An LSTM Model for Cloze-Style Machine Comprehension

Shuohang Wang and Jing Jiang

School of Information Systems, Singapore Management University
80 Stamford Road, Singapore
shwang.2014@phdis.smu.edu.sg,jingjiang@smu.edu.sg

Abstract. Machine comprehension is concerned with teaching machines to answer reading comprehension questions. In this paper we adopt an LSTM-based model we designed earlier for textual entailment and propose two new models for cloze-style machine comprehension. In our first model, we treat the document as a premise and the question as a hypothesis, and use an LSTM with attention mechanisms to match the question with the document. This LSTM remembers the best answer token found in the document while processing the question. Furthermore, we observe some special properties of machine comprehension and propose a two-layer LSTM model. In this model, we treat the question as a premise and use LSTMs to match each sentence in the document with the question. We further chain up the final states of these LSTMs using another LSTM in order to aggregate the results. When evaluated on the commonly used CNN/Daily Mail dataset, both of our models are quite competitive compared with the state of the art, and the second two-layer model outperforms the first model.

1 Introduction

In recent years there has been much interest in machine reading comprehension [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]. The task is to train machines to answer questions related to a given document. The questions may be in different formats, including multiple choice questions [16,17], cloze-style questions [1,9,12] and even questions that require summarization and inference capabilities to answer [18]. In this paper, we focus on cloze-style questions, which are generated by masking certain words from a given document. A commonly-used cloze-style machine comprehension dataset is the CNN/Daily Mail dataset created by [1]. In this dataset, each question is generated by removing an entity from a manually created summary point of a news article, and the removed entity serves as the correct answer to the question. An example question in this dataset with part of its corresponding news article and its correct answer is shown in Table 1.[1]

In this paper, we focus on this cloze-style machine comprehension task and use the CNN/Daily Mail dataset for evaluation. Motivated by the observation

[1] Note that all the entities have been anonymized by [1]. This is to ensure that the questions cannot be answered trivially based on world knowledge learned elsewhere.

Table 1. An example document excerpt, question and answer.

Document
@entity5 (@entity6) an impressive art collection assembled by the late actress and @entity3 icon , @entity4 , has officially been offered for purchase. the collection , which includes works by some of the greatest artists of the 20th century , went under the hammer in @entity13 on march 31 , following a tour of @entity5 , @entity15 , @entity16 and @entity17 . ... the 750 - piece collection , which fetched a total of $ 3.64 million , featured bronze sculptures , jewelry , and a number of decorative arts and paintings , which were sold at @entity50 auction house in @entity13 . ...
Question
a collection of 750 items belonging to legendary actress @entity4 has been auctioned off at @entity50 in **@placeholder**
Answer
@entity13

that the answer to a question should appear in the document near tokens that are semantically similar to the tokens in the question, we approach the task by trying to match the question to the document (or vice versa). This sequence alignment process is similar to textual entailment where the goal is to determine whether a premise entails a hypothesis. Because of the similarity of the cloze-style machine comprehension task and textual entailment, we borrow a match-LSTM model for textual entailment that we proposed earlier [19], and study how match-LSTM could be applied to this task. For textual entailment, match-LSTM sequentially processes a given hypothesis sentence and updates the prediction of whether it matches a given premise sentence or not. For machine comprehension, however, we use the match-LSTM to sequentially process a given question and update our belief of which token in the given passage should be the correct answer token.

Observing some special properties of the machine comprehension task we are dealing with, we further propose a second model which uses a two-layer hierarchical LSTM architecture. Specifically, we treat the question as a premise instead in this model, and break down the passage into sentences and treat each sentence as a hypothesis to be matched with the question using an LSTM. These LSTMs serve two purposes: They measure how well a sentence from the passage matches the question, and they remember potential answers to the question found in each sentence. Another LSTM at the top layer then combines the final states from each of these bottom-layer LSTMs to make the final prediction.

Using the CNN/Daily Mail dataset created by [1], we show that both models can perform competitively compared with a number of baseline methods we consider, and our second hierarchical model works better than the first model. We also conduct further analyses to understand the underlying mechanisms of our two models.

The main contribution of our work is to show that the match-LSTM model we proposed earlier for textual entailment can be easily adapted to solve the cloze-style machine comprehension problem and performs competitively with other state-of-the-art models.

2 Related Work

Machine Comprehension: Although it has always been one of the central goals of artificial intelligence, the task of machine comprehension of text (MCT) has recently gained more attention, partly because of the creation of a few benchmark datasets including the MCTest data [16], the Children's Book Test data [9], the CNN/Daily Mail dataset [1] and the Stanford QA dataset (SQuAD) [8]. Although all of them were created for machine comprehension, there are some minor differences in the way the task is defined. In this paper, we focus on the setup by [1] and thus use their CNN/Daily Mail data for evaluation. Although with the answer being a single token, their setup is somewhat simplified, we believe that techniques developed for this simple setup still have potentials to be extended for more realistic settings of machine comprehension.

Researchers have noted the similarity between machine comprehension and other traditional NLP tasks including question answering (QA) and textual entailment (TE). As a result, many solutions to machine comprehension are based on techniques for QA and TE. For example, [3] formulated the task as textual entailment and proposed a max-margin framework to learn the answer-entailing structure. [6] presented a number of CNN models for machine comprehension, and the best performing one also treats the task as textual entailment. In our work, we also adopt an LSTM-based method we recently developed for textual entailment to approach the machine comprehension task, but we make some significant changes to the neural network architecture to suit our task.
Modeling Text with LSTMs and Attention Mechanisms: Long short-term memory (LSTM) networks are a special kind of recurrent neural networks (RNNs) and have recently been widely used to encode textual sequences to solve a number of NLP problems, including machine translation [20], abstractive text summarization [21], textual entailment [19] and machine comprehension [1]. Oftentimes, these models incorporate neural attention mechanisms. The idea is to adjust the machine's attentions to different regions of the text sequence that has been processed in order to make new decisions, and the attention weights are based on non-linear transformations learned from the training data. In our work, we also adopt LSTMs with attention mechanisms.
Comparison with Other Models: There have been several models proposed in recent years for machine comprehension [1,7,9,10,11,12,13,14,15]. There are roughly two directions of these developing models. One direction is to focus more on modeling the interaction between the document and the question. For example, attention-over-attention [12], gated-attention [13] and bidirectional attention flow [15] are proposed to build representations between the document and the question. The other direction is to more reasonably predict the answer based on the representations built between the document and the question. For example, [11] applied the pointer network for answer extraction, [7] and [14] fine-

tuned the answer prediction with a multi-step reasoner. In our work, we focus on the first direction. Unlike previous studies which do not consider the document structure and build attention weights independently, our model consists of document hierarchical structure and dynamic attention mechanism for building the representations between the document and the question.

3 Models

3.1 Preliminaries and General Framework

We consider the following setup of the task. The input is a pair of a document d and a question q, each of which is a sequence of tokens. Furthermore, one of the tokens in q is a special placeholder `X` that replaces the correct answer token[2]. The expected output is a token a from document d that answers the question. With a set of training triplets (d, q, a), we would like to learn a model that can predict a given any pair of (d, q).

We assume that q is represented by $\mathbf{X}^q = (\mathbf{x}_1^q, \mathbf{x}_2^q, \ldots, \mathbf{x}_{N_q}^q)$, where each $\mathbf{x}_i^q \in \mathbb{R}^l$ is an embedding vector representing the i^{th} token in q. Similarly each d is represented by $\mathbf{X}^d = (\mathbf{x}_1^d, \mathbf{x}_2^d, \ldots, \mathbf{x}_{N_d}^d)$ where each $\mathbf{x}_j^d \in \mathbb{R}^l$. In our experiments, these token embeddings were pre-trained and not updated during the training of the machine comprehension models.

As we have pointed out, machine comprehension has been treated as a textual entailment problem before with good results [3,6]. The general idea is to treat the document as the premise and the question together with a candidate answer as the hypothesis. If the premise can entail the hypothesis, then there is a good chance that the candidate answer is correct. For the machine comprehension task we are dealing with, however, there are too many candidate answers to consider because each token in the document is one. We therefore do not combine the question with a candidate answer to form a hypothesis but simply treat the question containing the placeholder token `X` as the hypothesis. And instead of predicting whether the hypothesis can be entailed from the premise, we learn models that directly predict the answer token. This framework is the same as the one employed by [1].

We use long short-term memory (LSTM) networks to process the document and the question just like [1], but our model is significantly different from theirs. We borrow the idea of a match-LSTM architecture for textual entailment that we previously proposed [19]. Specifically, we use an LSTM to process the document and the question at the same time, where the hidden states of this LSTM represent the model's updated belief of which token from the document is the best answer. The last hidden state of the LSTM can then be used to predict the answer token. Our first model is based on this idea.

We further observe that there are some potential limitations with our first model. We therefore propose a second model that uses two layers of LSTMs.

[2] Entity recognition and coreference resolution were performed when the dataset was created. Each answer is therefore always a single token.

The bottom-layer LSTMs match the question with sentences of the document while the top-layer LSTM aggregates the results from the bottom-layer LSTMs. This novel two-layer architecture specifically designed for our machine comprehension task allows us to more easily assess the probability of a token being the answer based on its context, and our experiments show that this second model outperforms the first model.

3.2 A Multi-pass Reader

The first model we propose works as follows. First, we use two standard LSTMs to encode the document and the question, respectively. Let us use $\mathbf{h}_i^q \in \mathbb{R}^h$ to denote the produced hidden state representing the i^{th} token in the question and $\mathbf{h}_j^d \in \mathbb{R}^h$ the hidden state representing the j^{th} token in the document.

We then use another LSTM which we call the answer-LSTM (or a-LSTM) to further process $\mathbf{h}^q$. At position i, we first use an attention mechanism to derive a series of attention weights $s_{i,j}$ linking $\mathbf{h}_i^q$ to each $\mathbf{h}_j^d$ in the document:

$$\begin{aligned}
\mathbf{m}_{i,j} &= \tanh(\mathbf{W}^q\mathbf{h}_i^q + \mathbf{W}^d\mathbf{h}_j^d + \mathbf{W}^a\mathbf{h}_{i-1}^a),\\
s_{i,j} &\propto \exp(\mathbf{w}^s\mathbf{m}_{i,j}),
\end{aligned}$$

where $\mathbf{W}^q, \mathbf{W}^d, \mathbf{W}^a \in \mathbb{R}^{h\times h}$ are weight matrices, $\mathbf{w}^s \in \mathbb{R}^h$ is a weight vector and $\mathbf{h}_{i-1}^a$ is the hidden state produced by the a-LSTM at the previous position. We can think of $s_{i,j}$ as the probability that the i^{th} token in the question is aligned with the j^{th} token in the document. For example, we expect that for the placeholder token `X` in the question, ideally it should be aligned with the correct answer token in the document, and therefore the attention weight for `X` with that token presumably should be high.

Using these attention weights, we obtain a weighted sum of the document's hidden representations as follows:

$$\overline{\mathbf{h}}_i^d = \sum_j s_{i,j}\mathbf{h}_j^d.$$

Subsequently, we concatenate this $\overline{\mathbf{h}}_i^d$ with $\mathbf{h}_i^q$ into $\mathbf{e}_i = [\overline{\mathbf{h}}_i^d; \mathbf{h}_i^q]$, and this $\mathbf{e}_i$ is used as input to the a-LSTM.

Following the general design of LSTMs, the hidden state $\mathbf{h}_i^a$ of the a-LSTM is derived as follows:

$$\begin{aligned}
\mathbf{i}_i^a &= \sigma(\mathbf{W}^{\text{i}}\mathbf{e}_i + \mathbf{V}^{\text{i}}\mathbf{h}_{i-1}^a + \mathbf{b}^{\text{i}}),\\
\mathbf{f}_i^a &= \sigma(\mathbf{W}^{\text{f}}\mathbf{e}_i + \mathbf{V}^{\text{f}}\mathbf{h}_{i-1}^a + \mathbf{b}^{\text{f}}),\\
\mathbf{o}_i^a &= \sigma(\mathbf{W}^{\text{o}}\mathbf{e}_i + \mathbf{V}^{\text{o}}\mathbf{h}_{i-1}^a + \mathbf{b}^{\text{o}}),\\
\mathbf{u}_i^a &= \tanh(\mathbf{W}^{\text{u}}\mathbf{e}_i + \mathbf{V}^{\text{u}}\mathbf{h}_{i-1}^a + \mathbf{b}^{\text{u}}),\\
\mathbf{c}_i^a &= \mathbf{f}_i^a \odot \mathbf{c}_{i-1}^a + \mathbf{i}_i^a \odot \mathbf{u}_i^a,\\
\mathbf{h}_i^a &= \mathbf{o}_i^a \odot \mathbf{c}_i^a,
\end{aligned}$$

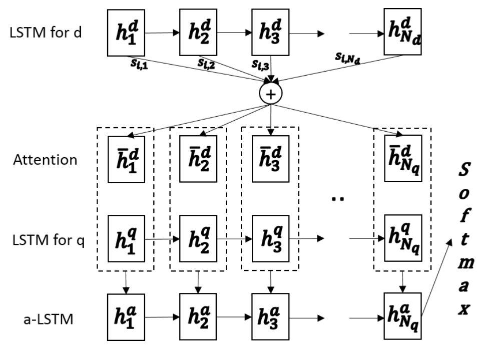


Fig. 1. The Multi-pass Reader model. The dot rectangular is the concatenation of a document attention representation and a question state.

where the $\mathbf{i}_i^a$, $\mathbf{f}_i^a$, $\mathbf{o}_i^a$ and $\mathbf{c}_i^a$ are the input gates, forget gates, output gates and cell state at position i, and the different $\mathbf{W}^*$, $\mathbf{V}^*$ and $\mathbf{b}^*$ are weight matrices and weight vectors to be learned.

Here the a-LSTM essentially uses its hidden states to record and keep updating the best answer found so far from the document. Eventually, the final hidden state $\mathbf{h}_{N_q}^a$ of the a-LSTM is used to predict the answer token through a softmax layer. This model is depicted in Figure 1.

We can see that in this model, at each position i of the question, the a-LSTM has to go back and revisit every position in the document in order to derive the attention weights. This is analogous to a reader scanning the document multiple times, once for every token in the question. We therefore call this model the multi-pass reader model.

3.3 A Single-pass Reader

We observe that there are some limitations of the multi-pass reader model above applied to machine comprehension. First, in machine comprehension, the document is a long sequence, consisting of multiple sentences, whereas the question is usually a single sentence. With the multi-pass reader, we will need to scan the long document multiple times, which seems inefficient and unlikely to be the way humans read. To address this problem, we propose to switch the roles the document and the question play in the answer-LSTM. Imagine a different reader who, before reading the document, first reads the question. With the question in mind, he then reads the document only once, and while he is reading, he decides whether or not to use the current token in the document as the answer to the question. This can be accomplished by using an LSTM to process the document and using an attention mechanism with respect to the question kept in memory to help determine the input gates and forget gates. We can thus design a single-pass reader model.

Second, we also observe that in our dataset the question is oftentimes a paraphrase of one or a few consecutive sentences of the document. This means

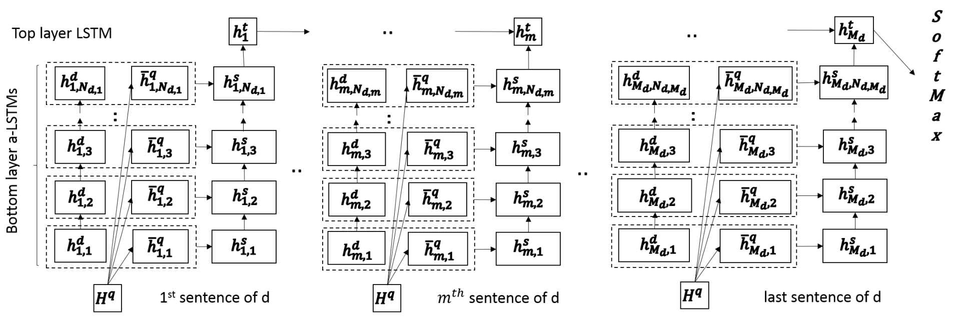


Fig. 2. The Single-pass Reader model. Here $\mathbf{h}^d_{m,n}$ refers to the hidden state produced by some preprocessing LSTM for the n^{th} token in the m^{th} sentence in document d, and $N_{d,m}$ is the number of tokens in the m^{th} sentence in d. $\mathbf{H}^q$ represents the hidden states produced by LSTM for the tokens in question and $\overline{\mathbf{h}}^q_{m,i}$ is the weighted sum of question hidden states $\mathbf{H}^q$.

when we try to assess whether the current token in the document is a good answer, we should check whether its surrounding tokens match the tokens in the question well. Based on this observation, we propose to split the document into sentences first and match each sentence with the question using an LSTM. On top of these token-level LSTMs for each sentence, we then build a sentence-level LSTM to aggregate the results.

Specifically, the single-pass reader model works as follows. Suppose the document $\mathbf{X}^d$ has been split into a sequence of sentences $(\mathbf{X}^d_1, \mathbf{X}^d_2, \ldots, \mathbf{X}^d_{M_d})$, where each $\mathbf{X}^d_m$ itself is a sequence of tokens. Given a particular $\mathbf{X}^d_m$, we can follow the same procedure as described in Section 3.2 but treat $\mathbf{X}^q$ as the document and $\mathbf{X}^d_m$ (the m^{th} sentence) as the question. After using the answer-LSTM to process these two sequences, we obtain the last hidden state of the answer-LSTM, which we denote with $\mathbf{h}^s_{m,N_{d,m}}$ (where $N_{d,m}$ is the number of tokens in the m^{th} sentence). This hidden state records the best answer token we can find from this sentence. Furthermore, we expect this hidden state to also indicate overall how well this sentence matches the question, or in other words, whether this sentence contains many tokens that are semantically similar to the tokens in the question. A sentence that better matches the question is more likely to contain the correct answer token.

Next, we treat these $\mathbf{h}^s_{m,N_{d,m}}$ as the input to another LSTM. This top-layer LSTM works at sentence level and aggregates the results from the sentences. What it tries to do is to remember answers from sentences that match the question well but forget answers extracted from poorly-matched sentences. The last hidden state $\mathbf{h}^t_{M_d}$ from this top-layer LSTM is used to predict the final answer through a softmax layer. This model is depicted in Figure 2.

Table 2. Some statistics of the CNN/Daily Mail dataset. The last row refers to the relabeled data by [10], where the entities in a document were re-labeled in order, such that the first entity is labeled as @entity1, the second entity as @entity2, etc., rather than being randomly labeled.

	CNN	Daily Mail
# Train	380,298	879,450
# Dev	3,924	64,835
# Test	3,198	53,182
avg # tokens per doc	762	813
avg # sentences per doc	33.4	32.6
avg # entities per doc	26.2	26.2
vocab size	118,497	208,045
# classes	405	415
# classes (relabeled)	145	156

4 Experiments

4.1 Data and Experiment Setup

We use the CNN/Daily Mail dataset created by [1] to perform evaluation. The documents are news articles from CNN/Daily Mail and the questions were created from the bullet points that came with the articles. Details of the data preparation, including how the entities were recognized and anonymized, can be found in the original paper [1]. Some statistics of the data set are shown in Table 2.

We consider the following baselines:

Attentive Reader: In this model proposed by [1], attention mechanism is used to compute a weighted sum of the document hidden states for the prediction.

Impatient Reader: The impatient reader is also developed by [1]. It sequentially accumulates the weighted sum of document states for each state in the question.

MemNNs: This is a machine comprehension model developed by [9] using Memory Networks.

Entity-Centric Classifier: This model proposed by [10] is based on human engineered features such as whether the entity occurs in the question, the frequency of the entity in the document, etc.

Attention Sum Reader: This is the attention sum reader model recently developed by [11]. It is a relatively simple model that uses attention mechanisms to directly pick the answer.

Stanford Attentive Reader: This model developed by [10] is an attentive reader with a different attention mechanism and experiment settings from [1].

DER Network: This is a model recently developed by [22] using dynamic entity representation.

EpiReader: This is a model developed by [7] combining the components of reasoner and extractor.

Table 3. Results on the CNN and the Daily Mail data sets with a single model. Note that the bottom section shows the performance of the models trained on the relabeled data sets.

Model	CNN		Daily Mail	
	dev	test	dev	test
Attentive Reader [1]	61.6	63.0	70.5	69.0
Impatient Reader [1]	61.8	63.8	69.0	68.0
MemNN [9]	63.4	66.8	N/A	N/A
Entity-Centric Classifier [10]	67.1	67.9	69.1	68.3
Attention Sum Reader [11]	68.6	69.5	75.0	73.9
Stanford Attentive Reader [10]	72.5	72.7	76.9	76.0
DER Network [22]	71.3	72.9	N/A	N/A
EpiReader [7]	73.4	74.0	N/A	N/A
AOA Reader [12]	73.1	74.4	N/A	N/A
ReasoNet [14]	72.9	74.7	77.6	76.6
BiDAF [15]	76.3	76.9	80.3	79.6
GA [13]	77.9	77.9	81.5	80.9
Multi-pass Reader (our model)	67.5	70.0	73.9	73.5
Single-pass Reader (our model)	73.3	74.3	77.7	76.7
Stanford Attentive Reader (on relabeled data) [10]	73.8	73.6	77.6	76.6
Single-pass Reader (on relabeled data) (our model)	75.5	76.6	79.4	78.6

AOAReader: This model [12] uses the structure of attention-over-attention for answer prediction.

ReasoNet: This model [14] considers multi-step reasoning by making use of reinforcement learning.

BiDAF: This model [15] uses bidirectional attention flow to build the interaction between the document and the question.

GA: This model [13] proposes the gated attention mechanism for solving the RC problem.

For all the experiments, we use accuracy as the evaluation metric, which is defined as the percentage of questions that are correctly answered. For our experiments, we use token embeddings trained on the CNN/Daily Mail corpus itself. We first use word2vec [23] embeddings to initialize the embeddings of known tokens. We then use CBOW to train 300-dimensional word embeddings on the CNN/Daily Mail data set and we do not update the embeddings when training our model. We use the ADAGRAD [24] method for optimization. The learning rate is set to be 0.01. The batch size is set to be 256. The dimension of all the hidden states is 150. The dropout ratio on the embedding layer is 0.3. The vocabulary size we used is 89291 for CNN and 153712 for Daily Mail. The number of classes for the softmax layer is 405 for CNN and 415 for Daily Mail. After relabeling by [10], the number of classes is set to be 145 for CNN and 156 for Daily Mail. For our Hierarchical Reader, processing one long document may

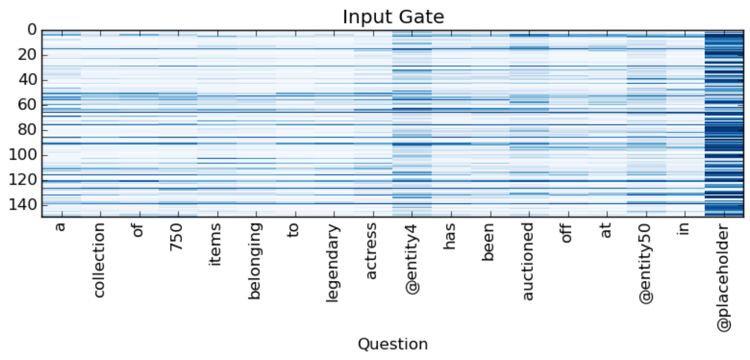


Fig. 3. The input gates of the answer-LSTM in the multi-pass reader for a document-question pair. @placeholer with its document attention representation has higher input gate values.

take more than 4G of memory. Due to the limitation of GPU memory, we adopt multi-threads/multi-machines training on CPUs.

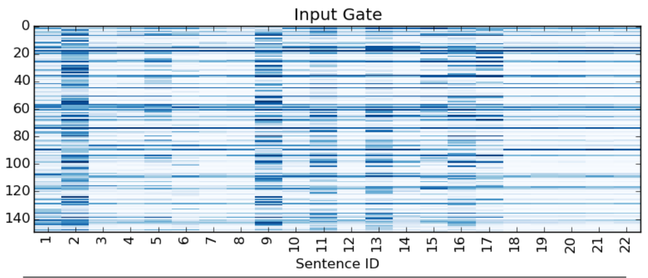
4.2 Main Results

Table 3 shows the performance of our two models and the various baselines on both the development set and the test set. We can observe the following: (1) Our Single-pass Reader outperforms the Multi-pass Reader. This shows that indeed the Single-pass Reader has a network design that is more suitable for the task. (2) After relabeling, our Single-pass Reader achieves a better performance. This shows that with fewer classes in the softmax layer, the method is more robust. (3) Our Single-pass Reader is competitive to the best model on the CNN/Daily Mail dataset.

4.3 Further Analyses

In this section, we conduct a few analyses on our two models in order to verify some hypotheses we have on how the two models work.

Some of our analyses are based on observing the values of the input gates and the forget gates because they indicate which tokens or sentences have contributed more to the final answers. In all the analyses below, we will show only the input gates. This is because we find that the input gates and the forget gates tend to be negatively correlated. In particular, using a random sample of these gate values, we find that the Pearson's correlation coefficients between the input gates and the forget gates are -0.319, -0.558 and -0.470 for the LSTMs in the multi-pass reader, the top-layer LSTMs and the bottom-layer a-LSTMs in the single-pass reader, respectively.



Question	a collection of 750 items belonging to legendary actress @entity4 has been auctioned off at @entity50 in @placeholder

ID	Context sentence
2	the collection , which includes works by some of the greatest artists of the 20th century , went under the hammer in @entity13 on march 31 , following a tour of @entity5 , @entity15 , @entity16 and @entity17 .
9	the 750 - piece collection , which fetched a total of $ 3.64 million , featured bronze sculptures , jewelry , and a number of decorative arts and paintings , which were sold at @entity50 auction house in @entity13 . ”

Fig. 4. The input gates of the top-layer LSTM in the single-pass reader for a document-question pair. Two sentences with high input gate values are also shown together with the question.

The Multi-pass Reader For the multi-pass reader, we would like to verify the following two hypotheses: (1) The final answer comes mostly from the tokens in the document that match the placeholder token `X` in the question. (2) The correct answer token in the document tends to have a higher attention weight when matched to token `X`.

For the first hypothesis, we show the values of the input gates of the answer-LSTM at every position of the question for one document-question pair in Figure 3. We can see that indeed the placeholder token (shown as `@placeholder` in the figure) has much higher input gate values compared with other tokens. We further compute the average input gate values for the placeholder token in all the questions and compare it with the average input gate values of other tokens. We find that the former has an average value of 0.654 with a standard deviation of 0.032 while the latter has an average value of 0.192 with a standard deviation of 0.103. This confirms that indeed the final answer found by the answer-LSTM comes mostly from the tokens matched to the placeholder.

To verify the second hypothesis, for each document-question pair, we rank the tokens in the document by their attention weights when matched to `X` in the question. We then check whether the correct answer token is within the top-5 of this ranked list. We find that out of the 3924 document-question pairs, 2037 pairs have the correct answer within the top-5 matching tokens to `X`. This is

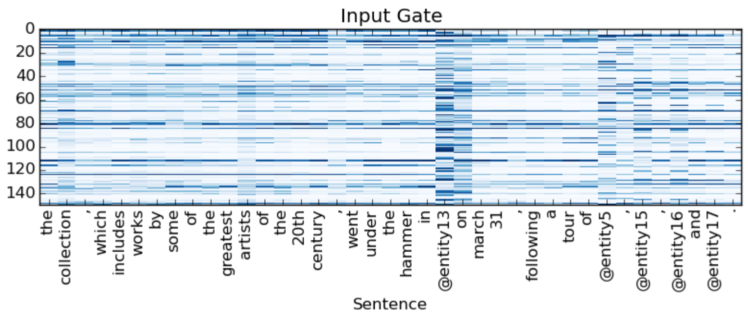


Fig. 5. The input gate values of the bottom-layer a-LSTM for a sentence. Here the correct answer token is `@entity13`.

much higher than random, and it shows that indeed the answer-LSTM picks up the correct answer by matching it to X through the attention mechanism.

Single-pass Reader

For the single-pass reader, we would like to verify the following hypotheses: (1) The top-layer LSTM is able to pick up sentences that semantically match the question better using its input gates. (2) The bottom-layer a-LSTMs are able to pick up the correct answer tokens using their input gates.

To verify the first hypothesis, we show the input gate values of the top-layer LSTM for one document-question pair in the top plot of Figure 4. We can see that a few sentences have received relatively high input gate values, including Sentence 2 and Sentence 3. A closer look at these two sentences and comparing them with the question show that indeed these sentences have better overlap with the question. To check whether this is generally the case, for each document-question pair, we rank the sentences from the document based on their cosine similarities with the question and pick the top one. We check whether this sentence also receives the highest average input gate value among all the sentences. We find that out of the 3924 document-question pairs, 2037 (51.9%) have the same top-ranked sentence by cosine similarity and by average input gate value. Although the percentage is not very high, it is much higher than random, and note that the cosine similarity is only a crude measure of the semantic similarity between a sentence and a question. Interestingly, we find that out of the 3924 document-question pairs, 3222 of them (82.1%) have the correct answer token inside the sentence that has the highest average input gate value. In contrast, a randomly selected sentence has a 21.9% chance on average to contain correct answer token. This shows that the top-layer LSTM is often able to pick up the sentence that contains the correct answer.

To verify the second hypothesis above, we compute the average input gate values of the correct answer token in the bottom-layer a-LSTMs of the Single-pass Reader. We compare this with the average gate values of the other tokens.

We find that the average input gate value for the correct answer token is 0.632 with a standard deviation of 0.041, while the average value for other tokens is 0.154 with a standard deviation of 0.093. We can see that indeed the correct answer token tends to receive higher input gate values. We also show the input gate values for one sentence in Figure 5.

5 Conclusions

We presented two LSTM-based neural network models for machine comprehension. By treating the task as a textual entailment problem, the first model, which we call the multi-pass reader, tries to match the question with the document using an LSTM with attention mechanisms. Based on some observations with the task, we further proposed a novel two-layer LSTM model, which we call the single-pass reader, where at the bottom layer we match each sentence in the document with the question and at the top layer we use an LSTM to combine the results. Experiments on a large CNN data set showed that both our models are competitive compared with a number of recent methods for machine comprehension, and our single-pass reader outperforms the multi-pass reader. By carefully examining the input gates and attention weights learned in our models, we also verified a few hypotheses about how the two models work.

Based on our analyses, we see a number of ways to further improve our work. First, our single-pass reader model can pick the sentence that contains the correct answer over 82% of the time, but its final accuracy is still below 76%. We therefore need to further strengthen the mechanism of identifying the answer token from a good matching sentence. Second, currently we split the document into non-overlapping sentences in the single-pass reader. However, sometimes the question may be derived from a few consecutive sentences. We therefore plan to look into more flexible ways of dividing the document into segments such that we can better match the question with any arbitrary subsequence of the document.

References

1. Hermann, K.M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., Blunsom, P.: Teaching machines to read and comprehend. In: Advances in Neural Information Processing Systems. (2015) 1684–1692
2. Narasimhan, K., Barzilay, R.: Machine comprehension with discourse relations. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing. (2015) 1253–1262
3. Sachan, M., Dubey, K., Xing, E., Richardson, M.: Learning answer-entailing structures for machine comprehension. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing. (2015) 239–249
4. Wang, H., Bansal, M., Gimpel, K., McAllester, D.: Machine comprehension with syntax, frames, and semantics. In: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing. (2015) 700–706

5. Smith, E., Greco, N., Bosnjak, M., Vlachos, A.: A strong lexical matching method for the machine comprehension test. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. (2015) 1693–1698
6. Yin, W., Ebert, S., Schütze, H.: Attention-based convolutional neural network for machine comprehension. arXiv preprint arXiv:1602.04341 (2016)
7. Trischler, A., Ye, Z., Yuan, X., Suleman, K.: Natural language comprehension with the epireader. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. (2016)
8. Rajpurkar, P., Zhang, J., Lopyrev, K., Liang, P.: SQuAD: 100,000+ questions for machine comprehension of text. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing. (2016)
9. Hill, F., Bordes, A., Chopra, S., Weston, J.: The Goldilocks principle: Reading children's books with explicit memory representations. In: Proceedings of the International Conference on Learning Representations. (2016)
10. Chen, D., Bolton, J., Manning, C.D.: A thorough examination of the CNN/Daily Mail reading comprehension task. In: Proceedings of the Conference on Association for Computational Linguistics. (2016)
11. Kadlec, R., Schmid, M., Bajgar, O., Kleindienst, J.: Text understanding with the attention sum reader network. In: Proceedings of the Conference on Association for Computational Linguistics. (2016)
12. Cui, Y., Chen, Z., Wei, S., Wang, S., Liu, T., Hu, G.: Attention-over-attention neural networks for reading comprehension. In: Proceedings of the Conference on Association for Computational Linguistics. (2017)
13. Dhingra, B., Liu, H., Yang, Z., Cohen, W.W., Salakhutdinov, R.: Gated-attention readers for text comprehension. In: Proceedings of the Conference on Association for Computational Linguistics. (2017)
14. Shen, Y., Huang, P.S., Gao, J., Chen, W.: Reasonet: Learning to stop reading in machine comprehension. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM (2017) 1047–1055
15. Seo, M., Kembhavi, A., Farhadi, A., Hajishirzi, H.: Bidirectional attention flow for machine comprehension. In: Proceedings of the International Conference on Learning Representations. (2017)
16. Richardson, M., Burges, C.J., Renshaw, E.: MCTest: A challenge dataset for the open-domain machine comprehension of text. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. (2013) 193–203
17. Lai, G., Xie, Q., Liu, H., Yang, Y., Hovy, E.: RACE: Large-scale reading comprehension dataset from examinations. Proceedings of the Conference on Empirical Methods in Natural Language Processing (2017)
18. Kočiský, T., Schwarz, J., Blunsom, P., Dyer, C., Hermann, K.M., Melis, G., Grefenstette, E.: The NarrativeQA reading comprehension challenge. Transactions of the Association for Computational Linguistics (2018)
19. Wang, S., Jiang, J.: Learning natural language inference with LSTM. In: Proceedings of the Conference on the North American Chapter of the Association for Computational Linguistics. (2016)
20. Bahdanau, D., Cho, H., Bengio, Y.: Neural machine translation by jointly learning to align and translate. In: Proceedings of the International Conference on Learning Representations. (2015)
21. Rush, A.M., Chopra, S., Weston, J.: A neural attention model for abstractive sentence summarization. Proceedings of the Conference on Empirical Methods in Natural Language Processing (2015)

22. Kobayashi, S., Tian, R., Okazaki, N., Inui, K.: Dynamic entity representation with max-pooling improves machine reading. Proceedings of the North American Chapter of the Association for Computational Linguistics (2016)
23. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. (2013) 3111–3119
24. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. The Journal of Machine Learning Research **12** (2011) 2121–2159