbf{s}, \{\mathbf{h}_s^{(3)}\}_{\mathbf{s} \twoheadrightarrow \mathbf{u}}\}) \ \forall \mathbf{s} \in \mathrm{S}_u \qquad \textbf{LEVEL 3}$$

Using their respective scores, $\phi_s^{(k)}$, the models $M_1 ... M_3$ define a distribution over all spans, while M_4 uses $\phi_{\mathbf{u}}^{(4)}$ to define a distribution over all unique candidates, as follows:

$$p^{(k)}(\mathbf{s}|\mathbf{q}, \mathbf{d}) = \frac{\exp \phi_{\mathbf{s}}^{(k)}}{\sum_{\mathbf{s}' \in \mathbf{S}} \exp \phi_{\mathbf{s}'}^{(k)}} \quad k \in \{1, 2, 3\} \qquad p^{(4)}(\mathbf{u}|\mathbf{q}, \mathbf{d}) = \frac{\exp \phi_{\mathbf{u}}^{(4)}}{\sum_{\mathbf{u}' \in \mathbf{S}_u} \exp \phi_{\mathbf{u}'}^{(4)}} \tag{1}$$

In training, our total loss is given by an interpolation of losses for each of $M_1, .., M_4$. However, during inference we make a single prediction, simply computed as $\hat{u} = \arg\max_{u \in \mathbf{S}_u} \phi_{\mathbf{u}}^{(4)}$.

3.3 Level 1: Averaged Bags of Embeddings

The first submodel is the simplest, and it takes as input only bags of embeddings. This submodel has two parts, one that looks at the span and question together, and another that looks at the span along with the local context in which the span appears. First we define the span representation used by both of these parts. Then we will describe the question and context representations, and how they are used to predict which span is the correct answer.

Span: The length o span, $\mathbf{s}$ is represented using averaged embeddings $\tilde{\mathbf{s}} = \frac{1}{o} \sum_{j=1}^{o} \mathbf{d}_{\mathbf{s_j}}$.

We also include a binary feature γ_{qs_i} indicating if any question words are in the span, following (Chen et al., 2017a). The question-in-span feature is motivated by the observation that questions rarely contain the answers that they are seeking and it helps the model avoid answers that are over-complete — containing information already known by the questioner.

3.3.1 QUESTION + SPAN (M_1)

The question + span component of the level 1 submodel predicts the correct answer using a feed-forward neural network on top of fixed length question and span representations. The span representation is taken from above and we represent the question with a weighted average of the question embeddings.

Question: Motivated by Lee et al. (2016) we learn a weight δ_{q_i} for each of the words in the question using the parameterized function defined below. This allows the model to learn to focus on the important words in the question. δ_{q_i} is generated with a two-layer feed-forward net with rectified linear unit (ReLU) activations (Nair & Hinton, 2010; Glorot et al., 2011).

$$\begin{aligned}\mathbf{h}_{q_i} &= \mathbf{ReLU}\{\mathbf{U}\{\mathbf{ReLU}\{\mathbf{V}\mathbf{q_i} + \mathbf{a}\}\} + \mathbf{b}\} \\ &= \mathbf{ffnn}_q(\mathbf{q_i}) \\ \delta_{q_i} &= \mathbf{w}^T\mathbf{h}_{q_i} + \mathbf{z} \\ &= \mathbf{linear}_q(\mathbf{h_{q_i}})\end{aligned}$$

where $\mathbf{U}$, $\mathbf{V}$, $\mathbf{w}$, $\mathbf{z}$, $\mathbf{a}$ and $\mathbf{b}$ are parameters of the feed-forward network.

Since all three submodels rely on identically structured feed-forward networks and linear prediction layers, from now on we will use the abbreviations $\mathbf{ffnn}$ and $\mathbf{linear}$ as shorthand for the functions defined above.

The scores, δ_{q_i} are normalized and used to generate an aggregated question vector $\tilde{\mathbf{q}}$ as follows.

$$\tilde{\mathbf{q}} = \sum_{i=1}^{m} \frac{\exp \delta_{q_i} \mathbf{q_i}}{\sum_{j=1}^{m} \exp \delta_{q_j}}$$

Now that we have representations of both the question and the span, the question + span model computes a hidden layer representation of the question, span pair as well as a scalar score for the span candidate as follows:

$$\mathbf{h}_s^{(1)} = \mathbf{ffnn}_{qs}([\tilde{\mathbf{s}}; \tilde{\mathbf{q}}; \gamma_{qs}]), \quad \phi_{s_i}^{(1)} = \mathbf{linear}_{qs}(\mathbf{h}_s^{(1)})$$

where $[\mathbf{a}; \mathbf{b}]$ represents the concatenation of $\mathbf{a}$ and $\mathbf{b}$.

3.3.2 SPAN + SHORT CONTEXT (M_2)

The span + short context component builds a representation of each span in its local linguistic context. We hypothesize that words in the left and right context of a candidate span are important for the modeling of the span's semantic category (e.g. person, place, date).

Context: We represent the left context $\mathbf{c_s^L}$, and right context $\mathbf{c_s^R}$ using averaged embeddings

$$\mathbf{c_s^L} = \frac{1}{k}\sum_{j=1}^{k} \mathbf{d_{s_1-j}}, \quad \mathbf{c_s^R} = \frac{1}{k}\sum_{j=1}^{k} \mathbf{d_{s_o+j}}$$

and use these to generate a span-in-context representation $\mathbf{h}_s^{(2)}$ and answer score $\phi_s^{(2)}$

$$\mathbf{h}_s^{(2)} = \mathbf{ffnn}_c([\mathbf{s}; \mathbf{c_s^L}; \mathbf{c_s^R}; \gamma_{qs}]), \quad \phi_s^{(2)} = \mathbf{linear}_x(\mathbf{h}_s^{(2)}).$$

3.4 LEVEL 2: ATTENDING TO THE CONTEXT (M_3)

Unlike level 1 which builds a question representation independently of the document, the level 2 submodel considers the the question in the context of each sentence in the document. This level aligns the question embeddings with the embeddings in each sentence using neural attention (Bahdanau et al., 2014), specifically the attend and compare steps from the decomposable attention model of Parikh et al. (2016), and these aligned representations are used to predict if the sentence could contain the answers. We apply this attention to sentences, not individual span contexts, to prevent our computational requirements from exploding with the number of spans. Subsequently, it is only because level 2 includes the level 1 representations $\mathbf{h}_s^{(1)}$ and $\mathbf{h}_s^{(2)}$ that it can assign different scores to different spans in the same sentence.

Sentence: We define $\mu_{\mathbf{s}} = \{\mathbf{d}_{\mu_{\mathbf{s1}}} \dots \mathbf{d}_{\mu_{\mathbf{s}\ell}}\}$ to be the embeddings of the ℓ words of the sentence that contains $\mathbf{s}$. First, we measure the similarity between every pair of question and sentence embeddings by passing them through a feed-forward net, $\mathbf{ffnn}_{\text{att}_1}$ and using the resulting hidden layers to compute a similarity score, η. Taking into account this similarity, attended vectors $\bar{\mathbf{q}}_{\mathbf{i}}$ and $\bar{\mathbf{d}}_{\mu_{\mathbf{sj}}}$ are computed for each question and sentence token, respectively.

The original vector and its corresponding attended vector are concatenated and passed through another feed-forward net, $\mathbf{ffnn}_{\text{att}_2}$ the final layers from which are summed to obtain a question-aware sentence vector $\bar{\mu_{\mathbf{s}}}$, and a sentence context-aware question vector, $\bar{\mathbf{q}}$.

$$\eta_{ij} = \mathbf{ffnn}_{\text{att}_1}(\mathbf{q_i})^T \mathbf{ffnn}_{\text{att}_1}(\mathbf{d}_{\mu_{\mathbf{sj}}})$$

$$\bar{\mathbf{q}}_{\mathbf{i}} = \sum_{j=1}^{\ell} \frac{\exp \eta_{ij}}{\sum_{k=1}^{\ell} \exp \eta_{ik}} \mathbf{d}_{\mu_{\mathbf{sj}}} \qquad \bar{\mathbf{d}}_{\mu_{\mathbf{sj}}} = \sum_{i=1}^{m} \frac{\exp \eta_{ij}}{\sum_{k=1}^{m} \exp \eta_{kj}} \mathbf{q_i}$$

$$\bar{\mathbf{q}} = \sum_{i=1}^{m} \mathbf{ffnn}_{\text{att}_2}([\mathbf{q_i}; \bar{\mathbf{q}}_{\mathbf{i}}]) \qquad \bar{\mu_{\mathbf{s}}} = \sum_{j=1}^{\ell} \mathbf{ffnn}_{\text{att}_2}([\mathbf{d}_{\mu_{\mathbf{sj}}}; \bar{\mathbf{d}}_{\mu_{\mathbf{sj}}}])$$

Using these attended vectors as features, along with the hidden layers from level 1 and the question-span feature, new scores and hidden layers are computed for level 2:

$$\mathbf{h}_s^{(3)} = \mathbf{ffnn}_{L_2}([\mathbf{h}_s^{(1)}; \mathbf{h}_s^{(2)}; \bar{\mathbf{q}}; \bar{\mu_{\mathbf{s}}}; \gamma_{qs}]), \quad \phi_s^{(3)} = \mathbf{linear}_{L_2}(\mathbf{h}_s^{(3)})$$

3.5 Level 3: Aggregating Multiple Mentions (M_4)

In this level, we aggregate information from all the candidate answer spans which occur multiple times throughout the document. The hidden layers of every span from level 2 (along with the question-in-span feature) are passed through a feed-forward net, and then summed if they correspond to the same unique span, using the $s \twoheadrightarrow u$ map. The sum, $\mathbf{h}_u$ is then used to compute the score and the hidden layer[3] for each unique span, u in the document.

$$\mathbf{h}_u^{(4)} = \mathbf{ffnn}_{L_3}\Big(\sum_{s \in u} \mathbf{ffnn}_{\text{agg}}([\mathbf{h}_s^{(3)}; \gamma_{qs}])\Big), \quad \phi_s^{(4)} = \mathbf{linear}_{L_3}(\mathbf{h}_u^{(4)})$$

3.6 Overall objective and learning

To handle distant supervision, previous approaches use the first mention of the correct answer span (or any of its aliases) in the document as gold (Joshi et al., 2017). Instead, we leverage the existence of multiple correct answer occurrences by maximizing the probability of all such occurrences. Using Equation 1, the overall objective, $\ell(\mathbf{U}_*, \mathbf{V}_*, \mathbf{w}_*, \mathbf{z}_*, \mathbf{a}_*, \mathbf{b}_*)$ is given by the total negative log likelihood of the correct answer spans under all submodels:

$$-\sum_{k=1}^{3} \lambda_k \log \sum_{\hat{\mathbf{s}} \in S^*} p^{(k)}(\hat{\mathbf{s}}|\mathbf{q}, \mathbf{d}) - \lambda_4 \log \sum_{\hat{u} \in S^*} p^{(4)}(\hat{\mathbf{u}}|\mathbf{q}, \mathbf{d})$$

where $\lambda_1, .., .\lambda_4$ are hyperparameters that weigh the contribution of each loss term.

3.7 Computational Complexity

We briefly discuss the asymptotic complexity of our approach. For simplicity assume all hidden dimensions and the embedding dimension are ρ and that the complexity of matrix($\rho \times \rho$)-vector($\rho \times 1$) multiplication is $O(\rho^2)$. Thus, each application of a feed-forward network has $O(\rho^2)$ complexity. Recall that m is the length of the question, n is the length of the document, and l is the maximum length of each span. We then have the following complexity for each submodel:

[3]The hidden layer in level 3 is used only for computing the score $\phi_u^{(4)}$, mentioned here to preserve consistency of notation.

- **Level 1 (Question + Span)**: Building the weighted representation of each question takes $O(m\rho^2)$ and running the feed forward net to score all the spans requires $O(nld^2)$, for a total of $O(m\rho^2 + Onl\rho^2)$.
- **Level 1 (Span + Short Context)**: This requires $O(nl\rho^2)$.
- **Level 2**: Computing the alignment between the question and each sentence in the document takes $O(n\rho^2 + m\rho^2 + nm\rho)$ and then scoring each span requires $O(nl\rho^2)$. This gives a total complexity of $O(nl\rho^2 + nm\rho)$, since we can reasonably assume that $m < n$.
- **Level 3**: This requires $O(nl\rho^2)$.

Thus, the total complexity of our approach is $O(nl\rho^2 + mn\rho)$. While the nl and mn terms can seem expensive at first glance, a key advantage of our approach is that each sub-model is trivially parallelizable over the length of the document (n) and thus very amenable to GPUs.

4 Experiments and Results

4.1 TriviaQA Dataset

The TriviaQA dataset Joshi et al. (2017) contains a collection of 95k trivia question-answer pairs from several online trivia sources. To provide evidence for answering these questions, documents are collected independently, from the web and from Wikipedia. In addition to the answers from the trivia sources, aliases for the answers are collected from DBPedia; on an average, there are 16 such aliases per question-answer pair. Answers and their aliases can occur multiple times in the document; the average occurrence frequency is almost 15 times per document in either domain. The dataset also provides a subset on the development and test splits which only contain examples determined by annotators to have enough evidence in the document to support the answer.

4.2 Experimental Settings

Data preprocessing: All documents are tokenized using the NLTK[4] tokenizer. Each document is truncated to contain at most 6000 words and at most 1000 sentences. Sentences are truncated to a maximum length of 50. Spans only up to length $l = 5$ are considered; we discard cross-sentence spans. To be consistent with the evaluation setup of Joshi et al. (2017), for the Wikipedia domain we create a training instance for every question (with all its associated documents), while on the web domain we create a training instance for every question-document pair.

Hyperparameters: We use GloVe embeddings (Pennington et al., 2014) of dimension 300 (trained on a corpus of 840 billion words) that are kept fixed during training. Each embedding vector was normalized to have ℓ-2 norm of 1. Out-of-vocabulary words are hashed to one of 1000 random embeddings each initialized to zero mean and one variance. Dropout regularization (Srivastava et al., 2014) is applied to all ReLU layers (but not for the linear layers). We additionally tuned the following hyperparameters using grid search: network size (2-layers, each with 300 neurons), dropout ratio (0.1), learning rate (0.05), context size (1), and loss weights ($\lambda_1 = \lambda_2 = 0.35$, $\lambda_3 = 0.2$, $\lambda_4 = 0.1$) where the optimal values are indicated in parentheses. We use Adagrad (Duchi et al., 2011) for optimization with a default initial accumulator value of 0.1 and a batch size of 1. Each hyperparameter setting took 2-3 days to train on a single NVIDIA P100 GPU. The model was implemented in Tensorflow (Abadi et al., 2016).

4.3 Results

Table 1 presents our results on both the full test sets as well as the verified subsets, using the exact match (EM) and F_1 metrics. Our approach achieves state-of-the-art performance on both the Wikipedia and web domains outperforming considerably more complex models. In the Wikipedia domain, we surpass the leaderboard-best[5] (as of 24th October, 2017) on all metrics. In the web

[4] `http://www.nltk.org`

[5] Since these approaches might not be published as yet, it is not possible for us to draw comparisons with their models.

	Wikipedia				Web			
	Full		Verified		Full		Verified	
	EM	F_1	EM	F_1	EM	F_1	EM	F_1
BiDAF Seo et al. (2016); Joshi et al. (2017)	40.33	45.91	44.17	50.52	40.73	47.05	48.63	55.07
Smarnet Chen et al. (2017b)	42.41	48.84	50.51	55.90	40.87	47.09	51.11	55.98
Reinforced Mnemonic Reader Hu et al. (2017)	46.90	52.90	54.50	59.50	46.70	52.90	57.00	61.50
Leaderboard Best (10/24/2017)	48.64	55.13	53.42	59.92	50.56	56.73	**63.20**	**67.97**
Neural Cascades (Ours)	**51.59**	**55.95**	**58.90**	**62.53**	**53.75**	**58.57**	**63.20**	66.88

Table 1: TriviaQA results on the test set. EM stands for Exact Match accuracy.

domain, except for the verified F_1 scores, we see a similar trend. Surprisingly, we outperform approaches which use multi-layer recurrent pointer networks with specialized memories Chen et al. (2017b); Hu et al. (2017)[6].

	Wikipedia Dev (EM)
3-Level Cascade, Multi-loss (Ours)	**52.18**
3-Level Cascade, Single Loss	43.48
3-Level Cascade, Combined Level 1 **	**52.25**
Level 1 + Level 2 Only	46.52
Level 1 (Span + Context) Only	19.75
Level 1 (Question + Span) Only	15.75

Table 2: Model ablations on the full Wikipedia development set. For row labeled **, explanation provided in section §4.3.

Table 2 shows some ablations that give more insight into the different contributions of our model components. Our final approach (3-Level Cascade, Multi-loss) achieves the best performance. Training with only a single loss in Level 3 (3-Level Cascade, Single Loss) leads to a considerable decrease in performance, signifying the effect of using a multi-loss architecture. It is unclear if combining the two submodels in Level 1 into a single feed-forward network that is a function of the question, span and short context (3-Level Cascade, Combined Level 1) is significantly better than our final model. Although we found it could obtain high results, it was less consistent across different runs and gave lower scores on average (49.30) compared to our approach averaged over 4 runs (51.03). Finally, the last three rows show the results of using only smaller components of our model instead of the entire architecture. In particular, our model without the aggregation submodel (Level 1 + Level 2 Only) performs considerably worse, showing the value of aggregating multiple mentions of the same span across the document.

The effect of truncation on accuracy in Figure 4.3 (left) indicates that more scalable approaches that can take advantage of longer documents have the potential to perform better on the TriviaQA task. TriviaQA answers and their aliases typically reoccur in the document (15 times per document on an average). To verify whether our model is able to predict answers which occur frequently in the document, we look at the frequencies of the predicted answer spans in Fig 4.3 (right). This distribution follows the distribution of the gold answers very closely, showing that our model learns the frequency of occurrence of answer spans in the document.

5 Conclusion

We presented a 3-level cascaded model for TriviaQA reading comprehension. Our approach, through the use of feed-forward networks and bag-of-embeddings representations, can handle longer evidence documents and aggregated information from multiple occurrences of answer spans

[6]We cannot compare to MEMEN Pan et al. (2017) in this table since they provide only development set results; our Wikipedia dev EM is 52.18 compared to their 43.16

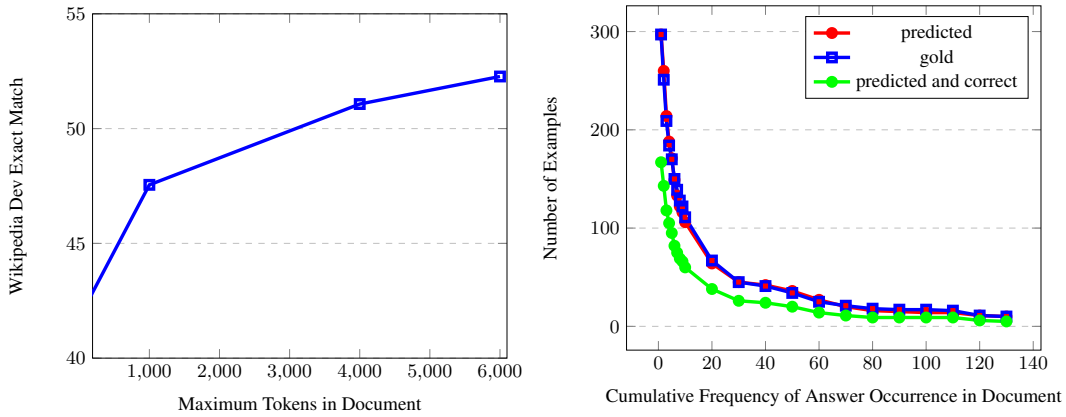

Figure 3: Analysis of model predictions. Left: Effect of truncation on performance. Right: Distribution of answer frequencies in the document, for model predictions (red), correct model predictions (green) and true answers (blue).

throughout the document. We achieved state-of-the-art performance on both Wikipedia and web domains, outperforming several complex recurrent architectures.

REFERENCES

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv:1603.04467*, 2016.

Rami Al-Rfou, Marc Pickett, Javier Snaider, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Conversational contextual cues: The case of personalization and history for response ranking, 2016. URL `http://arxiv.org/abs/1606.00372`. arXiv:1606.00372.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014. arXiv:1409.0473.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading Wikipedia to answer open-domain questions, 2017a. arXiv:1704.00051.

Zheqian Chen, Rongqin Yang, Bin Cao, Zhou Zhao, Deng Cai, and Xiaofei He. Smarnet: Teaching machines to read and comprehend like human, 2017b. arXiv:1710.02772.

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

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul), 2011.

Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proc. of AISTATS*, 2011.

Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend, 2015. URL `http://arxiv.org/abs/1506.03340`. arXiv:1506.03340.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. The Goldilocks principle: Reading children's books with explicit memory representations, 2015. URL `http://arxiv.org/abs/1511.02301`. arXiv:1511.02301.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Lynette Hirschman, Marc Light, Eric Breck, and John D Burger. Deep read: A reading comprehension system. In *Proc. of ACL*, 1999.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997.

Minghao Hu, Yuxing Peng, and Xipeng Qiu. Mnemonic reader for machine comprehension, 2017. URL http://arxiv.org/abs/1705.02798. arXiv:1705.02798.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proc. of ACL*, 2017.

Kenton Lee, Shimi Shalant, Tom Kwiatkowski, Ankur P. Parikh, Dipanjan Das, and Jonathan Berant. Learning recurrent span representations for extractive question answering, 2016. URL http://arxiv.org/abs/1611.01436. arXiv:1611.01436.

Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted Boltzmann machines. In *Proc. of ICML*, 2010.

Boyuan Pan, Hao Li, Zhou Zhao, Bin Cao, Deng Cai, and Xiaofei He. MEMEN: Multi-layer embedding with memory networks for machine comprehension, 2017. URL http://arxiv.org/abs/1707.09098. arXiv:1707.09098.

Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference, 2016. arXiV:1606.01933.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global vectors for word representation. In *Proc. of EMNLP*, 2014. URL http://www.aclweb.org/anthology/D14-1162.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text, 2016. arXiv:1606.05250.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension, 2016. URL http://arxiv.org/abs/1611.01603. arXiv:1611.01603.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer, 2016. arXiv:1608.07905.

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

Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering, 2016. arXiv:1611.01604.