Attention-based Parallel Affinity Exploiting for Machine Comprehension

Abstract

Recently, attention mechanisms have shown great power in many NLP tasks, especially in Machine Comprehension (MC) Task, which needs to answer a query according to the corresponding document. And usually, these attention mechanisms depend on an affinity matrix which depicts the relevance between document and query to capture the most relevant parts. In this paper, we proposed a new attention-based Parallel Affinity Exploiting model to improve the computation of attentions as well as the query-aware document representation. First, we compute more parallel affinity matrices with no-sharing based on the document and query, or document itself, by using a difference loss. Then, we exploit these affinity matrices in different hierarchical ways, such as directly using a single one or affecting one with another, to make most of them so as to improve the performance of MC Task. Besides, we make further efforts to set up a parallel-architecture experiment proposal to test the effectiveness and substitutability of each single computation step through the whole attention computation flow. The effectiveness of parallel affinity matrices is proved by an ablation experiments. And, the empirical study on CNN/Daily Mail cloze-style datasets demonstrates the efficacy of our proposed model.

1 Introduction

In recent years, machine comprehension(MC) and question answering(QA) have been critical tasks in natural language processing which aims to teach machine to read, understand and answer the questions corresponding to the documents. With the release of large reading comprehension datasets, such as CNN/Daily Mail [Hermann *et al.*, 2015], CBT [Hill *et al.*, 2015] and SQuAD [Rajpurkar *et al.*, 2016] , many deep learning methods have been evaluated on these benchmarks and have gained pretty good performances. One of the key factors contributing to the advancement is neural attention mechanism [Bahdanau *et al.*, 2014], which has shown great power in many tasks.

The attention mechanisms used in previous work on MC Task are applied to capture the relevant areas within the two contexts, document and query, in order to improve the computation of the query-aware document representation which then be sent to the prediction progress afterwards. Although their purposes to gain key words in both contexts are the same, their methods to exploit the document, query and their relevance matrix are different and can be divided into three classes.

Attention and operation on it: In [Hermann *et al.*, 2015] , several architectures are introduced to use LSTMs with attentions to compute a document-query representation for candidate answer selection, such as DeepLSTM Reader. Attentive Reader [Hermann *et al.*, 2015], tries to get a query-aware document representation in which each word is weighted by it's attention on query. The Attention-Sum (AS) Reader [Kadlec *et al.*, 2016] with GRU networks [Cho *et al.*, 2014], and the Attention-over-Attention (AoA) Reader [Cui *et al.*, 2016] , both do some operations on attention, with sum-pointer in the former and attended attentions again in the latter.

Multi-hop architecture to utilize attention: Memory Networks (MemNets) was propose in [Weston *et al.*, 2014; Kumar *et al.*, 2015; Sukhbaatar *et al.*, 2015], where the information of query is aggregated by its attentions with document in each step. In addition, Iterative Attentive Reader [Sordoni *et al.*, 2016] and Gated-attention (GA) Reader [Dhingra *et al.*, 2016], also do iterative utilization to improve their reasoning capability. All of these models try to maximize the utilization of the information of document and query by aggregating their attentions iteratively.

Explore more attentions on affinity matrix: Recently, more forms of attentions based on the affinity matrix are proposed by BIDAF [Seo *et al.*, 2016] , which aims to explore more query-aware information on affinity matrix to pass to follow-up fusion layer as input to next prediction progress.

In this paper, we propose a new attention-based Parallel Affinity Model, which computes more affinity matrices based on the document and query, or document itself as well, so as improve the performance of MC Task. After the computation of these parallel affinity matrices, whose diversity is controlled by a difference loss, hierarchical exploitations on them were done, such as directly or affecting one with another, in order to make the most of them as to gain better attentions which will improve the query-aware document representa-

tion in turn. By the way, we further compare several compositional operations on these parallel affinity matrices and their extracted attention vectors. And motivated by this parallel architecture, we make further efforts to set up a parallel-architecture experiment proposal to test the effectiveness and substitutability of each single computation step through the whole attention computation flow.

The contributions of this paper lie in three folds:

1. We proposed a new attention-based Parallel Affinity Exploiting model, which computes multi-parallel affinity matrices based on the document and query, or document itself as well, in different hierarchical ways and can maximize the parallel exploitation on them in order to gain better and diverse attentions which are the key to the performance of MC Task.
2. We do further study to make most of these affinity matrices and their corresponding extracted attention vectors, by trying different compositional operations on them as ablations experiments. And we apply parallel affinity methods to the cloze-style test on CNN/Daily Mail dataset, which demonstrate the efficacy of our proposed model.
3. Motivated by this parallel architecture, we make further efforts to set up a parallel-architecture experiment proposal to test the effectiveness and substitutability of each single computation step through the whole attention computation flow.

2 Task Description

2.1 Cloze Test

The CNN/Daily Mail corpus are two large reading comprehension datasets, which are generated from news articles on website and the script is provided[1] or may directly got the generated datasets from the website[2]. The documents are given by the full articles with short, bullet-point summary statements accompanied, while the queries are generated by replacing a named entity within each article summary with an anonymous placeholder token.

Formally, we present these datasets as a set of tuples: (Q, D, C, A), where $Q = (q_1, q_2, \cdots, q_m)$ is the context of question with m words, $D = (x_1, x_2, \cdots, x_n)$ is the context of document with n words, C is the set of candidate answers and $A = a \in C$ is the answer, which is usually a single word. For each query, a placeholder token is substituted for the real answer a. Statistics on the datasets are reported in Table 1.

3 Attention-based Parallel Affinity Expoliting

Our machine comprehension model is depicted in Figure 1, it consists of these three layers: (1) An encoding layer that encodes the question and the document using LSTMs [Hochreiter and Schmidhuber, 1997]. (2) A parallel affinity computation and exploiting layer, which first computes more parallel affinity matrices between the two contexts or the document itself, and then exploit on them by different ways such as single and parallel to gain different information for follow-up computation, and finally make a fusion of them to pass to final layer. (3) An aggregation and output layer that attend the previous vectors to the document, aggregates them and predict the answer.

The difference from other models lies in the second layer, in which we compute more parallel affinity matrices, whose diversity is controlled by a difference loss, to depict the score between document and query or document itself, and exploit on them in different ways such as directly using a single one and affecting one with another, so that they can provide complimentary information to each other, and finally fusion the information they carried to enrich the query-aware document representation and improve the performance of prediction layer in turn.

	CNN	Daily Mail
# Train	380,298	879,450
# Valid	3,924	64,835
# Test	3,198	53,182
Max # length	2,000	2,000
Vocabulary	118,497	208,045

Table 1: Statistics of CNN and Daily Mail.

3.1 Bi-LSTM encoding

Let $Q = (q_1, q_2, \cdots, q_m)$ denote the token embeddings of the document, and $D = (x_1, x_2, \cdots, x_n)$ denote the token embeddings of the query. We first map each word to a continuous vector space to obtain the fixed word embedding by using pre-trained word vectors, GloVe [Pennington *et al.*, 2014] . And then we use Bi-LSTM to get further contextual embedding of each context.

LSTM

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.

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

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

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

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:

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

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

[1] https://github.com/deepmind/rc-data.

[2] http://cs.nyu.edu/ kcho/DMQA/.

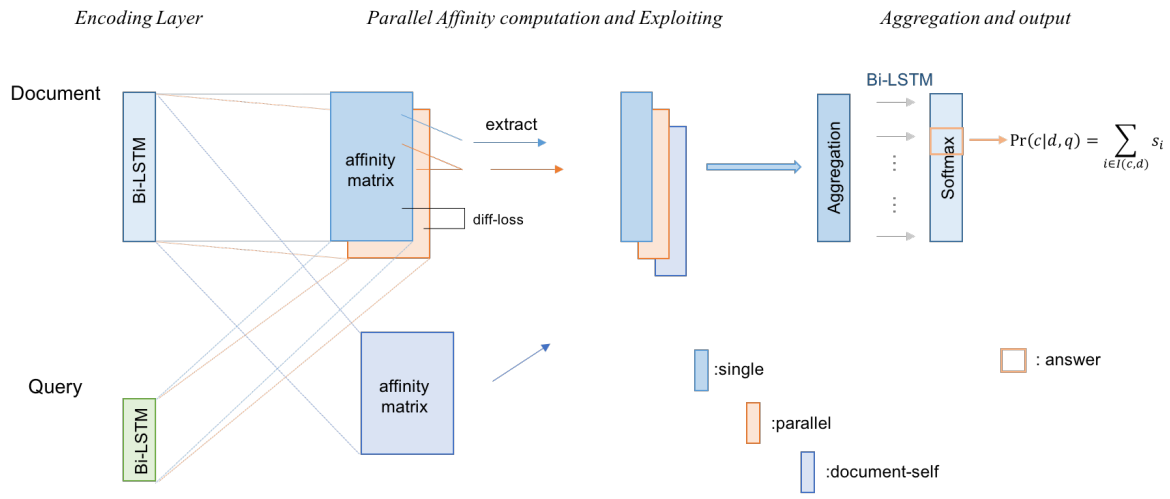

Figure 1: Overview of Parallel Affinity Exploiting Model.

Bi-LSTM

In order to get further contextual embedding of each context, bi-directional LSTMs (Bi-LSTM) are adopted to processes the sequence in both forward and backward directions to produce two sequences, which are concatenated at the output:

$$\text{Bi-LSTM}(x) = (\overrightarrow{\mathbf{h}}_1 \oplus \overleftarrow{\mathbf{h}}_1, ..., \overrightarrow{\mathbf{h}}_n \oplus \overleftarrow{\mathbf{h}}_n) \tag{5}$$

where Bi-LSTM denotes the final contextual embedding of context x by concatenating each forward state $\overrightarrow{\mathbf{h}}_t$ and backward state $\overleftarrow{\mathbf{h}}_t$ at the t-th word in given x.

3.2 Parallel affinity computation and exploitation

In this layer, we'll use the contextual embeddings of the two contexts from the previous layer to compute parallel affinity matrices, do exploitations on them in different ways, and fusion the information they carried to next layer. This takes three steps: first, we need to parallel compute multi-different affinity matrices between document and query, or document itself, with the diversity of these matrices ensured by adding a different-loss L to the final task loss as well as the different exploitation on them; then we exploit on these matrices in different ways such as directly using a single one and affecting one with another before sending them to the attention computation flow; and finally, we fusion the results of these distinct computations to gain attentions which will be taken to compute query-aware document representations in the next layer.

computation for affinity matrix

First, we compute multi-different affinity matrices $A \in \mathbb{R}^{n \times m}$ according to each pair of document $D \in \mathbb{R}^{n \times 2d}$ and query $Q \in \mathbb{R}^{m \times 2d}$ by:

$$\mathbf{A}_{ij} = f(\mathbf{D}_{i:}, \mathbf{Q}_{j:}) \tag{6}$$

where f is a trainable scalar function that encodes the similarity between its two input vectors, $\mathbf{D}_{i:}$ is i-th row vector of $\mathbf{D}$, and $\mathbf{Q}_{j:}$ is j-th row vector of $\mathbf{Q}$. Here, we choose linear function for f as below:

$$f(\mathbf{d}, \mathbf{q}) = \mathbf{w}_{(\mathbf{A})}^{\top}[\mathbf{d}; \mathbf{q}] + \mathbf{b}, \tag{7}$$

where d, q are each one word in D, Q, $\mathbf{w}_{(\mathbf{A})} \in \mathbb{R}^{4d}$ is a trainable weight vector, $\mathbf{b}$ is the bias, $\circ$ is element-wise multiplication, and $[;]$ is vector concatenation across row.

Likewise, we can get more parallel affinity matrices eg. $(A^1, A^2, ...)$ with no parameters sharing. Meanwhile, we apply a difference loss $\mathbf{L}_{\text{diff}}$ on these matrices to further ensure the diversity of them:

$$L_{\text{difference}} = \left\| \mathbf{A}^{1\top} \mathbf{A}^2 \right\|_F^2, \tag{8}$$

where $\| \cdot \|_F^2$ is the squared Frobenius norm.

Here, the difference loss $\mathbf{L}_{\text{diff}}$ encourages the affinity matrices to capture different aspects of the information between D and Q via the soft subspace orthogonality constraint.

single affinity for attention flow

With this affinity matrix, we compute a attention which is supposed to signify which words in document are most relevant to the query by using the softmax function. The attention weight for each context weight is:

$$\mathbf{a}_t = \text{softmax}(\mathbf{A}_{t:}) \in R^m \tag{9}$$

where t means the t-th words in context and a_t represent its weights on the corresponding query words. Then attention for each query-attended document word is computed by:

$$\tilde{\mathbf{U}}_{t:} = \sum_j \mathbf{a}_{tj} \mathbf{Q}_{j:} \tag{10}$$

where t means the t-th words in context and $\tilde{\mathbf{U}}_t$ is the attention computed for query-aware document representation in the next step. So $\tilde{\mathbf{U}} \in \mathbb{R}^{n \times 2d}$ involves the information that query attended to document.

parallel affinity with exploitation
With parallel affinity matrices, we can exploit both of them by affecting one of them with another one to further finetune the latter, so as to use it for another computation flow, thus they can provide complimentary information for each other. The exploitation on them can be as follows:

$$\tilde{\mathbf{A}} = \mathbf{A}^1 \odot \mathrm{func}(\mathbf{A}^2) \tag{11}$$

where $\mathbf{A}^1, \mathbf{A}^2$ are two no-sharing affinity matrices, $\odot$ means the operators that composite these two together, and func provides a rectification function.

Then, $\tilde{\mathbf{A}}$ will be sent to another computation flow like before in Eq. (9-10). Here, we adopt * as $\odot$, and $\mathrm{sigmoid}$ as func in a simple way. And more research needs to be conducted in the future.

document self-attention affinity
Besides query-to-document attention, we can also compute a document self attention here so as to improve the utilization of the document, which is usually omitted by other models for MC Task. And, this is convenient to do so due to our parallel architecture. First, we compute a special affinity matrix like before:

$$\mathbf{A}_{ij} = f(\mathbf{D}_{i:}, \mathbf{D}_{j:}) \tag{12}$$

Then, similarly, an attention computation flow for $\mathbf{D}$ itself is computed as follows:

$$\mathbf{a}_t = \mathrm{softmax}(\mathbf{A}_{t:}) \in R^n \tag{13}$$

$$\tilde{\mathbf{U}}_{t:} = \sum_j \mathbf{a}_{tj} \mathbf{D}_{j:} \tag{14}$$

Fusion with compositional functions
After we get diverse attentions with parallel different affinity matrices, compositional functions need to be done to further exploit the information from these vectors. Generally, these extracted vectors carry weights information on each word of document, either query-aware or document-self refining. Thus, the final output of $\mathbf{U}$ is computed as:

$$\tilde{\mathbf{U}} = \mathbf{U}^1 \odot \mathbf{U}^2 \odot etc.... \tag{15}$$

where $\odot$ means the operators that composite these attentions together such as +, *, linear function or simply, concatenate to pass to next layer. And $U^1, U^2, etc....$ represent diverse attentions extracted from different affinity matrices, along with query or document.

Later we'll do an ablation study on these compositional functions since this is a crucial step for the entire model.

3.3 Aggregation and output

Finally, we need to do an aggregation step to encode the query-aware representations of document and then send them to a special output layer for the task.

Let $\tilde{\mathbf{U}} \in \mathbb{R}^{n \times 2d}$ be the intermediate output attention, pass it to encode the final document representation:

$$\tilde{\mathbf{D}} = [\mathbf{D}; \mathbf{D} * \tilde{\mathbf{U}}] \in \mathbb{R}^{n \times 2d}. \tag{16}$$

We use an another layer of bi-directional LSTM then, with the output size of d for each direction, to capture the interaction in $\tilde{\mathbf{D}}$ between document and query. Hence we obtain a $\mathbf{D}' \in \mathbb{R}^{n \times 2d}$, with which we do prediction to the queries in the following:

$$\mathbf{s} = \mathrm{softmax}(\mathbf{w}_{(\mathbf{s})}^\top [\tilde{\mathbf{D}}; \mathbf{D}']), \tag{17}$$

$$P(w|\mathrm{D}, \mathrm{Q}) = \sum_{i \in I(w, \mathrm{D})} s_i, \;\; w \in V \tag{18}$$

where vectors $\mathbf{s}$ defines the probability distributions over the $|\mathbf{D}|$ tokens in the document, and $\mathbf{I}(\mathbf{w}, \mathbf{D})$ indicate the positions that word w appears in the docuemnt $\mathbf{D}$.

Training
As the training objectives, we define the training loss as the sum of the log probabilities of the correct answer by the predicted distributions:

$$L_{\mathrm{task}}(\theta) = -\frac{1}{N} \sum_{i=1}^{N} \log p(x), \;\; x \in C \tag{19}$$

where θ is the set of all trained parameters, N is the size of dataset and $\mathbf{C}$ means all candidate answers.

Also, former $L_{\mathrm{difference}}$ should be added as:

$$L = L_{\mathrm{task}} + \alpha L_{\mathrm{difference}} \tag{20}$$

where α is a scalar weight for $L_{\mathrm{difference}}$ and need to be tuned as a hyper parameter.

4 Experiment and Discussion

4.1 datasets

We evaluate our model on cloze-style test using CNN/Daily Mail datasets. These two are generated from articles of the popular CNN and Daily Mail websites, consisting of 380k/4k/3k and 879k/65k/53k (train/dev/test) examples. Each pair of document and query is an article and an incomplete sentence with missing word. These sentences are all extracted from human-written summary of the articles and the missing word is always an named entity, anonymized with random IDs. To be mentioned, we extract nearly 10% of the origin CNN dataset as a mini dataset for further study.

settings
The model architecture used in this task is fully depicted in Figure 1. During the encoding layer, every document and query are tokenized by a regular-expression-based word tokenizer (PTB Tokenizer), and each token is embedded with a 100-dimension pre-trained Global Vectors(GloVe)[Pennington *et al.*, 2014] along with trained character-level embeddings [Kim, 2014]. The hidden state size is set to 200 for all recurrent units and linear layers and the initial learning rate was 5×10^{-4}. We used stochastic gradient descent with the AdaDelta Optimizer [Zeiler, 2012] with mini batch-size set to 32 running with multi-GPUs. All models are implemented and trained with TensorFlow [Abadi *et al.*, 2016].

Model	CNN		DailyMail	
	val	test	val	test
Attentive Reader [Hermann *et al.*, 2015]	61.6	63.0	70.5	69.0
MemNN [Hill *et al.*, 2015]	63.4	66.8	-	-
AS Reader [Kadlec *et al.*, 2016]	68.6	69.5	75.0	73.9
DER Network [Kobayashi *et al.*, 2016]	71.3	72.9	-	-
Iterative Attention [Sordoni *et al.*, 2016]	72.6	73.3	-	-
EpiReader [Trischler *et al.*, 2016]	73.4	74.0	-	-
Stanford AR [Chen *et al.*, 2016]	73.8	73.6	77.6	76.6
GAReader [Dhingra *et al.*, 2016]	73.0	73.8	76.7	75.7
AoA Reader [Cui *et al.*, 2016]	73.1	74.4	-	-
ReasoNet [Shen *et al.*, 2016]	72.9	74.7	77.6	76.6
PAE(Ours)	73.7	74.2	77.2	76.4
MemNN* [Hill *et al.*, 2015]	66.2	69.4	-	-
ASReader* [Kadlec *et al.*, 2016]	73.9	75.4	78.7	77.7
Iterative Attention* [Sordoni *et al.*, 2016]	74.5	75.7	-	-
GA Reader* [Dhingra *et al.*, 2016]	76.4	77.4	79.1	78.1
Stanford AR* [Chen *et al.*, 2016]	77.2	77.6	80.2	79.2

Table 2: Results on CNN/DailyMail datasets. Ensemble methods(marked with *) are also included for completeness.

Model	CNN	
	Val	Test
Baseline	67.8	68.6
single-affinity exploit	66.9	68.5
paralle-affinity explot	**69.0**	**70.4**

Table 3: Comparison of exploiting on one single affinity matrix and multi parallel affinity matrices.

4.2 ablations study

Table 3 shows the results of performance between our model and its ablations on our mini CNN dataset. The baseline is the model which only compute one attention vector from the relevance matrix, while single-affinity exploit model tries to exploit more attentions as we described before from one single affinity matrix. And our model, parallel-affinity exploit model, trying to extract different information on both single and parallel no-sharing affinity matrices and compute parallel computation attention flow using them, gains nearly 2% improvement comparing with single-affinity exploit model. So this comparison demonstrate the capability of these parallel affinity matrices to further exploit among the document, query and the affinity matrices themselves through different ways.

Operation Functions	CNN	
	Val	Test
Sum	68.6	69.6
Linear	66.3	67.3
Concatenate	**69.0**	**70.4**

Table 4: Performances by using different compositional functions.

In table 4 , we do research on compositional functions used in part 3.2. We can see from the table that the + and concatenate operator both have excellent improvement for attentions aggregation. One way to explain is that + and concatenate may improve the performance as they can independently utilize the information of diverse attentions from multi-way computations. Conversely, linear function tries to mix these attention vectors which are in different representation space. Also, * operator is still bad due to the same reason and so we omit it here.

4.3 results and discussion

The performance of our model with previously published on CNN/Daily Mail datasets are shown as Table 2, and our model achieves competitive results.

discussion

In contrast to other competitive models, our attention-based Parallel Affinity Exploiting Model (PAE Model) puts more emphasis on exploiting parallel affinity matrices, whose differences are controlled by a difference loss to extract different attention vectors instead of complicated operations on a same relevance matrix. By doing so, less conflicts are supposed to happen due to the different immediate representation space. By the way, the operations that we adopt to perform on these matrices are hierarchical and can provide complimentary information to each other. Meanwhile, benefiting from this parallel architecture, one more document self-attention can be taken in as well, for example.

5 A parallel-architecture experiment proposal for evaluating operations through the whole attention computation

Previously, attentions used to be evaluated and visualized only by its final weighted aggregation of words from query to documents, instead of through the immediate function it can provided. Sometimes, the computation of neural network may loss interpretability and exceeds our expectation. Also, there exists no means to measure the effectiveness and substitutivity of every single computing operation for those man-made design of attentions, like AoA Reader [Cui *et al.*, 2016] and BIDAF [Seo *et al.*, 2016].

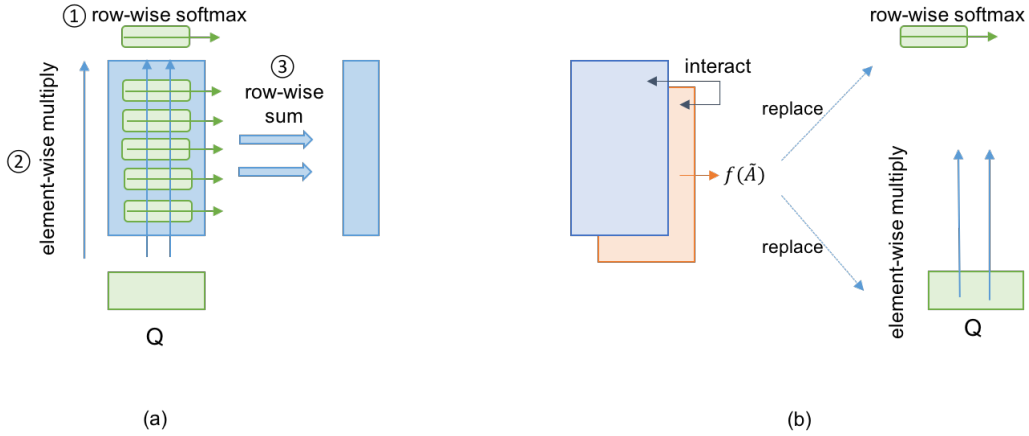

Figure 2: Evaluating on every single operation through the whole attention computation with parallel affinity. (a) is the decomposition of one attention computation. (b) means using parallel affinity matrix to replace each single step.

5.1 Evaluating on every single operation with parallel affinity

Motivated by the parallel affinity architecture we mentioned before, we can make further efforts to use this to serve for evaluating the effectiveness or substitutivity of each single computation step through the whole attention computation. What we need to do is first to compute parallel affinity matrices like before, and then alternate the corresponding single computing operation with one of these matrices.

Take the most popular attention computing way as example, shown in picture 2 above. We can divide the whole process into 3 steps : (1) a row-wise softmax function on A; (2) a element-wise multiply with query; (3) a row-wise sum function cross A.

evaluation on softmax

Review that the softmax function, in Equation 9, can be expanded as follows:

$$\sigma(a_t)_i = \frac{e^{A_{t:i}}}{\Sigma_{j=1}^{m} e^{A_{t:j}}} \tag{21}$$

where the σ represents softmax function with input as a_t, A means the affinity matrix we mentioned before.

Since this step is going to retune itself, we can directly multiply it with another parallel affinity matrix which is put into a sigmoid function. So we can just adjust the origin step to:

$$\mathbf{a}_t = \mathbf{A}_{t:} * \text{sigmoid}(\tilde{\mathbf{A}}_{t:}) \in R^m \tag{22}$$

where sigmoid means a sigmoid function and $\tilde{A}$ is an another parallel affinity matrix.

evaluation on query multiply

The same evaluation on the query-multiply(element-wise) is:

$$\tilde{\mathbf{a}}_t = \mathbf{a}_{t:} * \text{sigmoid}(\tilde{\mathbf{A}}_{t:}) \in R^m \tag{23}$$

Model	CNN	
	Val	Test
w/o softmax	-61.0	-61.6
w/o query×	-39.5	-39.3
para-rep softmax	-2.0	-1.3
para-rep query×	-1.0	+0.3

Table 5: Evaluation on single step with parallel affinity as alternatives.

5.2 Sample results and discussion

The evaluation results are depicted in Table 5. And we can see that by replacing with parallel affinity matrix, it seems more meaningful than directly removing which makes no sense.

discussion

From the results, at least two points can be summarized:

1. It seems that we can use this to do evaluation on each step of different attention computation flow, which may help to design new attention mechanism with complicated progress.
2. There are still much information we can exploiting on affinity matrices, both in the beginning or middle of the computation.

More attention mechanisms like $max_{row} + softmax + h\times$ in BiDAF and AoA etc., can be evaluated for future work.

6 Conclusion

In this paper, we proposed a novel attention-based Parallel Affinity Exploiting Model (PAE Model), which puts more emphasis on exploitation on multi parallel affinity matrices between two contexts, or document itself as well, including directly using a single one and retuning one with another hierarchically. And we do experiments and ablations study on CNN/Daily Mail cloze-style datasets. Further efforts are made to serve parallel-architecture to evaluate on each single step through the whole attention computation flow.

References

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

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

[Chen *et al.*, 2016] Danqi Chen, Jason Bolton, and Christopher D Manning. A thorough examination of the cnn/daily mail reading comprehension task. *arXiv preprint arXiv:1606.02858*, 2016.

[Cho *et al.*, 2014] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of EMNLP*, 2014.

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

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

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

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

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

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

[Kim, 2014] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[Kobayashi *et al.*, 2016] Sosuke Kobayashi, Ran Tian, Naoaki Okazaki, and Kentaro Inui. Dynamic entity representation with max-pooling improves machine reading. In *Proceedings of NAACL-HLT*, pages 850–855, 2016.

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

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

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

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

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

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

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

[Trischler *et al.*, 2016] Adam Trischler, Zheng Ye, Xingdi Yuan, and Kaheer Suleman. Natural language comprehension with the epireader. *arXiv preprint arXiv:1606.02270*, 2016.

[Weston *et al.*, 2014] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. *CoRR*, abs/1410.3916, 2014.

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