Iterative Co-Attention for Text Comprehension

Anonymous ACL submission

Abstract

Recently, attention mechanisms play a key role in end-to-end neural text comprehension tasks. Typically, these attention mechanisms are elaborate tailored to capture the complicated interactions between the question and the document. In this paper, we propose a novel dynamic memory-based co-attention network to tackle text comprehension tasks. Unlike previous models, our iterative co-attention mechanism can not only capture bidirectional attentions simultaneously, but also infer the relation between query, document and answer with multi-turn reasoning. Our model outperforms state-of-the-art baselines in the large machine comprehension benchmarks SQuAD.

1 Introduction

Text comprehension is a crucial task in both artificial intelligence and natural language processing, which requires that machine can read, understand, and answer questions about a text. Benefiting from the the rapid development of deep learning techniques (Goodfellow et al., 2015) and large-scale benchmarks (Hermann et al., 2015; Hill et al., 2015; Rajpurkar et al., 2016), the end-to-end neural based methods achieve promising results on text comprehension tasks, in which various neural attention mechanisms play a very important role. Attention mechanism (Bahdanau et al., 2014) allows a differentiable neural model to focus on attentive parts of context document which are relevant to the query.

Attention mechanisms in previous work typically can be roughly divided into two lines: memory-based and memoryless mechanisms.

Memory-based mechanisms The system can repeatedly re-read document or query in multiple turns to infer the answer, such as impatient attentive reader (Hermann et al., 2015), gated-attention (Dhingra et al., 2016) iterative alternating attention (Sordoni et al., 2016) , and ReasoNet (Shen et al., 2016). These models can be regarded as variants of memory networks (Kumar et al., 2015; Sukhbaatar et al., 2015), and repeatedly compute attentions between document and query with multi-layer network, also referred to multi-hop architecture. Intuitively, the memory-based attention mechanisms allows the reader to incrementally refine document or query representations. Since there are some queries whose answer cannot be inferred by single-turn reasoning, memory-based attentions generally produce a better answer.

However, the existing memory-based attention mechanisms just use uni-directional attention, query to document attention (Hermann et al., 2015) or document to query attention (Dhingra et al., 2016).

Memoryless mechanisms The memoryless mechanisms compute the attention weights once. Some early models compute unidirectional attention from query to document, in which the attention weights of different tokens in the document are computed according to their relevance to the query, such as Attentive Reader (Hermann et al., 2015), Attention Sum Reader (Kadlec et al., 2016). Recently, bidirectional attentions are proposed to fuse information from both of document and query, such as attention-over-attention (Cui et al., 2016), bidirectional attention (Seo et al., 2016), co-attention (Lu et al., 2016; Xiong et al., 2016).

These bidirectional attentions have demonstrated their superior performances consistently

than unidirectional attentions. However, these bidirectional attention mechanisms need be elaborate tailored to model their interactive and dependent manners. Besides, these models also cannot make multi-turn reasoning.

The model is able to encode co-dependent representations of the question and the document, ??

In this paper, we propose a novel iterative co-attention model for text comprehension tasks, which combines the advantages of bidirectional and memory-based attention mechanisms. The model first reads the document and the query using a recurrent neural network. Then, it deploys an iterative inference process to uncover the inferential links that exist between the query and the document. In each step, a word-level co-attention weights are computed from the question and document simultaneously according to the co-attention vector of previous step. The attentions of current step are computed based on these new co-attention weights, which is fed to the next step. This permits our model to reason about different parts of the query in a sequential way, based on the information that has been gathered previously from the document. After a fixed number of iterations, the model uses a summary of its inference process to predict the answer. Our model obtains state-of-the-art result on the large machine comprehension benchmarks SQuAD.

The contributions of this paper can be summarized as follows.

1. Our proposed model combines the advantages of bidirectional and memory-based attention mechanisms, which can not only compute the word-level co-attention weights from the question and document simultaneously, but also make multi-turn reasoning.

2. We define a memory-based co-attention mechanism and its corresponding updating strategy.

3. Our iterative co-attention mechanism can capture bidirectional attentions simultaneously, query-to-document and document-to-query, which provide complimentary information to each other.

2 Task Definition

Text comprehension (TC) is one type of question answering (QA) task. Given a question and a context document, a TC model need to answer the question depends on the understanding of the context document. Normally, the context document is assumed to contain the answer, and the correct answer span is a consecutive subphrase of the document.

There are two popular styles of benchmarks to evaluate the MC models. One consists of the Cloze-style questions, such as e CBT (Hill et al., 2015) and CNN (Hermann et al., 2015) corpora, in which a set of possible answers are provided. Another contains more challenging questions whose correct answers can be any sequence of tokens from the given text, such as Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016).

SQuAD were created by humans through crowdsourcing, which makes the dataset more realistic. Given these advantages of the SQuAD dataset, in this paper, we focus on this new dataset to study machine comprehension of text.

Formally, we can represent the SQuAD dataset as a set of tuples (Q, D, A), where $Q = (q_1, q_2, \cdots, q_m)$ is the question with m words, $D = (x_1, x_2, \cdots, x_n)$ is the context document(passage) with n words, and $A = x_{r_s:r_e}$ is the answer, r_s and r_e are the start and ending points. The MC task can be represented as estimating the conditional probability $P(A|Q, D)$. To do well on this task, a MC model need to comprehend the question, reason among the passage, and then identify the answer span.

3 Iterative Co-Attention for Text Comprehension

An overview of our architecture is in Figure xxx, which consists of threefour layers: (1) An encoding layer that encodes the question and the document using LSTMs. (2) A co-attention layer that tries to encode co-dependent representations of the question and the document. (3) An aggregation layer that compute query-aware representation of the document. (4) An output layer that uses two pointer network to select a set of words from the document as the answer.

The main difference between our model and the others lies in the second layer, in which we use co-attention. iterative co-attention to make the most of the information among document, query and their interactions.

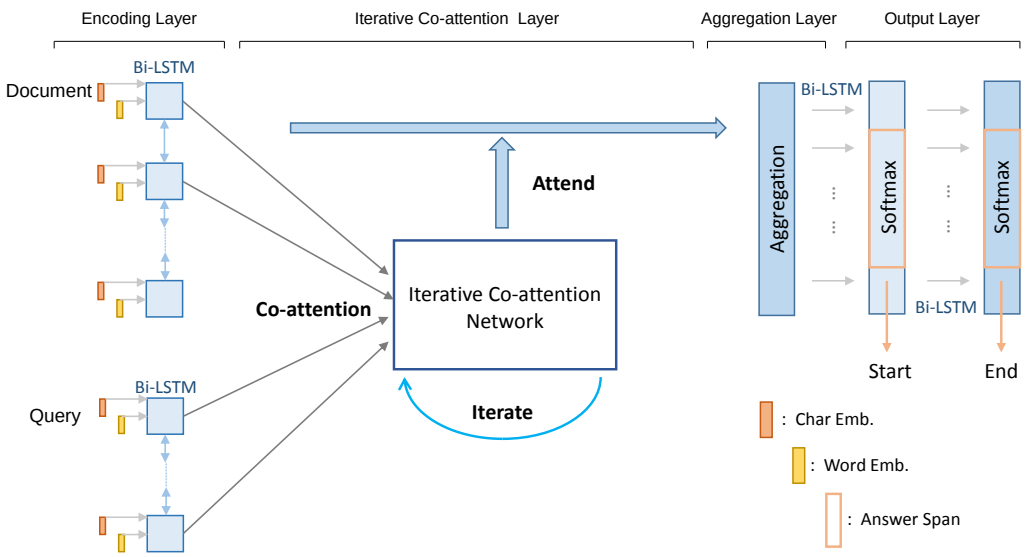

Figure 1: Iterative co-attention for text comprehension.

3.1 Document and Query Encoders

Let $Q = (q_1, q_2, \cdots, q_m)$ and $D = (x_1, x_2, \cdots, x_n)$ denote the words in the question and the document respectively. We first map each word to a continuous vector space. Here, we use pre-trained word vectors, GloVe (Pennington et al., 2014), to obtain the fixed word embedding of each word.

Then we use an bidirectional long short-term memory network (LSTM) to encode the document and question.

LSTM Long short-term memory network (LSTM) (Hochreiter and Schmidhuber, 1997) is a type of recurrent neural network (RNN) (Elman, 1990), and specifically addresses the issue of learning long-term dependencies.

While there are numerous LSTM variants, here we use the LSTM architecture used by (Jozefowicz et al., 2015), which is similar to the architecture of (Graves, 2013) but without peep-hole connections.

We define the LSTM *units* at each time step t to be a collection of d-dimension vectors: an *input gate* $\mathbf{i}_t$, a *forget gate* $\mathbf{f}_t$, an *output gate* $\mathbf{o}_t$, a *memory cell* $\mathbf{c}_t$ and a hidden state $\mathbf{h}_t$. The LSTM is precisely specified as follows.

$$\begin{bmatrix} \tilde{\mathbf{c}}_t \\ \mathbf{o}_t \\ \mathbf{i}_t \\ \mathbf{f}_t \end{bmatrix} = \begin{bmatrix} \tanh \\ \sigma \\ \sigma \\ \sigma \end{bmatrix} \left(\mathbf{W} \begin{bmatrix} \mathbf{x}_t \\ \mathbf{h}_{t-1} \end{bmatrix} + \mathbf{b} \right), \quad (1)$$

$$\mathbf{c}_t = \tilde{\mathbf{c}}_t \odot \mathbf{i}_t + \mathbf{c}_{t-1} \odot \mathbf{f}_t, \quad (2)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t), \quad (3)$$

where $\mathbf{x}_t \in \mathbb{R}^k$ is the input at the current time step; $\mathbf{W} \in \mathbb{R}^{4d \times k}$ and $\mathbf{b} \in \mathbb{R}^{4d}$ are parameters of affine transformation; σ denotes the logistic sigmoid function and $\odot$ denotes elementwise multiplication.

The update of each LSTM unit can be written precisely as follows:

$$\mathbf{h}_t = \text{LSTM}(\mathbf{h}_{t-1}, \mathbf{x}_t, \theta). \quad (4)$$

Here, the function $\text{LSTM}(\cdot)$ is a shorthand for Eq. (1-3), and θ represents all the parameters of LSTM.

In order to capture the phrasal information of word x_t, we use two LSTMs with forward and backward directions. Therefore, the encoding phase maps each word x_t to a contextual representation by concatenating of the forward and backward LSTM hidden states $\mathbf{h}_t = \overrightarrow{\mathbf{h}}_t \oplus \overleftarrow{\mathbf{h}}_t$, where $\oplus$ is concatenation operation.

Hence we obtain the question representation $\mathbf{Q} \in \mathbb{R}^{2d \times m}$ and the document representation $\mathbf{D} \in \mathbb{R}^{2d \times n}$ and . Each column vector in D and U can be regarded as phrasal representation of each word in them.

3.2 Co-Attention Layer

We propose a co-attention mechanism that attends to the question and document simultaneously. Dif-

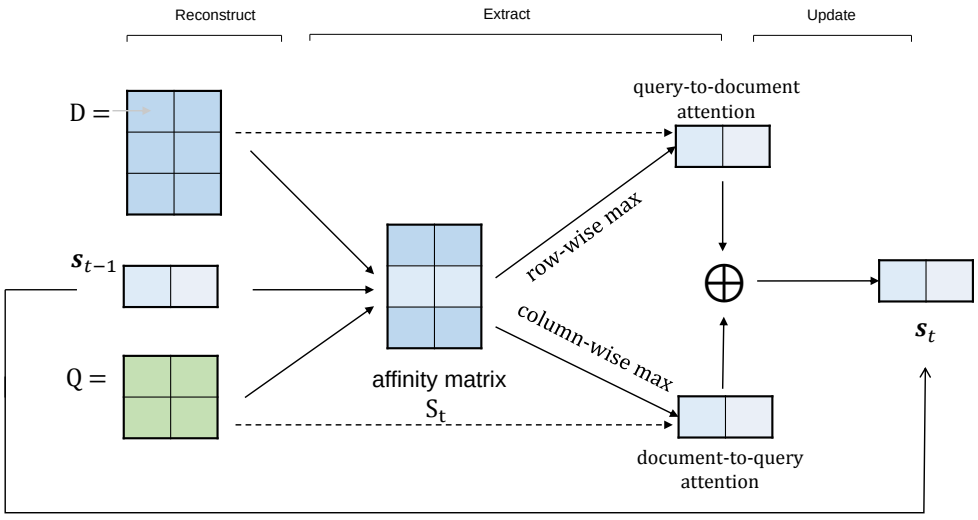

Figure 2: Update of iterative co-attention.

ferent with the co-attention in (Lu et al., 2016), our co-attention depends on history.

We first compute the affinity matrix S_t, which contains affinity scores corresponding to all pairs of document words and query words:

$$S_t = \mathbf{D}^\top \mathbf{diag}(\mathbf{W}_s \mathbf{s}_{t-1} + \mathbf{b}_s)\mathbf{Q} \in \mathbb{R}^{n \times m}, \quad (5)$$

where $\mathbf{s}_{t-1} \in \mathbb{R}^{2d}$ is the **co-attention vector** from both document and query in previous step, $\mathbf{W}_s \in \mathbb{R}^{2d \times 2d}$ and $\mathbf{b}_s in \in \mathbb{R}^{2d}$ are learnable parameters to control the impact of accumulative co-attention $\mathbf{s}_{t-1}$. The initial co-attention vector $\mathbf{s}_0$ is set to all one vector.

Specifically, we can set $\mathbf{W}_s = 0$ and $\mathbf{b}_s = \mathbf{1}$ to make the attention mechanism non-iterative and independent of $\mathbf{s}_{t-1}$. This is similar to the co-attention or bi-directional attention mechanisms used in (Seo et al., 2016; Lu et al., 2016; Xiong et al., 2016).

After computing this affinity matrix, we can compute the bidirectional attentions: **query-to-document attention** $A_t^{Q \to D}$ and **document-to-query attention** $A_t^{D \to Q}$.

$$A_t^{D \to Q} = \mathrm{softmax}(S_t^\top) \in \mathbb{R}^{m \times n}, \quad (6)$$

$$A_t^{Q \to D} = \mathrm{softmax}(S_t) \in \mathbb{R}^{n \times m}, \quad (7)$$

where $\mathrm{softmax}(\cdot)$ is column-wise normalized function.

The i-th column of document-to-query attention $A_t^{D \to A}$ is the attention weights across the query for the i-th word in the document, and $\sum_j (A_t^{D \to Q})_{ji} = 1$ for all i. Likewise, the j-th column of query-to-document attention $A_t^{Q \to D}$ is the attention weights across the document for the j-th word in the query, and $\sum_i (A_t^{Q \to D})_{ij} = 1$ for all j.

Next, we compute the two directional attention vectors both from document and query.

$$\mathbf{a}_t^{D \to Q} = \frac{1}{n} Q \sum_{i=1}^{n} (A_t^{D \to Q})_{:i} \in \mathbb{R}^{2d \times 1}, \quad (8)$$

$$\mathbf{a}_t^{Q \to D} = \frac{1}{m} D \sum_{j=1}^{m} (A_t^{Q \to D})_{:j} \in \mathbb{R}^{2d \times 1}, \quad (9)$$

where $\mathbf{a}_t^{D \to Q}$ and $\mathbf{a}_t^{Q \to D}$ denote the **document-to-query attentive vector** and **query-to-document attentive vector** respectively. Here, we just simple average operation to compute the attentive vector, but more sophisticated operations also can be adopted.

Update co-attention vector Then we integrate the two attention vectors into a co-attention vector, which also depends on its previous state. Here, we use LSTM to model their combination.

$$\mathbf{s}_t = \mathrm{LSTM}(\mathbf{s}_{t-1}, \mathbf{x}_t, \theta_p) \in \mathbb{R}^{2d \times 1}, \quad (10)$$

where $\mathbf{x}_t = \mathbf{a}_t^{D \to Q} \oplus \mathbf{a}_t^{Q \to D} \in \mathbb{R}^{4d \times 1}$ is the concatenation of two directional attention vectors.

3.3 Query-Aware Document Encoding Layer

After T steps, we get word-level document-to-context attention vectors

$$D^{D \to Q} = Q A_T^{Q \to D} \in \mathbb{R}^{2d \times n}. \quad (11)$$

Each column of $D^{D \to Q}$ is attentive query encoding, which is the encoding of the query in light of each word of the document.

Fusion Finally, the query-aware representation of i-th document word is

$$\tilde{D}_{:i} = f(D_{:i}, D^{Q \to D}_{:i}), \quad (12)$$

where $f(\cdot)$ is a fusion function that fuses the attentive query encoding and original document encoding. $f(\cdot)$ can be an arbitrary vector function, such as neural networks, concatenation $\oplus$ or element-wised multiplication $\odot$. Here, we use the element-wised multiplication $\odot$ to model the interactions between $D_{:i}$ and $\tilde{D}_{:i}$.

Finally, we get the fused query-aware representation of document $\tilde{D} = [\tilde{D}_{:1}, \cdots, \tilde{D}_{:n}] \in \mathbb{R}^{2d \times n}$.

3.4 Aggregation Layer

Following (Wang et al., 2016; Sordoni et al., 2016), we further aggregate the query-aware document representation by a bidirectional LSTM.

$$\mathbf{m}_i = \text{BLSTM}(\overrightarrow{\mathbf{m}}_{i-1}, \overleftarrow{\mathbf{m}}_{i+1}, \tilde{D}_{:i}, \theta_{ag}), \quad (13)$$

where $\mathbf{m}_i = [\overrightarrow{\mathbf{m}}_i; \overleftarrow{\mathbf{m}}_i]$ is the concatenation of two output vectors of two different directional LSTMs at position i.

The role of this layer is to capture the interaction among the document words conditioned on the query. This is different from the phrase embedding layer, which captures the interaction among context words independent of the query.

Thus, the final aggregation representation is defined as

$$M = [\mathbf{m}_1, \mathbf{m}_2, \cdots, \mathbf{m}_n] \in \mathbb{R}^{4d \times n}, \quad (14)$$

which is passed onto the output layer to predict the answer.

3.5 Output Layer

The output layer is task-specific. For the QA task of SQuAD(Rajpurkar et al., 2016), the answer is a sub-phrase of the context. Therefore, our output layer can be customized to predict the start and the end indices of the phrase in the context.

We use$p(r_s = i, r_e = j|D, Q)$ to represent the probability that the sub-phrase $x_{i:j}$ is the answer of query Q conditioned on document D. We decouple to joint probability by

$$p(r_s = i, r_e = j|D, Q)$$

$$= p(r_s = i|D, Q)p(s_e = j|r_s = i, D, Q). \quad (15)$$

We feed the aggregation vector of each word into the feed-forward neural network individually, calculate a value for each word, then normalize the values across the entire passage with softmax operation.

We denote $\mathbf{y}_s$ as the probability distribution of the start index over all the words in the context.

$$\mathbf{y}^s = \text{softmax}(M^\top \mathbf{w}_s) \in \mathbb{R}^m, \quad (16)$$

where $\mathbf{w}_s$ is weight vector.

Since the probability of the end index depends on the start index, we use another bidirectional LSTM layer to get M', which implicit models the information of start index, document and query.

We denote $\mathbf{y}_e$ as the probability distribution of the start index over all the words in the context.

$$\mathbf{y}^e = \text{softmax}(M'^\top \mathbf{w}_e) \in \mathbb{R}^m, \quad (17)$$

where $\mathbf{w}_e$ is weight vector.

The final answer span (k, l) with the maximum value of $\mathbf{y}^s_k \mathbf{y}^e_l$ is chosen, which can be computed in linear time with dynamic programming.

Training We define the training loss as the sum of the log probabilities of the true start and end indices by the predicted distributions, averaged over all examples:

$$L(\theta) = -\frac{1}{N} \sum_{i=1}^{N} \log \mathbf{y}^s_k + \log \mathbf{y}^e_l, \quad (18)$$

where k and l are the true start and end indices of the i-th example, θ represents all the parameters in the model.

4 Experiments

In this section, we present our experiment results and perform some analyses to better understand how our models works.

4.1 Dataset

We conduct our experiments on the Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016), which is a new and challenging reading comprehension dataset. Each in SQuAD is a single paragraph from 536 articles of Wikipedia, which cover a wide range of different topics. There are around 5 associated questions posed by crowdworkers for each document. The answer to

each question is a text segment from the corresponding document.

SQuAD is significantly larger than previous text comprehension datasets. In total, there are $23,215$ passages and $107,785$ questions, which are split into training set, development set and unpublished test set. The training set contains $87,599$ question-answer pairs, and development set contains $10,570$ question-answer pairs.

4.2 Settings

We first tokenize all the paragraphs and questions by a regular-expression-based word tokenizer (PTB Tokenizer). The resulting vocabulary contains 117K unique words. We use word embeddings from GloVe (Pennington et al., 2014) to initialize the model, and the embeddings for out-of-vocabulary words are set to zero. The word embeddings are fixed during the training of the model. The hidden state size (d) of the model is 100. For the iterative co-attention, we set the maximum number of iterations to 4. We train optimize the model using AdaDelta (Zeiler, 2012) optimizer, with a minibatch size of 30. We use a max sequence length of 600 and dropout to regularize our network during training (Srivastava et al., 2014). All models are implemented and trained with TensorFlow (Abadi et al., 2016).

The performance is measured by two metrics: exact match (EM) and word-level F1 score. The EM score calculates the percentage of exact string match between the predicted answer and a ground truth answer. The F1 score calculates the overlap between words in the predicted answer and a ground truth answer. Since that in the development set and the test set each document-query pair has around three ground truth answers, the EM and F1 scores are taken to be the maximum value across all ground truth answers. The overall metric is the average over all document-query pairs.

4.3 Results

Performance across iterations of co-attention A natural point of interesting is to analyze how much iterative co-attention helps performance of the model, we changes the number of iterations and compare their different performances on development set. To make an intuitive comparison, we use a baseline model of non-iterative attention by set $\mathbf{W}_s = 0$ and $\mathbf{b}_s = \mathbf{1}$ in Eq (5). The Comparison of results is shown in Table 1. We can see that the iterative and memory-based co-attention increases the performance greatly. The co-attention model with 3 iterations achieves the best performance. We also train an ensemble model consisting of 5 separately trained models with the same architecture and hyper-parameters, in which the number iterations is set to 3. The final ensemble model achieves an F1 score of 79.5, outperforming the single model by about 3.65%.

Iterations	Dev EM	Dev F1
Baseline	65.5	75.6
1	66.5	76.2
2	66.7	76.5
3	**66.9**	**76.7**
4	66.9	76.5
ensemble (3 iters)	70.8	79.5

Table 1: Performances across iterations of co-attention on development set.

	Dev EM	Dev F1
No D2Q attention	58.1	68.6
No Q2D attention	57.1	67.8
Single Model	66.9	76.7

Table 2: Model ablations.

Model Ablations To evaluate the importance of co-attention for text comprehension, we remove document-to-query (D2Q) attention $\mathbf{a}_t^{D\to Q}$ or query-to-document (Q2D) attention $\mathbf{a}_t^{Q\to D}$ in Eq(10) while updating the co-attention vectors.

Table 2 shows the performance of our model and its ablations on the SQuAD dev set. We can see that both attentions are critical for the final performance. Without D2Q attention, the performance drops more than 11.68.1% on F1 score. Without Q2D attention, the performance drops more than 10.68.9% on F1 score.

Performance across length We also investigate how the performance of the varies with respect to the length of document. As in shown in Figure 3, there is no notable performance degradation for longer documents and questions. This suggests that the iterative co-attention can find out the small relevant text segmentation while ignoring the rest of the unrelated document.

Performance across question types In order to gain insights into the performance across question types, we also break the results down by the first one or two words in the questions. As

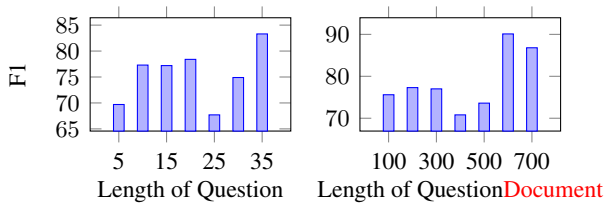

Figure 3: Performance across length.

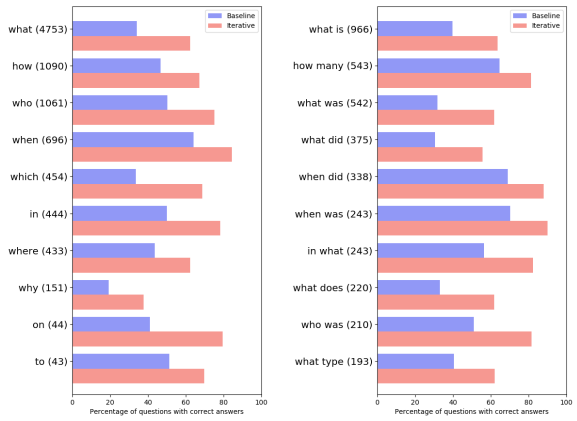

Figure 4: Performance across question types.

shown in Figure 4, our model outperforms the traditional baseline comfortably in every category. Like other models (Wang and Jiang, 2016; Xiong et al., 2016; Seo et al., 2016), our model also does well in "when" questions and is weak in the complex "why" questions.

Performance across question typesVenn diagram visualization We also compare our single model with baseline model in detail. Figure 5 shows a Venn diagram of the dev set questions correctly answered by the models. Our model is able to answer more than 81.5% of the questions correctly answered by the baseline.

Comparison with state-of-the-art methods We also compare our model with state-of-the-art methods and the baseline model on the SQuAD dataset[1]. The baseline model is based on logistic regression with carefully designed features (Rajpurkar et al., 2016) .

As shown in Table 3, our single-model outperforms most of state-of-the-art methods on the

[1]The offical SQuAD evaluation is hosted on CodaLab https://worksheets.codalab.org. The training and development sets are publicly available while the test set is blind.

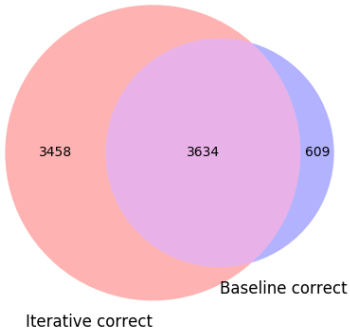

Figure 5: Performance across question type.Venn diagram visualization

leaderboard[2]. Our model achieves state-of-the-art performance at xxx% exact match and xxx% F1 on the test data.

5 Related Work

Recently, the bidirectional attentions have demonstrated their superior performances consistently than unidirectional attentions on text comprehension.

Cui et al. (2016) used a mechanism called attention-over-attention to explicitly calculate the weights between different individual document-level attentions, and get the final attention by computing the weighted sum of them.

Seo et al. (2016) proposed a bi-directional attention flow network, which used a bi-directional attention flow mechanism to achieve a query-aware context representation without early summarization.

Xiong et al. (2016) also used co-attention to learn the co-dependent representations of the query and the document in order to focus on relevant parts of both.

However, these models need carefully design the interactive and dependent manners of the bidirectional attentions. Besides, these models are computed in single-turn and memory-less, resulting in struggling for multi-turn reasoning. Although the memory-based models, such as gated-attention (Dhingra et al., 2016) and ReasoNet (Shen et al., 2016), can compute the attention in multi-turn, but the attentions in these models are unidirectional and not word-to-word level.

[2]https://rajpurkar.github.io/SQuAD-explorer

Model	Dev EM	Dev F1	Test EM	Test F1
Single model				
Baseline (Rajpurkar et al., 2016)	40.0	51.0	40.4	51.0
R-net (Microsoft Research Asia)*	65.9	75.2	71.3	79.7
RaSoR (Lee et al., 2016)	66.4	74.9	–	–
Multi-Perspective Matching (Wang et al., 2016)	–	–	68.9	77.8
Bidirectional Attention Flow (Seo et al., 2016)	–	–	68.0	77.3
Dynamic Chunk Reader (Yu et al., 2016)	62.5	71.2	62.5	71.0
Match-LSTM with Bi-Ans-Ptr (Wang and Jiang, 2016)	64.1	73.9	64.7	73.7
Dynamic Co-attention (Xiong et al., 2016)	65.4	75.6	66.2	75.9
Iterative Co-attention (**Ours**)	66.9	**76.7**	-	-
Human (Rajpurkar et al., 2016)	81.4	91.0	82.3	91.2

Table 3: Comparison with state-of-the-art methods. * indicates that the model used for submission is unpublished.

Different from the above models, our proposed attention mechanism combines the advantages of bidirectional word-to-word attention and multi-turn memory-based attention.

More recently, a similar idea of iteratively computing the bidirectional attentions were proposed in (Sordoni et al., 2016). Different to our model, their iterative way is to alternately compute the unidirectional attention: first computing attention over the query, then computing attention over the document, and repeating this process. Their attention is also sentence-to-word level and cannot find out the local alignment between query and document. Conversely, our iterative co-attention does not collapse the query or document into a single vector to compute the attention, and deploys an iterative co-attention mechanism that allows a fine-grained exploration of both the query and the document.

6 Conclusion

We proposed the, an end-to-end neural network architecture for question answering. The consists of a coattention encoder which learns co-dependent representations of the question and of the document, and a dynamic decoder which iteratively estimates the answer span. We showed that the iterative nature of the model allows it to recover from initial local maxima corresponding to incorrect predictions. On the dataset, the achieves the state of the art results at xxx% F1 with a single model and xxx% F1 with an ensemble. The significantly outperforms all other models.

References

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

D. Bahdanau, K. Cho, and Y. Bengio. 2014. Neural machine translation by jointly learning to align and translate. *ArXiv e-prints* .

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2016. Attention-over-attention neural networks for reading comprehension. *arXiv preprint arXiv:1607.04423* .

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

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science* 14(2):179–211.

Ian Goodfellow, Aaron Courville, and Yoshua Bengio. 2015. Deep learning. Book in preparation for MIT Press. http://goodfeli.github.io/dlbook/.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* .

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*. pages 1684–1692.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301* .

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

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of The 32nd International Conference on Machine Learning*.

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

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *arXiv preprint arXiv:1506.07285* .

Kenton Lee, Tom Kwiatkowski, Ankur Parikh, and Dipanjan Das. 2016. Learning recurrent span representations for extractive question answering. *arXiv preprint arXiv:1611.01436* .

Jiasen Lu, Jianwei Yang, Dhruv Batra, and Devi Parikh. 2016. Hierarchical question-image coattention for visual question answering. *arXiv preprint arXiv:1606.00061* .

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. *Proceedings of the Empiricial Methods in Natural Language Processing (EMNLP 2014)* 12:1532–1543.

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

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603* .

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2016. Reasonet: Learning to stop reading in machine comprehension. *arXiv preprint arXiv:1609.05284* .

Alessandro Sordoni, Phillip Bachman, and Yoshua Bengio. 2016. Iterative alternating neural attention for machine reading. *arXiv preprint arXiv:1606.02245* .

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. 2015. End-to-end memory networks. In *Advances in Neural Information Processing Systems*. pages 2431–2439.

Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-LSTM and answer pointer. *arXiv preprint arXiv:1608.07905* .

Zhiguo Wang, Haitao Mi, Wael Hamza, and Radu Florian. 2016. Multi-perspective context matching for machine comprehension. *arXiv preprint arXiv:1612.04211* .

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

Yang Yu, Wei Zhang, Kazi Hasan, Mo Yu, Bing Xiang, and Bowen Zhou. 2016. End-to-end answer chunk extraction and ranking for reading comprehension. *arXiv preprint arXiv:1610.09996* .

Matthew D Zeiler. 2012. Adadelta: An adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .