Coattention-Based Multi-Perspective Matching Network for Machine Comprehension

Qiujiang Jin **Bowei Ma**
Institute of Computational and Mathematical Engineering
Stanford University
Stanford, CA 94305
{qiujiang, boweima}@stanford.edu
Codalab username: QiujiangJin, boweima
Codalab worksheet: cs224n-QiujiangJin
Codalab submission group: c224n-JQJandBWM

Abstract

Machine comprehension, the task for machine to understand the contextual meaning of a passage, is a central yet challenging problem in natural language processing. In this paper, we focus on the question answering task based on the SQuAD dataset and construct a system utilizes the power of coattention mechanism and multi-perspective context matching to formulate a powerful question-paragraph co-dependent encoder in order to aggregate the matching vectors from multiple perspectives to predict the answer span to the question. Experiment result on the test set of SQuAD shows that our model achieves a decent prediction result.

1 Introduction

Reading comprehension is as the level of understanding of a text/message. This understanding comes from the interaction between the words that are written, and how they trigger knowledge outside the text/message. Usually reading comprehension contains the process of reading the context, processing the words or sentences, and then extract conceptual meaning of the whole document. The procedure is known to be a convoluted system involving semantic processing (encoding the meaning of a word and relate it to similar words), as well as structural and phonemic recognition and the processing of sentence and word structure.

Endowing machines to achieve such capacity is coveted goal for natural language processing. The task of Machine Comprehension (MC) is to enable machine to understand a given paragraph and then answer questions related to the paragraph. In recent years, several work has collected various datasets to tackle such task. However, the limited size of the previous generated dataset prevents researchers from building end-to-end deep neural network models, and the state-of-the-art performances are still dominated by the methods highly relying on hand-crafted features. Such situation was tremendously changed as Rajpurkar et al. (2016) [1] developed the Stanford Question Answering dataset (SQuAD). Comparing with other datasets, SQuAD is more challenging and promising since the volume of the dataset is over 100 thousands of question-paragraph-answer triplets, which is a reasonable amount for any end-to-end deep neural network models to fit, and also since the answer could be an arbitrary span span within the passage instead of a limited set of multiple choices.

In this work, we focus on the SQuAD dataset and implemented an end-to-end deep neural network model containing a multi-perspective matching scheme incorporated with coattention to identify the answer span by constructing a coattention encoder which matches the context of each word in the paragraph with the question from multiple perspectives. Experimental result on the test set of SQuAD shows that our model achieves a good training result.

In following parts, we start with a discussion of task definition and related work in Section 2, followed by the details of our deep neural network model in Section 3. Then the numerical experiment results in Section 4.

2 Question Answering Mechanics

2.1 Task Definition

Generally, a Question Answering (QA) system instance involves a question, a paragraph containing the answer, and the correct answer span within the passage. For the SQuAD dataset, such triplet is represented by the following examples:

Table 1: Example from the SQuAD dataset

Question: Because such oaths are in violation of the First Amendment, they're what?
Paragraph: The required beliefs of these clauses include belief in a Supreme Being and belief in a future state of rewards and punishments. (Tennessee Constitution Article IX, Section 2 is one such example.) Some of these same states specify that the oath of office include the words "so help me God." In some cases these beliefs (or oaths) were historically required of jurors and witnesses in court. At one time, such restrictions were allowed under the doctrine of states' rights; today they are deemed to be *in violation of the federal First Amendment*, as applied to the states via the 14th amendment, and hence **unconstitutional and unenforceable**.
Answer: unconstitutional and unenforceable

Mathematically, one sample from the SQuAD dataset could be formulated by a triplet (Q, P, A), where $Q = (x_1^Q, x_2^Q, \cdots, x_m^Q)$ denote the sequence of word vectors corresponding to the word in the question with length m, $P = (x_1^P, x_2^P, \cdots, x_n^P)$ denote the sequence of word vectors corresponding to the word in the paragraph with length n, and $A = (a_s, a_e)$ denote the answer span such that a_s is the index of the starting position of the answer and a_e is the index of the ending position of the answer. Then, the question-answering task could be be represented by estimating the conditional probability $\Pr(A|Q, P)$ and predicting answer for testing data by

$$A^* = \underset{A \in \mathcal{A}(P)}{\arg\max} \Pr(A|Q, P) \tag{1}$$

where $\mathcal{A}(P)$ is a set of possible answer spans generated by some metric function of answer prediction. In this work, we make a simple Bayes assumption that the prediction of starting and ending index are independent, which simplifies the prediction mechanism as

$$A^* = \underset{1 \le a_s \le a_e \le n}{\arg\max} \Pr(a_s|Q, P) \Pr(a_e|Q, P) \tag{2}$$

2.2 Related Work

Traditional approaches to question answering typically involve rule-based algorithms or linear classifiers over hand-engineered feature sets. Wang et al. (2015) [2] described a statistical model utilizing syntactic features such as part of speech tags and dependency parses, as well as frame semantic features.

Wang et al. (2015) described a statistical model using frame semantic features as well as syntactic features such as part of speech tags and dependency parses. For the SQUAD dataset, the original paper from Rajpurkar et al. (2016) [1] implemented a linear model with sparse features based on n-grams and part-of-speech tags present in the question and the candidate answer.

However, since the emerge of SQuAD dataset, various neural attention models have been proposed to tackle the QA task. Wang et al. (2016) [3] proposed a multi-perspective context matching model to match question embeddings with paragraph embeddings and then aggregate the matching vectors to make prediction. Xiong et al. (2016) [4]

proposed an encoding-decoding mechanism utilizing coattention after encoding questions and paragraphs and then use the coattention summary to train a dynamic Bi-LSTM to predict the answer span. In this work, we constructed a more complex model utilizing the merits of the above formulations to complete the QA task.

3 Multi-Perspective Coattention Network

In this section, we illustrate the detailed formulation of our Multi-perspective Coattention Network (MPCN). Figure 1 shows the architecture of our MPCN model.

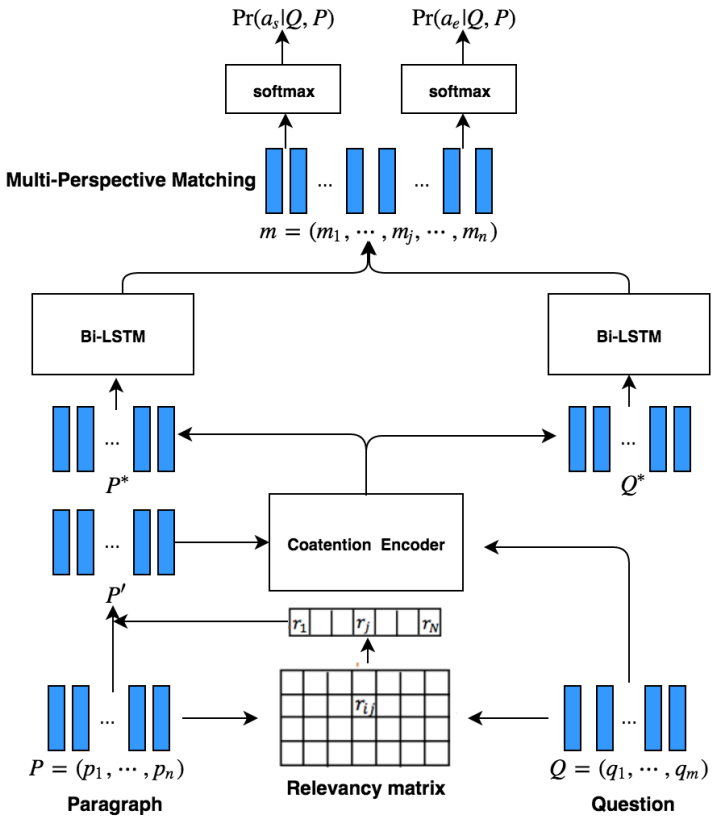


Figure 1: Architecture for Multi-perspective Coattention Network

3.1 Paragraph And Question Coattention Encoder

Given the sequence of word vectors P, Q, the word embeddings is a pre-trained set of vectors with GloVe. We feed each word into a Long Short-Term Memory (Hochreiter & Schmidhuber, 1997) [5]. That is, we encoding the question as: $q_t = \text{LSTM}(q_{t-1}, x_t^Q)$ and we define the encoding matrix as $Q = [q_1, q_2, \cdots, q_m] \in \mathbb{R}^{l \times m}$, where l is the hidden state size of the LSTM.

The paragraph embeddings are computed with the same LSTM to share representation power: $p_t = \text{LSTM}(p_{t-1}, x_t^P)$ and the resulting encoding matrix $P = [p_1, p_2, \cdots, p_n] \in \mathbb{R}^{l \times n}$.

Notice that in most cases, only a small portion of the paragraph is relevant to the corresponding question. For instance, the only needed information in the example in Table 1 is the last sentence in the paragraph. Hence, using a filter to aggregate the influence of question onto the paragraph should be beneficial for our learning purpose.

Inspired by Wang et al.(2016) [3], we compute the relevancy degree $r_{i,j}$ between each word pair $q_i \in Q$ and $p_j \in P$ by calculating the cosine similarity

$$r_{i,j} = \frac{q_i^T p_j}{\|q_i\|\|p_j\|}\mathbb{R}$$

Then, for each word $p_j \in P$, we compute the relevancy of p_j and Q as $r_j = \max_{i\in\{1,\cdots,m\}} r_{i,j}$ and filter each word by $p'_j = r_j p_j$ and the resulting filtered paragraph encoding as $P' = [p'_1, p'_2, \cdots, p'_n]$.

Then we use a coattention mechanism that attends to the question and paragraph in the same time, similar to the coattention mechanism described in (Xiong et al., 2016) [4], which compute the affinity matrix $L = (P')^T Q$ and normalized row-wise and column-wise respectively by softmax to produce attention weight matrices A^Q and A^P. Mathematically, we compute

$$A^Q = \text{softmax}(L) \in \mathbb{R}^{n\times m}, \quad A^P = \text{softmax}(L^T) \in \mathbb{R}^{m\times n}$$

Next, the attention summary C^Q of paragraph corresponding to each word in the question is computed as $C^Q = PA^Q$, and similarly we also compute the attention summary C^P of paragraph corresponding to each word in the question is computed as $C^P = QA^P$.

Then we concatenate the columns of C^Q and Q to form a coattention matrix Q^* representation for question Q, where each column q_j^* can be represented as $q_j^* = [q_j; c_j^Q]$ as a co-dependent representation of the paragraph, as the attention context. Similar procedure is carried out for the paragraph which results a coattention matrix P^* for paragraph P where each column $p_j^* = [p'_j; c_j^P]$. Note that the symbol $[a; b]$ is used as the concatenation of vectors a and b.

The last step is to incorporate contextual information into the representation of each time step in the passage and the question via a bi-directional LSTM. That is, we run a Bi-LSTM to encode contextual embedding for each question word:

$$\overrightarrow{h}_i^Q = \overrightarrow{\text{LSTM}}(\overrightarrow{h}_{i-1}^Q, q_i^*), \quad i = 1, \cdots, m \tag{3}$$

$$\overleftarrow{h}_i^Q = \overleftarrow{\text{LSTM}}(\overrightarrow{h}_{i+1}^Q, q_i^*), \quad i = m, \cdots, 1 \tag{4}$$

Also, the same Bi-LSTM is utilized to encode the paragraph:

$$\overrightarrow{h}_i^P = \overrightarrow{\text{LSTM}}(\overrightarrow{h}_{i-1}^P, p_i^*), \quad i = 1, \cdots, n \tag{5}$$

$$\overleftarrow{h}_i^P = \overleftarrow{\text{LSTM}}(\overrightarrow{h}_{i+1}^P, p_i^*), \quad i = n, \cdots, 1 \tag{6}$$

3.2 Multi-Perspective Context Matching

The goal of multi-perspective context matching is to compare each contextual embedding of the passage with the question with multi-perspectives. As suggested in (Wang et. al) [3], we first define a comparison metric function

$$w = f_w(v_1, v_2; W) \in \mathbb{R}^d \tag{7}$$

where d is the number of perspectives, $v_1, v_2 \in \mathbb{R}^l$ and $W \in \mathbb{R}^{d\times l}$ is a trainable parameter matrix. In this way, each component $w_k \in m$ represents a matching value from the kth perspective, and is calculated by the cosine similarity between the two weight vectors

$$z_1 = W_k \circ v_1 \tag{8}$$

$$z_2 = W_k \circ v_2 \tag{9}$$

$$w_k = \frac{z_1^T z_2}{\|z_1\|\|z_2\|} \tag{10}$$

where the symbol $\circ$ represents the Hadamard product of vectors, and W_k is the k-th row of W. Then, three strategies of comparing contextual embeddings are conducted:

- Full-Matching: for each position in the paragraph, the paragraph embedding of the such position is compared to the first forward and backward representation of the entire question (i.e., the first and last embedding vectors of the question)

$$\overrightarrow{w}_j^{full} = f_w(\overrightarrow{h}_j^P, \overrightarrow{h}_m^Q; W_1) \tag{11}$$

$$\overleftarrow{w}_j^{full} = f_w(\overleftarrow{h}_j^P, \overleftarrow{h}_1^Q; W_2) \tag{12}$$

The reason for utilizing such comparison is to match the information in the paragraph on the left or right with respect to the entire question. For instance, in the example given in Table 1, only the left context information is needed to match the entire question.

- Maxpooling-Matching: for each position in the paragraph, the paragraph embedding of the such position is compared to every forward and backward embedding of the question, and then we take the maximum result:

$$\overrightarrow{w}_j^{max} = \max_{i=\{1,\cdots,m\}} f_w(\overrightarrow{h}_j^P, \overrightarrow{h}_i^Q; W_3) \tag{13}$$

$$\overleftarrow{w}_j^{max} = \max_{i=\{1,\cdots,m\}} f_w(\overleftarrow{h}_j^P, \overleftarrow{h}_i^Q; W_4) \tag{14}$$

The reason for utilizing such comparison is to match the most important and relevant information in the paragraph with respect to the entire question. For instance, in the example given in Table 1, only the left context information is needed to match the entire question.

- Meanpooling-Matching: similar to maxpooling-matching, for each position in the paragraph, the paragraph embedding of the such position is compared to every forward and backward embedding of the question, and then we take the average value:

$$\overrightarrow{w}_j^{mean} = \frac{1}{m}\sum_{i=1}^{m} f_w(\overrightarrow{h}_j^P, \overrightarrow{h}_i^Q; W_6) \tag{15}$$

$$\overleftarrow{w}_j^{mean} = \frac{1}{m}\sum_{i=1}^{m} f_w(\overleftarrow{h}_j^P, \overleftarrow{h}_i^Q; W_6) \tag{16}$$

The reason for taking the average matching metric is to try avoiding information loss and capturing the overall meaning of the question as well as incorporate the meaning with the paragraph.

Finally, we construct a matching vector for each position in the paragraph by concatenating all the previous computed matching vectors. Hence, for position j in the paragraph the final matching vector is

$$w_j = [\overrightarrow{w}_j^{full}; \overleftarrow{w}_j^{full}; \overrightarrow{w}_j^{max}; \overleftarrow{w}_j^{max}; \overrightarrow{w}_j^{mean}; \overleftarrow{w}_j^{mean}] \in \mathbb{R}^{6d} \tag{17}$$

3.3 Aggregation Decoder

Now we aggregate the matching vectors so that each position (time step) of the paragraph can interact with its surrounding positions by applying a Bi-LSTM as

$$u_j = \text{Bi-LSTM}(u_{j-1}, u_{j+1}, w_j) \tag{18}$$

Given that we have computed a fairly complex network for question and paragraph encoding, to avoid overfitting we just use two different simple one-layer feed-forward neural network to predict the probability distribution of $\Pr(a_s|Q,P)$ and $\Pr(a_e|Q,P)$ by normalize the logit results from the neural network by softmax operatiion.

4 Experiments

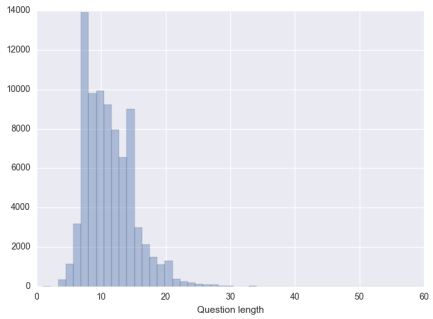
4.1 Experiment Settings

We evaluate our model using the SQuAD dataset with 81381 training examples and 4284 validation examples. We use pre-trained GloVe word vector embeddings (Manning et al., 2014) [6] with embedding size of 100. As suggested by the initial paper (Rajpurkar et al., 2016) [1], we employ Exact Match (EM) and F1 score as evaluation metrics.

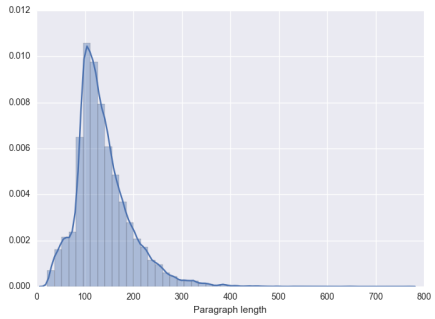
Due to the limitation of time and computing power, we set the size of hidden state to be 70, and the number of matching perspectives to be 30. We initialized the learning rate to be 0.01, and apply exponential decay of learning rate with ratio $\frac{\sqrt{5}-1}{2}$ after each epoch.

Figure 2 illustrates the length distributions of training set. For complexity consideration, we set the maximum length of question to be 30 and maximum length to be 300. More importantly, over 99.9% answer spans lay in the range of $(0, 300)$ so it is fairly safe to ignore the examples exceed the above limit during training.

To train the model, we minimize the average cross entropy loss of the starting and ending indices and use the ADAM optimizer (Kingma and Ba, 2014) [7] to update parameters.



(a) Distribution of question length



(b) Distribution of paragraph length

Figure 2: Length distributions of training data set (with estimated kernel density distribution)

4.2 Experiment Results

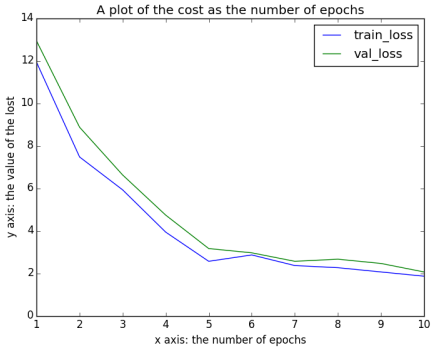
Table 2 summarizes the performance of our model. Further analysis of experiment are carried out in the following sections.

Table 2: Results on the SQuAD dataset

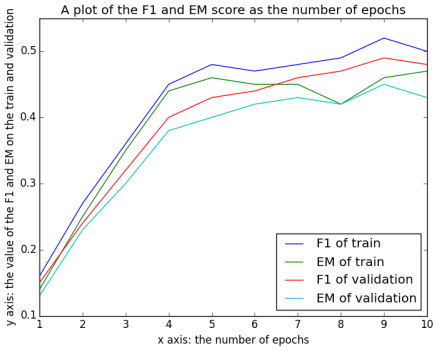
Dataset \ Evalution Metric	F1	EM
Train	58.7	49.6
Development	49.5	36.3
Test	49.9	37.4

4.3 Result Analysis

To better understand the behavior of our MPCN model, we conduct some analysis of the result of training process and analysis of errors on the dev set.



(a) Average training loss versus epochs



(b) Evaluation scores versus epochs

Figure 3: Training performance versus epochs

4.3.1 Training Performance Analysis

Figure 3 shows the training performance versus epochs, and from the plots we may notice that:

- The model learns well during the first 5 epochs. The loss drops form around 12 to around 3. The F1 and EM score on the train set and validation set rise to around 48 and 43, respectively.
- However, after epoch 5 the model seems to learn very slowly and those scores seem to not improving a lot even though the model is not overfitting.
- One possible explanation is that the small size of the hidden state and matching perspectives are impeding the learning ability of our model. The implementation of our model may not reflect the deeper semantic info of the paragraph with respect to the question.

4.3.2 Error Analysis

Table 3 summarizes the performance of several different question types for randomly selected 1000 examples in the dev set. We can notice that

- the performance for questions whose answer spans location, time, name are much higher than the others. One possible reason is that for those kind of questions usually the answer lies just a few words apart from the exact keywords of the question, and such temporal expression patterns are easier to detect by the use of full-matching vectors and the coattention scheme.

Table 3: Performance for selected different question types

Question \ Evalution Metric	F1	EM
what year	79.6	78.0
who	67.6	60.3
where	65.4	62.7
what is	63.2	55.6
how long	58.2	52.1
how many	55.3	51.5
why	32.1	25.2
what has	29.4	20.7
how did	21.3	16.5

- the performance for reasoning ”who”, ”how did” questions whose answer is usually much longer, needs comprehensive understanding of the context and also much further from the question keywords are much lower than the others. This may due to the use of maxpooling-match vectors which draws too much attention on the words which is the most similar to question keywords instead of understanding the context itself, and the lack of effectiveness of meanpooling-match vectors.
- Furthermore, Table 4 summarizes the exact accuracy of prediction of starting and ending indices. We may notice that our model has done a pretty good job on predicting the starting point of the answer, but has barely learnt the ability to decide where to stop. Such phenomenon may be the result of the coattention scheme putting too less attention on the actual ending point of answers and trying to predict an acceptable but more ”complete” answer instead.

Table 4: Accuracy of answer span prediction

Dataset \ Accuracy	Start index	End index
Train	70.4%	30.6%
Development	68.7%	26.3%

5 Conclusion

In this work we formulate a coattention-based multi-perspective matching model to realize the question answering task and get some proper result. The model contains a coattention encoder which learns the co-dependent representation of question and paragraph and a matching scheme for incorporating information of question into the paragraph from multiple perspectives. We have realized the complexity of machine comprehension and yet there is still many aspects for us to improve the system, such as a more careful training with dropout and gradient clipping, or more exploration of hyperparameter tuning by the cost of validation set and even revise the architecture of our matching scheme and representation of answer. Our model has done well on some specific types questions, but it still needs to be perfected to tackle questions which need more comprehensive understanding.

6 Contribution

- Qiujiang Jin: implementation of the model, literature review
- Bowei Ma: implementation of the model, project report writeup, result analysis

References

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

[2] Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. 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*, (Volume 2: Short Papers):700–706, 2015.

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

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

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

[6] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. *In ACL (System Demonstrations)*, pages 55–60, 2014.

[7] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.