Weaver: Deep Co-Encoding of Questions and Documents for Machine Reading

Martin Raison [1] **Pierre-Emmanuel Mazaré** [1] **Rajarshi Das** [2] **Antoine Bordes** [1]

Abstract

This paper aims at improving how machines can answer questions directly from text, with the focus of having models that can answer correctly multiple types of questions and from various types of texts, documents or even from large collections of them. To that end, we introduce the Weaver model that uses a new way to relate a question to a textual context by weaving layers of recurrent networks, with the goal of making as few assumptions as possible as to how the information from both question and context should be combined to form the answer. We show empirically on six datasets that Weaver performs well in multiple conditions. For instance, it produces solid results on the very popular SQuAD dataset (Rajpurkar et al., 2016), solves almost all bAbI tasks (Weston et al., 2015) and greatly outperforms state-of-the-art methods for open domain question answering from text (Chen et al., 2017).

1. Introduction

Being able to answer any question from large collections of unstructured text would give machines the ability to use as knowledge sources the huge amounts of varied information available in online encyclopedias (Wikipedia, etc.), but also in news articles, forums, blogs or social media posts. In theory, this unrestricted access to rich and dynamic information should lead to improved answering abilities. This comes, however, at the expense of having to solve a much harder problem than when using knowledge stored in structured databases. Here systems cannot rely on any ontology, and instead have to search large collections of documents and carefully read them to find the answers.

Chen et al. (2017) proposed the DrQA system to tackle this problem through machine reading at scale, that is, answering questions using spans of tokens extracted from Wikipedia. The DrQA pipeline system is composed of two modules: a document retriever and a document reader. The retriever is an information retrieval system based on TF-IDF matching that returns a subset of documents given a question. The reader is a model for the task of answering questions given a textual context, or *machine reading*, that uses bi-directional LSTMs (BiLSTMs) to get candidate answers from paragraphs and aggregate them afterwards.

Both modules perform quite well separately, but their combination experiences several limitations as illustrated by the following performance on the development set of the SQuAD dataset (Rajpurkar et al., 2016). When provided with the paragraph containing the answer, the reader can respond correctly to around 70% of the questions. When provided with the *entire* Wikipedia article containing the answer, however, this performance drops to 50%. This performance hit is compounded when integrated with the imperfect retriever. Considering the top 5 retrieved articles causes the performance of the reader to drop further to a final accuracy below 30% for DrQA. In this work, we focus on improving the reader to make it more general, more robust to longer contexts and hence more accurate overall.

Many recent advances in machine reading are done on the SQuAD dataset, with impressive results: the top performing methods on the leaderboard can compete with results of human subjects. However, the SQuAD dataset has its specificities, limitations and does not cover the whole range of question answering; there is a risk to overspecialize models for it. Recurrent neural networks and attention mechanisms are key components of those models. BiLSTMs are used for encoding questions and contexts into continuous representations, while attention is used for connecting them by building question-aware context representations and context-aware question representations. Self-attention is also used to give the model an opportunity to match different parts of the context with each other. Building such architectures requires multiple choices to decide how the various types of attention and recurrent layers should be combined together. For instance, the top published models on SQuAD[1] (Liu et al., 2017b; Huang et al., 2017; Xiong et al., 2017) all use at least four types of attention mechanisms in their models, whether through attention, co-attention or self-attention but have only been tested

[1]Facebook AI Research, Paris, France [2]University of Massachusetts, Amherst, USA. Correspondence to: Martin Raison <raison@fb.com>.

[1]as of February 2018.

on SQuAD or variants (Jia & Liang, 2017).

With the objective of being flexible to various types of questions, contexts and tasks, this paper introduces Weaver, a machine reading model that does not require any attention mechanism for building the representations for questions and contexts. Instead, Weaver relies on multiple layers of BiLSTMs that are woven to co-encode both questions and contexts simultaneously and hence learn to make both representations deeply interconnected, without having to specify how in the architecture. These representations are then used by an answering layer inspired by Memory Networks (Sukhbaatar et al., 2015) to produce the final response. We test Weaver on 6 different datasets for question answering including SQuAD and show that it performs remarkably well in various conditions.

In addition to strong results on SQuAD, Weaver is able to solve 17/18 bAbI tasks (Weston et al., 2015) that test different answering skills. It can even solve tasks that require to output words that do not belong to the context, something that limits most machine reading models. Weaver is also robust to finding responses in collections of short contexts as illustrated by its state-of-the-art results obtained on the QAngaroo WikiHop dataset (Welbl et al., 2017). As a result, its versatility and robustness make Weaver a very promising reader component for machine reading at scale, better suited than the original DrQA reader was. Indeed, while Weaver is around 5% better than DrQA for answering from paragraphs, when we use it in the pipeline for answering from the full Wikipedia, it boosts the overall performance by more than 15%.

2. Related Work

The machine reading task of learning to automatically answer questions given a provided piece of text (news article, fictional story, Wikipedia snippet, etc.) has been making great progress in recent years thanks to the creation of datasets like MCTest (Richardson et al., 2013), QACNN/DailyMail (Hermann et al., 2015), CBT (Hill et al., 2016), WikiQA (Yang et al., 2015), bAbI (Weston et al., 2015), SQuAD (Rajpurkar et al., 2016) or QAngaroo (Welbl et al., 2017) — as well as initiatives to group them together like ParlAI (Miller et al., 2017).

Most recent advances in machine reading expect that a supporting document is provided when a question is asked; they are not suited for the open-domain scenario in which one has to search through large databases to answer. Until recently, open-domain question answering had been mostly addressed through the task of answering from structured knowledge bases such as WikiData (Vrandečić & Krötzsch, 2014) or DBpedia (Auer et al., 2007). However, the limitations of knowledge bases (missing information, rigid schema, imperfect information, etc.) and the recent advances in machine reading have allowed new progress in the original trend of work of answering from large collections of unstructured text. Hence, new resources mixing questions with textual pieces of evidence returned by a search engine have been recently proposed. These include MSMARCO (Nguyen et al., 2016) where questions sampled from real anonymized user queries are paired with real web documents retrieved by the Bing search engine; SearchQA (Dunn et al., 2017) that mixes question-answer pairs from the J! Archive with text snippets retrieved by Google; and TriviQA (Joshi et al., 2017) that includes question-answer pairs authored by trivia enthusiasts along with independently gathered evidence documents. In this paper, we follow the setting used in (Chen et al., 2017; Wang et al., 2017a) to be comparable with the results therein and hence use open-domain versions of SQuAD, WebQuestions (Berant et al., 2013), WikiMovies (Miller et al., 2016) and CuratedTREC (Baudiš & Šedivỳ, 2015). Our method could be adapted to MSMARCO, TriviaQA or SearchQA, though this is left as future work.

Numerous neural models have been proposed to jointly encode questions and textual evidence for machine reading. Most of them follow the same general structure of first using recurrent architectures to encode questions and contexts separately and then using multiple types of attention mechanisms to connect them before running a final answering step. In this paper, we instead propose to co-encode questions and contexts with the same recurrent layers directly and use an answering step inspired by the multi-hops approach used in (Sukhbaatar et al., 2015; Liu et al., 2017b).

Most of the recent architectures have been developed and tested on a single or very few datasets, usually bAbI, QACNN and recently SQuAD, which leaves some questions regarding their capacity to adapt to multiple types of conditions. For instance, Hermann et al. (2015) and Kadlec et al. (2016) tested their models in the Cloze setting of QACNN/DailyMaill and CBT only. Sukhbaatar et al. (2015) and Seo et al. (2016) focused on solving the bAbI tasks. Recently, a large body of work have been proposed to tackle the SQuAD dataset only e.g. in (Dhingra et al., 2016; Wang et al., 2017b; Xiong et al., 2017; Liu et al., 2017a; Hu et al., 2017; Huang et al., 2017; Liu et al., 2017b). Besides, almost none of them has been applied to a setting requiring to answer from a large collections of documents. Some of the exceptions are the Reinforced Mnemonic Reader (Hu et al., 2017) that has been tested on SQuAD and TriviaQA or DrQA (Chen et al., 2017) and the Reinforced Reader-Ranker (Wang et al., 2017a) that both attempted to tackle the full pipeline problem combining retrieving relevant documents from Wikipedia and reading them to answer.

We show in our experiments in Section 5 that Weaver can perform well in various conditions from the bAbI tasks to SQuAD while also being applied in an open-domain setting where it outperforms by a vast margin both DrQA and the Reinforced Reader-Ranker. The latter method focuses on improving the connection between the retrieving and the reading modules through reinforcement learning but our results demonstrate major gains can already be obtained by improving the reading module alone.

3. The Weaver Architecture

Weaver is a model for machine reading that attempts to answer a question q by identifying a response span in a textual context c. The architecture of Weaver is composed of four parts: (i) embedding of the input words, (ii) deep co-encoding of q and c, (iii) a memory network step and (iv) a final answer prediction. An overall architecture diagram can be found in Fig. 1.

3.1. Input Embeddings

We first tokenize the question and context into individual words, and associate each word with a word embedding. The question and context are thus represented as sequences of respectively m and n word embeddings $\mathbf{q}_1, \mathbf{q}_2, ..., \mathbf{q}_m$ and $\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_n$. Unless otherwise noted, we use 300-dimensional FastText word embeddings trained on Common Crawl (Mikolov et al., 2017) and keep them fixed during training. Out-of-vocabulary words are represented with a fixed randomly initialized vector.

As in (Chen et al., 2017), we also associate each context word with an exact match feature and token features (part-of-speech, named entity recognition tags, term frequency). We denote these additional features $\mathbf{c}_j^{\text{extra}}$.

3.2. Question and Document Co-encoding

We consider a 2D map of size $m \times n$. For each coordinate (i, j) in the map, we combine the question and context feature vectors with a function

$$f : (\mathbf{q}_i, \mathbf{c}_j, \mathbf{c}_j^{extra}) \mapsto [\mathbf{q}_i; \mathbf{q}_i - \mathbf{c}_j; \mathbf{q}_i^\intercal \mathbf{c}_j; \mathbf{c}_j^{\text{extra}}] \quad (1)$$

where $[\cdot;\cdot]$ represents vector concatenation. This yields a 3D tensor M_0 of size $m \times n \times d$, where d is the dimensionality of the concatenated vector.

A simpler definition of f would be to just concatenate $\mathbf{q}_i$, $\mathbf{c}_j$ and $\mathbf{c}_j^{\text{extra}}$, but given that $\mathbf{q}_i$ and $\mathbf{c}_j$ live in the same vector space, using $\mathbf{q}_i - \mathbf{c}_j$ and $\mathbf{q}_i^\intercal \mathbf{c}_j$ explicitly makes it easier for the model to match tokens together. Note that f does not require any trainable parameters.

We then transform M_0 into a higher-level representation using recurrent layers. Since recurrent layers are typically one-dimensional, we propose to apply them alternatively along the question direction and the context direction. We call the sequence of one question-wise operation and one context-wise operation a *block*, and we propose to stack several blocks on top of each other.

More precisely, for the first block:

1. We first slice the input tensor M_0 in the context direction, giving us n slices of size $m \times d$. Considering each slice as a sequence of length m and input size d, we apply a single BiLSTM to each of those n slices. For each slice, the BiLSTM yields an output of size $m \times 2h$ (where h is the hidden layer size). We concatenate all those outputs back together to get a 3D tensor M_1 of size $m \times n \times 2h$.

2. Similarly, we then slice M_1 in the question dimension to get m slices of size $n \times 2h$. Considering each slice as a sequence of length n and input size $2h$, we apply another BiLSTM to each of those m slices. For each slice, the BiLSTM output is of size $n \times 2h$. We concatenate all those outputs back together to get a 3D tensor M_2 of size $m \times n \times 2h$.

We can stack such blocks by repeating these steps several times replacing d with $2h$ in the first step. The final output M_k has size $m \times n \times 2h$. All BiLSTMs used in this case have different sets of weights. We also add residual connections to all consecutive layers except the first one (which has different dimensionality), i.e. we add M_i to M_{i+1} before feeding to the next layer. Given the shape of the M_0 tensor, an approach using 2D convolutions at each layer would seem more intuitive, however we found the alternation of recurrent layers proposed here to perform significantly better on the development set of SQuAD.

From the M_k tensor we obtain the final representations for the question and the context in the following way. We compute two 3D tensors $U = \text{Conv}_{11}(M_k, \mathbf{W}^q)$ and $V = \text{Conv}_{11}(M_k, \mathbf{W}^v)$ as the result of convolutions of size 1 over M_k, where $\mathbf{W}^q$ and $\mathbf{W}^v$ are learned matrices of size $2h \times 2h$. These two different convolutions are important to distinguish questions and context features. We then derive a fixed-size representation for each token in the context by applying to V a max-pooling along the question dimension. This produces the *context representation*, a matrix $\mathbf{C}^h$ of size $n \times 2h$. Similarly, we compute a fixed-size representation of each token in the question by applying max-pooling along the context dimension to U, yielding the *question representation* a matrix $\mathbf{Q}^h$ of size $m \times 2h$.

The two representations $\mathbf{Q}^h$ and $\mathbf{C}^h$ are of variable length and maintain a sense of locality within the original context or question. For machine reading, we use the locality of information on the context representation to predict the

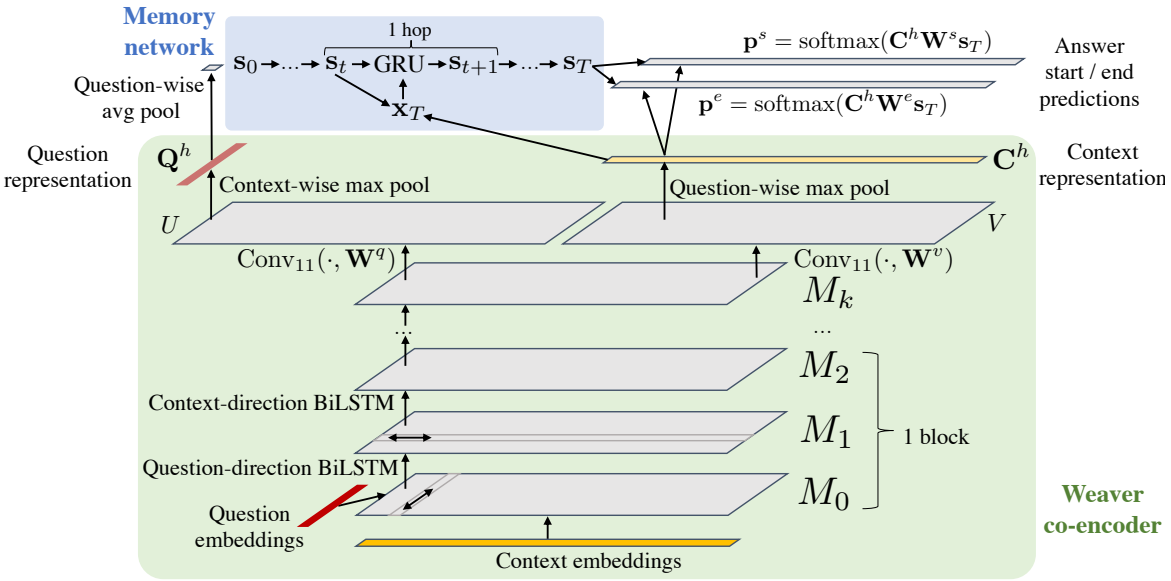

Figure 1. Weaver architecture. At the bottom of the figure, the Weaver co-encoder encodes the question and context into variable-length representations through a series of alternating BiLSTM layers. We reduce the question representation into a fixed size and use it as the initial state of our memory network (top left). At each hop, the memory network will update its state $\mathbf{s}_t$ based on the context representation $\mathbf{C}^h$. We then compute a bilinear projection of the final state of the memory network and the context representation to predict independently the start and end position of the answer span (top right).

likelihood for each token in the context to be the start or the end of the answer to the question.

In practice, recurrent layers are more efficiently applied to batches of sequences instead of isolated ones. By transposing and reshaping M_i appropriately at each layer, we can apply the BiLSTMs within blocks to all slices in parallel. This makes the model straightforward to implement.

3.3. Memory Network

We can directly use $\mathbf{Q}^h$ and $\mathbf{C}^h$ to predict the probability of each token to be the start or the end of an answer. However, multiple previous works have shown that it could be beneficial for the question to attend iteratively to the context before predicting an answer. The mutual conditioning of context and question representations is not sufficient to represent such an iterative process. We therefore add a multi-hop attention process before answering.

We apply an end-to-end memory network as introduced by Sukhbaatar et al. (2015). At each hop t, the memory network computes its updated state $\mathbf{s}_{t+1}$ depending on the context representation $\mathbf{C}^h$ and its previous state $\mathbf{s}_t$. We first apply an average-pooling to the question representation $\mathbf{Q}^h$ along the question dimension in order to reduce it to a single vector $\mathbf{s}_0$ of length $2h$ that is used as the initial state of the memory network. As proposed in (Liu et al., 2017b), each hop updates the state through a Gated Recurrent Unit (GRU) (Cho et al., 2014). We thus apply the following update rule for each hop t:

$$\mathbf{x}_t = \mathbf{C}^h\mathbf{W}^c\text{softmax}(\mathbf{C}^h\mathbf{W}^h\mathbf{s}_t)$$
$$\mathbf{s}_{t+1} = \text{GRU}(\mathbf{x}_t, \mathbf{s}_t)$$

with $\mathbf{W}^h$ and $\mathbf{W}^c$ learned matrices of size $2h \times 2h$.

The total number of hops T is an hyperparameter typically set between 1 and 5 in our experiments. Having 0 hop corresponds to bypassing the memory network entirely.

3.4. Answer Prediction

Unlike Liu et al. (2017b) that average all states $\mathbf{s}_t$ of the GRU to compute the probability of each answer span in the context, we only use the final state $\mathbf{s}_T$ to do so because we found it to be more accurate in practice. We define:

$$\mathbf{p}^s = \text{softmax}(\mathbf{C}^h\mathbf{W}^s\mathbf{s}_T)$$
$$\mathbf{p}^e = \text{softmax}(\mathbf{C}^h\mathbf{W}^e\mathbf{s}_T)$$

where $\mathbf{W}^s$ and $\mathbf{W}^e$ are learned matrices of size $2h \times 2h$, and $\mathbf{p}^s_i$ (resp. $\mathbf{p}^e_i$) represents the probability that position i is the start (resp. the end) of the answer span with $0 \leq i < n$.

At inference, we use unnormalized exponential instead of softmax to make scores compatible across paragraphs in

Dataset	Train		Test
	Plain	DS	
bAbI	10,000‡	-	1,000‡
Wikihop	43,738	-	5,129†
SQuAD	87,599	-	10,570†
CuratedTREC	1,486*	2,643	694
WebQuestions	3,778*	6,308	2,032
WikiMovies	96,185*	100,528	9,952

Table 1. Number of questions for each dataset used in this paper. DS: distantly supervised training data. *: These training sets are not used as is because no paragraph is associated with each question. †: Corresponds to Wikihop and SQuAD development set. ‡: All figures for bAbI tasks refer to a single task.

one or several documents. We select the span $[i, j]$ which maximizes $\mathbf{p}_i^s \mathbf{p}_j^e$ for $i \leq j \leq i + 15$.

4. Experimental Setup

4.1. Datasets

We test our model's ability to answer questions with various types of context, synthetic stories, paragraphs, documents, and full encyclopedia (Wikipedia), using the datasets described in this section. Two evaluation metrics are used depending on the dataset: the exact string match (EM) between the predicted span and the answer string and the F1 score, which measures the harmonic mean of precision and recall at the token level. All scores reported in this paper are percentages. In the full Wikipedia setting, we use the open-domain question answering datasets and the same Wikipedia dump[2] as (Chen et al., 2017; Wang et al., 2017a). Statistics of the datasets are given in Table 1.

BAbI Tasks This dataset consists in 20 simple dialog tasks. In line with previous literature on the dataset, we consider a task solved if the EM accuracy reaches 95 on the validation set. We select the set of tasks for which the answer is a single word, hence excluding tasks 8 and 19.

QAngaroo Wikihop This dataset was introduced in (Welbl et al., 2017). In this dataset, questions are not sentences but consist in the concatenation of a subject and a knowledge base relation, e.g. *place_of_birth caspar john* would mean *Where was Caspar John born?*. We do not use any special approach for such inputs apart from a forced tokenization around the _ character. In particular, the model does not learn representations for relations but only for the individual words that appear in the relation's text form, e.g. *place of birth*. Documents are sequences of *supports* that contain information coming from several Wikipedia pages. We concatenate all the supports for a given example and use it as a single context.

SQuAD The Stanford Question Answering Dataset (Rajpurkar et al., 2016) contains 87k questions over Wikipedia for training and 10k for development, with a large hidden test set which can only be accessed by the SQuAD creators. Each example is composed of a paragraph extracted from a Wikipedia article and an associated human-generated question. The answer is always a span from this paragraph and a model is given credit if its predicted answer matches it.

CuratedTREC This dataset is based on the benchmarks from the TREC question answering tasks that have been curated by Baudiš & Šedivỳ (2015). We use the large version, which contains a total of 2,180 questions.

WebQuestions Berant et al. (2013) built this dataset around the task of answering questions from the Freebase knowledge base. It was created by crawling questions through the Google Suggest API, and then obtaining answers using Amazon Mechanical Turk. Each answer has been converted by Chen et al. (2017) to text by using entity names so that the dataset does not reference Freebase IDs and is purely made of plain text question-answer pairs.

WikiMovies Miller et al. (2016) collected 96k question-answer pairs in the domain of movies. Originally created from the OMDb and MovieLens databases, the examples are built such that they can also be answered by using a subset of Wikipedia as the knowledge source (the title and the first section of articles from the movie domain).

4.2. Distant Supervision

Unlike recent machine reading comprehension datasets, CuratedTREC, WebQuestions and WikiMovies only contain question, answer pairs and lack any supporting documents. To gather supporting documents we resort to distant supervision (Mintz et al., 2009). We follow the same distant supervision setup as (Chen et al., 2017) but since we are interested in machine reading over long documents, we associate full documents instead of single paragraphs, and thus increase the character limit to 100,000 instead of 1,500 originally. We use those distantly supervised training sets to fine-tune a model trained on SQuAD. Corresponding results are given in Table 6.

4.3. Implementation Details

For training, we batch together examples with similar document sizes, padding all matrices in the question and context directions when necessary. Within each epoch, mini-

[2] We use the english-language Wikipedia dump of 2016-12-21 (https://dumps.wikimedia.org/enwiki/latest), which contains 5,075,182 articles consisting of 9,008,962 unique uncased token types.

Task	DrQA		Weaver	
	single	multi-task	single	multi-task
1	100	100	100	100
2	98.1	46.6	99.2	99.7
3	45.4	55.6	99.3	99.7
4	100	96.3	100	100
5	98.9	98.1	99.8	99.8
6	98.4	99.9	100	100
7	100	99.1	99.8	100
9	100	99.8	100	100
10	99.7	100	100	100
11	100	100	100	100
12	100	100	100	100
13	100	100	100	100
14	99.8	96.0	99.9	99.9
15	100	57.6	99.5	100
16	47.7	44.5	53.3	49.0
17	94.9	60.4	100	100
18	100	93.4	100	100
20	100	100	100	100
Failed	3	6	1	1

Table 2. Test accuracies of Weaver and DrQA on bAbI-10k. We did not test on tasks 8 and 19 because they require to answer with multiple words but DrQA and Weaver are not designed to do so.

batches are shuffled randomly. Weaver is trained via optimizing the sum of the cross-entropy losses for the start and end points of the answer span for each training example. To this end, we use Adamax with a learning rate of 0.002 on the padded mini-batches. Weights are initialized randomly according to a Gaussian of mean 0 and variance $1/n$ where n is the number of the neuron's input channels. We apply dropout at a rate of 0.2 to the output of all LSTMs as well as dropout at a rate of 0.3 on M_0. Model selection is done using the validation set of each dataset.

On all datasets but bAbI, we use the Stanford CoreNLP toolkit (Manning et al., 2014) to tokenize the input documents and generate part-of-speech and named entity recognition features. Weaver is implemented in PyTorch.[3]

For datasets providing a list of candidate answers (QAngaroo WikiHop, WebQuestions and WikiMovies), we restrict the answer span to be in this list during prediction.

5. Experimental Results

This section presents the performance of Weaver on a wide range of tasks, from reasoning on short synthetic stories (bAbI) to open domain question answering on Wikipedia.

[3] http://pytorch.org.

5.1. Reasoning on Synthetic Stories

On this dataset, we do not use fixed pretrained embeddings but learn embeddings of dimension 128 based on a random initialization. With a 1-block co-encoder with $h = 128$ and 3 hops, Weaver solves 17 out of the 18 tasks that are relevant for it, as shown in Table 2. It can solve them both in the single setting (each task trained separately) and the multi-task setting (all tasks trained together), while DrQA could solve only 15 single tasks and 12 in a multi-task setting. Note that solving all bAbI tasks is not trivial, since only two dedicated models did it (Seo et al., 2016; Henaff et al., 2016) to the best of our knowledge.

Several bAbI tasks require to answer with words that do not belong to the context such as "yes", "no" or "maybe". Standard span-based machine reading models cannot readily do that. We overcome this limitation here by prepending the context with a list of extra words that can be picked by the model to answer. This list is made as the union of all answers to these tasks on the training set, which forms a sequence of 18 words present at the beginning of every context. As shown by the good results in the multi-task setting for which we prepend those extra words for all examples, Weaver can learn to use them only when it is relevant.

The number of hops has a large impact on the performance on some tasks such as Task 3. As shown in Table 4, this task could not be solved without one memory network hop, and the performance increases steadily with the number of hops. This effect can also be observed on other datasets even though it is most visible on the bAbI dataset thanks to its synthetic nature.

5.2. Answering from Paragraphs

5.2.1. QAngaroo WikiHop

We use a 1-block co-encoder with $h = 200$ and between 1 and 5 memory network hops. As shown in Table 4, Weaver can achieve up to 64.1 EM accuracy on the development set, with a single hop being sufficient to achieve the best performance. Weaver achieves 65.3 EM accuracy on the hidden test set, which beats the previous state of the art by a 5-point margin.

5.2.2. SQuAD

We use a 2-block co-encoder with $h = 200$ and train for 20 epochs. Table 3 shows that our model achieves 82.8 F1 on the hidden test set, which is comparable to the best published architectures: it is better than Conductor-net (Liu et al., 2017a) and Reinforced Mnemonic Reader (M-Reader + RL) (Hu et al., 2017) and outperformed by DCN+ (Xiong et al., 2017), FusionNet (Huang et al., 2017) and SAN (Liu et al., 2017b). It is worth noting that all those methods have only been tested on SQuAD, except

	Dev set		Test set	
	EM	F1	EM	F1
DrQA	69.5	78.8	70.7	79.3
Conductor-net	72.1	81.4	72.6	81.4
M-Reader+RL	72.1	81.6	73.2	81.8
DCN+	74.5	83.1	75.1	83.1
FusionNet	75.3	83.6	76.0	83.9
SAN	**76.2**	**84.1**	**76.8**	**84.4**
Weaver	74.1	82.4	74.4	82.8

Table 3. Results of single models on SQuAD. We include as baselines the best published models as of February 2018.

Nb of Hops	bAbI Task 3 (EM)	WikiHop (EM)	SQuAD (Dev F1)
0	66.5	61.2	81.9
1	**99.0**	**64.1**	81.9
3	**99.3**	63.4	82.0
5	**99.5**	63.0	**82.4**
7	-	-	82.1
9	-	-	81.8

Table 4. Impact of the number of hops in the memory network final step of Weaver (EM: exact match accuracy).

M-Reader + RL that was also applied on TriviaQA.

The addition of the memory network (with 1 or more hops) improves the F1 performance by 0.9 points compared to using directly the question and context representations. We empirically find that a deeper memory network (i.e. more hops) requires a lighter co-encoding (i.e. fewer blocks) to achieve the same performance. We also find that the variability due to the random network initialization implies a standard deviation of the results of 0.2 F1 points.

We benchmarked several architecture variants. We added a character-level LSTM to embed out-of-dictionary words and incorporate this data into the M_0 layer. This addition did not improve performance measurably. We can attribute this to the fact that most words in SQuAD are covered by the FastText word embeddings: in fact, out-of-dictionary words represent only a total of 61k words or 1.5% of the full corpus (which comprises 3.9M words, duplicates included). Most of these missing words may be interpretable thanks either to surrounding words or to part-of-speech and named entity recognition features. In other experiments, we also added self-attention layers but they did not bring any improvement to what the plain co-encoder of Weaver can do.

5.3. Answering from Documents

We then test our model in the more difficult setting where a full document is given as context to answer the question.

	Train	Test	EM	F1
DrQA	paragraph	full doc.	49.4	58.0
DrQA*	paragraph	full doc.	59.1	67.0
DrQA*	full doc.	full doc.	64.7	73.2
Weaver	paragraph	full doc.	60.6	69.7
Weaver	full doc.	full doc.	**67.0**	**75.9**

Table 5. Document-scale results with training and testing on SQuAD (using the dev set for evaluation). For each architecture, we indicate the setting used for training and testing (single paragraph or full document). The DrQA* model refers to an improved version of DrQA (see section 5.3 for details).

On SQuAD, documents are on average 40 times larger than individual paragraphs. All paragraphs of the document are processed independently and we pick the answer span with the maximum score across all paragraphs.

We compare with DrQA trained on paragraphs and also with an improved version, which we term DrQA*, that has the following changes:

- We use the same FastText pretrained embeddings as Weaver. (Mikolov et al., 2017) showed that this improves the performance of DrQA on SQuAD, compared to using the GloVe embeddings (Pennington et al., 2014) as in the original DrQA model.
- The relatively small size of the DrQA model makes it practical to read the whole document contiguously, as opposed to processing paragraphs independently. This also helps it achieve better results.

For both models, training on full documents is critical to get good performance on this task. To speed up training of the Weaver model, instead of processing all paragraphs, we pick the paragraph containing the correct answer and sample 5 other paragraphs at random from the document. We take the final softmax across all paragraphs. As seen on Table 5, our model gains 6 points in F1 score compared to the model only trained to answer on single paragraphs and also significantly outperforms the baselines.

5.4. Answering from the Full English Wikipedia

To answer questions from all of Wikipedia, we use a 2-step pipeline similar to (Chen et al., 2017):

1. We first use the same retriever as (Chen et al., 2017) to find k documents related to the question.[4] This retriever uses bigram hashing and TF-IDF matching to score documents relative to the question.

[4] https://github.com/facebookresearch/DrQA

		SQuAD	CuratedTREC	WebQuestions	WikiMovies
YodaQA	- *addtl sources*	-	31.3	**39.8**	-
DrQA	- *SQuAD train*	27.1	19.7	11.8	24.5
	- *fine-tuning*	28.4	25.7	19.5	34.3
DrQA*	- *SQuAD train*	39.5	21.3	14.2	31.9
	- *fine-tuning*	40.4	28.8	24.3	**46.0**
Reinf. reader-ranker	- *fine-tuning*	29.1	28.4	17.1	38.8
Weaver	- *SQuAD train*	**42.3**	21.3	13.0	33.6
	- *fine-tuning*	-	**37.9**	23.7	43.9

Table 6. Results when answering questions using the full English Wikipedia (top-1 EM accuracy, using SQuAD evaluation script)

2. We then run the Weaver model (trained on full documents) as a document reader on the top k retrieved documents, and return the span with the maximum score across all documents.

All results are reported in Table 6, where we compare with DrQA, DrQA*, the Reinforced Reader-Ranker (Wang et al., 2017a) and YodaQA (Baudiš, 2015). In addition to showing better performance than the original DrQA model on a single full document, Weaver is also able to handle more documents properly. While the EM score for DrQA reaches its maximum with 10 documents, we are able to increase the performance of the full system with Weaver as a reader by increasing the number of retrieved documents up to 25, see Fig. 2. This allows to reach a final EM accuracy of 42.3 on SQuAD, which is more than 12 points better than the previous best reported performance.

We also evaluate our model on CuratedTREC, WebQuestions and WikiMovies, first by directly testing the model trained on the SQuAD training set, and then after fine-tuning this model on each dataset independently. For the fine-tuning stage on a given dataset, we take our best model on the SQuAD full document setting and train it for 1 epoch on the distantly supervised corresponding training set (20 epochs for WebQuestions). Performance on CuratedTREC presents a significant improvement over the state of the art (6.6 EM gain). Note that we only count an exact match if the provided regular expression matches the full span returned by the model. If we also count an exact match when the regex matches a substring (as intended by the dataset authors[5]), we obtain 39.2 EM with the model trained on SQuAD and 43.8 EM with the fine-tuned model. Performance on WikiMovies also increases significantly although our improved baseline DrQA* performs better when fine-tuned on that dataset.

Performance on WebQuestions remains lower than that of YodaQA but, in addition to Wikipedia, YodaQA uses Freebase as additional knowledge source, which gives a key advantage on this dataset created from Freebase.

[5] https://github.com/brmson/dataset-factoid-curated

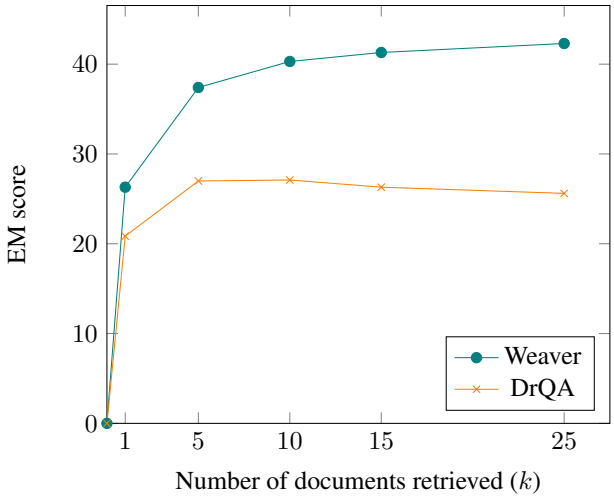

Figure 2. Impact of the number of documents retrieved for each question in the full Wikipedia setting.

5.5. Ablation Study

To verify that the prediction performance improvements come from the co-encoding module, we conduct an ablation study on the SQuAD dev set. We report results in Table 7. Architecture choices such as the exact input encoding or the presence of modules such as the convolutional layer or the memory network play a minor role in the prediction performance. However, a model without the stacked recurrent neural networks reaches only 33 F1. This shows that the RNNs applied in a weaving pattern are the major factor that enables a good prediction performance.

6. Conclusions and Future Work

In this paper, we introduced Weaver, a novel way of building question and context representations jointly for machine reading and showed how to use it to solve span-based question answering tasks. We found that Weaver was

Model	Dev EM	Dev F1
Weaver	**74.1**	**82.4**
CatQC	71.8	80.8
CatQCDotProduct	72.3	81.3
NoRnn	22.7	33.0
NoConv11	71.8	81.1
NoMemNet	72.9	81.9

Table 7. Ablation study on SQuAD dev set. Ref: reference model. CatQC: we substitute the input representation mentioned in Eq. (1) by a direct concatenation of question and context embeddings. CatQCDotProduct: starting from the CatQC architecture, we add back the dot-product between question and context embedding to the input representation. NoRnn: weaved RNN layers are replaced with a linear projection. NoConv11: we remove the convolution layer. NoMemNet: we set the number of hops to 0 in the memory network.

able to achieve near state-of-the-art performance on various closed-domain problems such as SQuAD or bAbI, while significantly outperforming the previous state of the art on the open-domain setting on various datasets.

Future work may entail improving the core of the co-encoder. Its architecture is based on BiLSTM building blocks. While they have very powerful sequence modeling capabilities, (Dauphin et al., 2017) has shown that they can be replaced by convolutional approaches in the domain of machine translation with both a gain in efficiency and task performance. Another angle of attack would be to learn jointly parts of the document retriever with the co-encoder in order to maximize the performance of the whole pipeline, as it was attempted in (Wang et al., 2017a) through reinforcement learning.

Acknowledgments We would like to thank Pranav Rajpurkar for results on the SQuAD test set, Johannes Welbl for results on the WikiHop test set, as well as Louis Martin and Frédéric Arnault for helpful discussions.

References

Auer, Sören, Bizer, Christian, Kobilarov, Georgi, Lehmann, Jens, Cyganiak, Richard, and Ives, Zachary. Dbpedia: A nucleus for a web of open data. *The semantic web*, pp. 722–735, 2007.

Baudiš, Petr. YodaQA: a modular question answering system pipeline. In *POSTER 2015-19th International Student Conference on Electrical Engineering*, pp. 1156–1165, 2015.

Baudiš, Petr and Šedivỳ, Jan. Modeling of the question answering task in the YodaQA system. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pp. 222–228. Springer, 2015.

Berant, Jonathan, Chou, Andrew, Frostig, Roy, and Liang, Percy. Semantic parsing on freebase from question-answer pairs. In *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1533–1544, 2013.

Chen, Danqi, Fisch, Adam, Weston, Jason, and Bordes, Antoine. Reading Wikipedia to Answer Open-Domain Questions. In *Association for Computational Linguistics (ACL)*, 2017.

Cho, Kyunghyun, van Merrienboer, Bart, Gülçehre, Çaglar, Bougares, Fethi, Schwenk, Holger, and Bengio, Yoshua. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078, 2014. URL http://arxiv.org/abs/1406.1078.

Dauphin, Yann N, Fan, Angela, Auli, Michael, and Grangier, David. Language modeling with gated convolutional networks. In *International Conference on Machine Learning (ICML)*, 2017.

Dhingra, Bhuwan, Liu, Hanxiao, Yang, Zhilin, Cohen, William W, and Salakhutdinov, Ruslan. Gated-attention readers for text comprehension. *arXiv preprint arXiv:1606.01549*, 2016.

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

Henaff, Mikael, Weston, Jason, Szlam, Arthur, Bordes, Antoine, and LeCun, Yann. Tracking the world state with recurrent entity networks. *arXiv preprint arXiv:1612.03969*, 2016.

Hermann, Karl Moritz, Kočiský, Tomáš, Grefenstette, Edward, Espeholt, Lasse, Kay, Will, Suleyman, Mustafa, and Blunsom, Phil. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

Hill, Felix, Bordes, Antoine, Chopra, Sumit, and Weston, Jason. The Goldilocks Principle: Reading children's books with explicit memory representations. In *International Conference on Learning Representations (ICLR)*, 2016.

Hu, Minghao, Peng, Yuxing, and Qiu, Xipeng. Reinforced mnemonic reader for machine comprehension. *CoRR, abs/1705.02798*, 2017.

Huang, Hsin-Yuan, Zhu, Chenguang, Shen, Yelong, and Chen, Weizhu. Fusionnet: Fusing via fully-aware attention with application to machine comprehension. *arXiv preprint arXiv:1711.07341*, 2017.

Jia, Robin and Liang, Percy. Adversarial examples for evaluating reading comprehension systems. *arXiv preprint arXiv:1707.07328*, 2017.

Joshi, Mandar, Choi, Eunsol, Weld, Daniel S., and Zettlemoyer, Luke. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, Vancouver, Canada, July 2017. Association for Computational Linguistics.

Kadlec, Rudolf, Schmid, Martin, Bajgar, Ondrej, and Kleindienst, Jan. Text understanding with the attention sum reader network. *arXiv preprint arXiv:1603.01547*, 2016.

Liu, Rui, Wei, Wei, Mao, Weiguang, and Chikina, Maria. Phase conductor on multi-layered attentions for machine comprehension. *arXiv preprint arXiv:1710.10504*, 2017a.

Liu, Xiaodong, Shen, Yelong, Duh, Kevin, and Gao, Jianfeng. Stochastic answer networks for machine reading comprehension. *arXiv preprint arXiv:1712.03556*, 2017b.

Manning, Christopher D, Surdeanu, Mihai, Bauer, John, Finkel, Jenny, Bethard, Steven J, and McClosky, David. The stanford corenlp natural language processing toolkit. In *Association for Computational Linguistics (ACL)*, pp. 55–60, 2014.

Mikolov, Tomas, Grave, Edouard, Bojanowski, Piotr, Puhrsch, Christian, and Joulin, Armand. Advances in pre-training distributed word representations. *arXiv preprint arXiv:1712.09405*, 2017.

Miller, Alexander H., Fisch, Adam, Dodge, Jesse, Karimi, Amir-Hossein, Bordes, Antoine, and Weston, Jason. Key-value memory networks for directly reading documents. In *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1400–1409, 2016.

Miller, Alexander H, Feng, Will, Fisch, Adam, Lu, Jiasen, Batra, Dhruv, Bordes, Antoine, Parikh, Devi, and Weston, Jason. Parlai: A dialog research software platform. *arXiv preprint arXiv:1705.06476*, 2017.

Mintz, Mike, Bills, Steven, Snow, Rion, and Jurafsky, Daniel. Distant supervision for relation extraction without labeled data. In *Association for Computational Linguistics and International Joint Conference on Natural Language Processing (ACL/IJCNLP)*, pp. 1003–1011, 2009.

Nguyen, Tri, Rosenberg, Mir, Song, Xia, Gao, Jianfeng, Tiwary, Saurabh, Majumder, Rangan, and Deng, Li. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.

Pennington, Jeffrey, Socher, Richard, and Manning, Christopher. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

Rajpurkar, Pranav, Zhang, Jian, Lopyrev, Konstantin, and Liang, Percy. SQuAD: 100,000+ questions for machine comprehension of text. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

Richardson, Matthew, Burges, Christopher JC, and Renshaw, Erin. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*, volume 1, pp. 2, 2013.

Seo, Minjoon, Min, Sewon, Farhadi, Ali, and Hajishirzi, Hannaneh. Query-reduction networks for question answering. *arXiv preprint arXiv:1606.04582*, 2016.

Sukhbaatar, Sainbayar, Szlam, Arthur, Weston, Jason, and Fergus, Rob. End-to-end memory networks. *Proceedings of NIPS*, 2015.

Vrandečić, Denny and Krötzsch, Markus. Wikidata: a free collaborative knowledgebase. *Communications of the ACM*, 57(10):78–85, 2014.

Wang, Shuohang, Yu, Mo, Guo, Xiaoxiao, Wang, Zhiguo, Klinger, Tim, Zhang, Wei, Chang, Shiyu, Tesauro, Gerald, Zhou, Bowen, and Jiang, Jing. R3: Reinforced reader-ranker for open-domain question answering. *arXiv preprint arXiv:1709.00023*, 2017a.

Wang, Wenhui, Yang, Nan, Wei, Furu, Chang, Baobao, and Zhou, Ming. 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, 2017b.

Welbl, Johannes, Stenetorp, Pontus, and Riedel, Sebastian. Constructing datasets for multi-hop reading comprehension across documents. *CoRR*, abs/1710.06481, 2017. URL http://arxiv.org/abs/1710.06481.

Weston, Jason, Bordes, Antoine, Chopra, Sumit, and Mikolov, Tomas. Towards ai-complete question answering: a set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015.

Xiong, Caiming, Zhong, Victor, and Socher, Richard. Dcn+: Mixed objective and deep residual coattention for question answering. *arXiv preprint arXiv:1711.00106*, 2017.

Yang, Yi, Yih, Wen-tau, and Meek, Christopher. WikiQA: A challenge dataset for open-domain question answering. In *EMNLP*, pp. 2013–2018, 2015.