Filter-Context Dynamic Coattention Networks for Question Answering

Yangxin Zhong
Stanford University
yangxin@stanford.edu

Peng Yuan
Stanford University
pengy@stanford.edu

Jian Huang
Stanford University
jhuang33@stanford.edu

Abstract

Question Answering (QA) is one of the most challenging and crucial tasks in Natural Language Processing (NLP) that has a wide range of applications in various domains, such as information retrieval and entity extraction. Traditional methods involve linguistically based NLP techniques, and recent researchers apply Deep Learning on this task and have achieved promising result. In this paper, we combined Dynamic Coattention Network (DCN) [1] and bilateral multi-perspective matching (BiMPM) model [2], achieved an F1 score of 63.8% and exact match (EM) of 52.3% on test set.

1 Introduction and related work

Question Answering (QA) is one of the most challenging and crucial tasks in natural language processing (NLP) that has a wide range of applications in various domains, such as information retrieval and entity extraction. In particular, given a paragraph that represents a series of facts, how can we have machines automatically answer a question that is inferred on these facts?

Traditionally, most of the researches of QA use a pipeline of linguistically based NLP techniques, such as parsing, part-of-speech tagging and co-reference resolution [3]. Recently, with rapid developments of deep learning, neural network based models have shown promising results for QA tasks [1]. However, lack of large-scale high quality datasets limits the training of deep neural networks. Previous datasets for QA task tend to be high in quality due to human annotation, but small in size [4]. Although researchers have developed large-scale datasets through semi-automated techniques [5, 6], they differ from human annotated datasets in the types of reasoning required to answer the questions [7].

To address the need for a large and high-quality reading comprehension dataset, Rajpurkar et al [8] released the Stanford Question Answering dataset (SQuAD), which consists of questions posed by crowdworkers on a set of Wikipedia articles, where the answers to every question is a segment of text from the corresponding reading paragraph. SQuAD is almost two orders of magnitude larger than previous manually labeled datasets, and thus is very suitable for deep neural network models to train.

Xiong et al [1] introduced the Dynamic Coattention Network (DCN), an end-to-end neural network for question answering, which consists of a coattentive encoder that captures interactions between the question and the document, as well as a dynamic pointing decoder that alternates between estimating the start and end of the answer span. Wang et al [2] proposed a bilateral multi-perspective matching (BiMPM) model, which first encodes two sentences with a bidirectional Long Short-Term Memory Network (BiLSTM), and then matched the two encoded sentences in two directions. In this paper, we propose to combine these two models, and our model obtains an F1 score of 63.8% and exact match (EM) 52.3%.

2 Dataset

The dataset we use for our project is the Stanford Question Answering Dataset (SQuAD), which consists of 107,785 question-answer pairs, along with a context paragraph [8]. The context paragraphs were extracted from 536 articles on Wikipedia by crowdworkers. The answer to every question is a segment of text, or span, from the corresponding reading passage [8].

3 Problem definition

Our task is to answer questions in SQuAD: given a context paragraph P and a question Q, find an answer span (s, e) for the question so that $P[s{:}e]$ gives the corresponding answer, where s is the answer start position and e is the answer end position. The performance of our model is evaluated by two metrics: F1 score and exact match (EM).

4 Dynamic coattention model with filter-context encoder

The overview of our model is shown in Figure 1. Our model combines part of the bilateral multi-perspective matching model [2] (filter-context encoder) with the entire dynamic coattention networks model [1] (coattention encoder and dynamic pointing decoder).

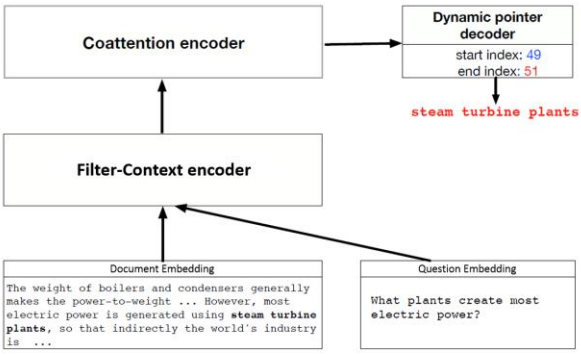

Figure 1: Model overview.

4.1 Filter-context encoder (component A)

Figure 2 shows the architecture of the filter-context encoder. Given a pair of question q and passage p, the filter-context encoder produces a new pair of word representations Q and D through the following three layers.

4.1.1 Word representation layer

This layer represents each word in question and passage with a d-dimension vector. We employ word embedding, which is pre-trained with GloVe [9], to construct the d-dimension vector. The

output of this layer is word vector sequences for question $q_{1:M}$ and passage $p_{1:N}$.

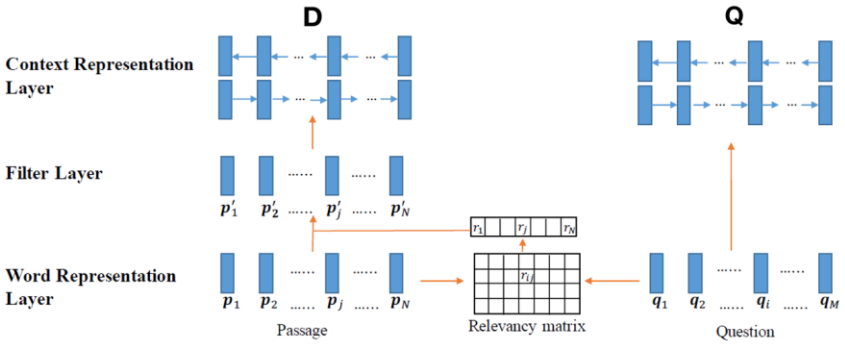

Figure 2: Filter-context encoder [2].

4.1.2 Filter layer

This layer filters out the redundant information that is not relevant to the question from the passage. First, it calculates a relevancy degree r_j for each p_j: it computes a relevancy degree r_{ij} for each pair of word vectors q_i and p_j , using cosine similarity $r_{ij} = \frac{q_i^T p_j}{\|q_i\| \cdot \|p_j\|}$; then $r_j = \max_i r_{ij}$. Second, it filters the word vector by $p'_j = r_j \cdot p_j$ and pass p'_j to next layer. The motivation here is a word more relevant to any word in question should have more information for answering the question and should be considered more.

4.1.3 Context representation layer

This layer incorporates the contextual information into the representation of each time step in passage and question. It employs two different bidirectional LSTM [10] (BiLSTM).

$$\vec{h}_i^q = \overrightarrow{LSTM}^q\left(\vec{h}_{i-1}^q, q_i\right) \in \mathbb{R}^{l/2}, \qquad \overleftarrow{h}_i^q = \overleftarrow{LSTM}^q\left(\overleftarrow{h}_{i+1}^q, q_i\right) \in \mathbb{R}^{l/2}$$

$$\vec{h}_j^p = \overrightarrow{LSTM}^p\left(\vec{h}_{j-1}^p, p_j\right) \in \mathbb{R}^{l/2}, \qquad \overleftarrow{h}_j^p = \overleftarrow{LSTM}^p\left(\overleftarrow{h}_{j+1}^p, p_j\right) \in \mathbb{R}^{l/2}$$

The output vectors $Q \in \mathbb{R}^{l \times M}$ and $D \in \mathbb{R}^{l \times N}$ from each BiLSTM in each time step are passed to the coattention encoder in next section as input.

The motivation of adding this filter-context encoder to our model is that in the dynamic coattention networks model [1], they used only one simple LSTM encoder to generate the contextual word representation. Filter-context encoder is more advanced for: 1) it filters out the redundant information in the passage according to question, and 2) it uses two different stronger BiLSTM for question and passage separately to encode the new representations. We will later show that the filter-context encoder can actually improve the performance of the whole model greatly.

4.2 Coattention encoder (component B)

Figure 3 shows the architecture of coattention encoder. This encoder introduces the attention technique to strengthen the word representations for both question and passage, and finally fuses both attention context.

First, the encoder compute the affinity matrix between question and passage. The inputs are word representations $Q \in \mathbb{R}^{l \times M}$ and $D \in \mathbb{R}^{l \times N}$ generated by filter-context encoder in the previous section, and the affinity matrix is $L = D^T Q \in \mathbb{R}^{N \times M}$. Then L is normalized row-wise to produce the attention weights A^Q and column-wise to produce A^D:

$$A^Q = softmax(L) \in \mathbb{R}^{N \times M}, \qquad A^D = softmax(L^T) \in \mathbb{R}^{M \times N}$$

Next, it utilizes these affinity weights to produce attention contexts representations:

$$C^Q = DA^Q \in \mathbb{R}^{l \times M}$$

$$C^D = [Q; C^Q]A^Q \in \mathbb{R}^{2l \times M}$$

The last step is to send C^D, the coattention context for document, to a BiLSTM to fuse the temporal information to generate our "super powerful" final word representation:

$$\vec{h}_t^{enc} = \overrightarrow{LSTM}^{enc}\left(\vec{h}_{t-1}^{enc}, [D_t; C_t^D]\right) \in \mathbb{R}^l$$

$$\overleftarrow{h}_t^{enc} = \overleftarrow{LSTM}^{enc}\left(\overleftarrow{h}_{t+1}^{enc}, [D_t; C_t^D]\right) \in \mathbb{R}^l$$

The output vector $U \in \mathbb{R}^{2l \times N}$ from this BiLSTM in each time step are passed to the dynamic pointing decoder as input. U is called coattention encoding.

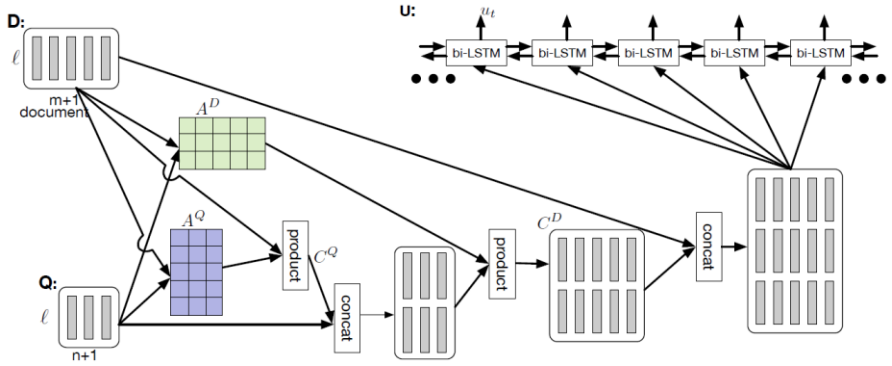

Figure 3: Coattention encoder [1].

Basically, what the coattention encoder does is to use mutual relevancy scores to filter out the redundant information and strengthen the related part for question embedding and passage embedding simultaneously; and then incorporate the temporal (contextual) information into the final embedding with BiLSTM. To some extent, this is similar to the filter-context encoder but it's more powerful since also strengthen the embedding of question using coattention while filter-context encoder is only for the passage.

The coattention encoder is chosen as part of our model because it can provide a much more powerful representation for the passage, which contains much mutual relevancy as well as contextual information. This strong encoding can be very helpful in answer prediction in the dynamic pointing decoder part.

4.3 Dynamic pointing decoder (component C)

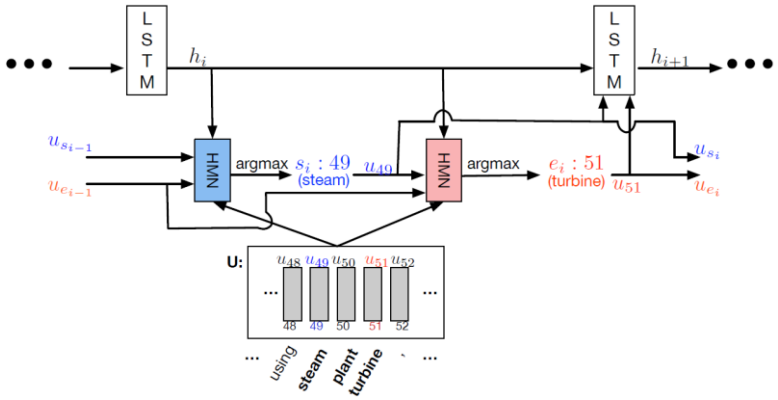

Figure 4: Dynamic pointing decoder [1].

Figure 4 shows the overview of dynamic pointing decoder. Given the coattention encoding U, the dynamic pointing decoder employs a LSTM to re-estimate the answer span for multiple times to recover from some local optima predictions:

$$h_i = LSTM^{dec}\left(h_{i-1}, \left[U_{s_{i-1}}; U_{e_{i-1}}\right]\right) \in \mathbb{R}^l,$$

where i means this is the i-th guess of the answer span, h_i is the hidden state of the i-th step, s_{i-1} and e_{i-1} are the predictions for start and end position of the answer span in the (i-1)-th guess, and $U_{s_{i-1}}$ and $U_{e_{i-1}}$ are the coattention encodings of the corresponding words at start and end position of the prediction. The output vector of LSTM at step i is denoted as $o_i \in \mathbb{R}^l$.

To predict the answer span at each iteration i, the dynamic pointing decoder generates two scores α_t and β_t for each word in the passage with its coattention encoding U_t and take the words with highest score as start and end predictions:

$$s_i = arg\max_{t=1:N} \alpha_t, \qquad e_i = arg\max_{t=1:N} \beta_t$$

For the t-th word, the start score α_t and end score β_t are calculated by two Highway Maxout Networks [1] (HMN) that don't share parameters. It is stated that HMN can pool across multiple model variations required for answering different question types and topics. It can be written as:

$$\alpha_t = HMN^{start}\left(U_t, o_i, U_{s_{i-1}}, U_{e_{i-1}}\right), \qquad \beta_t = HMN^{end}\left(U_t, o_i, U_{s_{i-1}}, U_{e_{i-1}}\right)$$

The detailed model of HMN is described as follows:

$$HMN\left(U_t, o_i, U_{s_{i-1}}, U_{e_{i-1}}\right) = \max\left(W^{(3)}\left[m_t^{(1)}; m_t^{(2)}\right] + b^{(3)}\right)$$

$$r = \tanh\left(W^{(D)}\left[o_i; U_{s_{i-1}}; U_{e_{i-1}}\right]\right) \in \mathbb{R}^l$$

$$m_t^{(1)} = \max\left(W^{(1)}[U_t; r] + b^{(1)}\right) \in \mathbb{R}^l$$

$$m_t^{(2)} = \max\left(W^{(2)} m_t^{(1)} + b^{(2)}\right) \in \mathbb{R}^l$$

where $r \in \mathbb{R}^l$ is the non-linear projection of current state with $W^{(D)} \in \mathbb{R}^{l \times 5l}$, $m_t^{(1)}$ is the output of the first maxout layer (pooling size p) with parameters $W^{(1)} \in \mathbb{R}^{p \times l \times 3l}$ and $b^{(1)} \in \mathbb{R}^{p \times l}$, and $m_t^{(2)}$ is the output of the second maxout layer with parameters $W^{(2)} \in \mathbb{R}^{p \times l \times l}$ and $b^{(1)} \in \mathbb{R}^{p \times l}$. Then both $m_t^{(1)}$ and $m_t^{(2)}$ are sent to the final maxout layer with parameters $W^{(3)} \in \mathbb{R}^{p \times 1 \times 2l}$ and $b^{(1)} \in \mathbb{R}^p$. which is called the highway connection.

For training, we minimize the cumulative softmax cross entropy of the start and end points across all iterations. We also tried different ways to combine the losses of different iterations: sum of all the losses, weighted sum of them, and only the last loss. Experiments show that the sum of all the losses can yield a better performance than the other two ways.

The motivation of adding the dynamic pointing decoder to our model is: 1) its LSTM part can predict the answer span for multiple times and is able to refine the predictions based on the previous ones and recover from local optima; and 2) its HMN part can train multiple model variations to handle different question types and topics and produce robust prediction scores.

5 Experiment, result, and discussion

5.1 Hyper-parameter selection and analysis

We implemented our model in TensorFlow and run multiple test cases to determine the best combination of hyper-parameters. Experiment results after training for 5 epochs is shown in Table 1. From it we found that 1) Cumulative softmax-CE loss gives better performance over other loss functions. This is expected as CE, or cross-entropy gives a natural loss for tasks like matching the

answer. 2) Component A (Filter-Context Encoder) boost the model a lot. This reflects that our adding of component A indeed helps the model by providing a stronger representation of the word which involves the mutual information between the paragraph and the question. 3) Change in length of LSTM in Dynamic Pointing Decoder, or H, doesn't impact the performance much, and H=4 yields a slightly better one. This is not surprising since the predicted start and end position should converge fast in order to get a minimal loss, thus making H=10 won't help much. 4) GloVe 840B is better for this task than GloVe 6B, though the improvement is limited. On one hand, GloVe 6B is trained based on Wikipedia data, thus should better match our dataset (which is from Wikipedia as well); on the other hand, GloVe 840B is trained on more text data thus the vector of each word should be more representative. Our results show that GloVe 840B is slightly better, implies that GloVe 840B could be used for more general QA tasks in the future.

Table 1: Experiments for hyper-parameter selection

test No.	d	l	H	learning rate	dropout	loss fun	F1	EM
1	200	100	4	0.005	0.1	cumulative softmax-CE	0.592	0.440
2	200	100	4	0.005	0.1	last stage softmax-CE	0.578	0.432
3	200	100	4	0.005	0.1	negative sampling	0.569	0.424
4	200	100	4	0.005	0.1	weighted CE	0.575	0.427
5	200	100	1	0.005	0.1	cumulative softmax-CE	0.575	0.433
6 (remove A)	200	100	4	0.005	0.1	cumulative softmax-CE	0.476	0.346
7	200	100	10	0.005	0.1	cumulative softmax-CE	0.586	0.435
8 (GloVe 840B)	300	100	4	0.005	0.1	cumulative softmax-CE	**0.598**	**0.450**
9	200	200	4	0.005	0.3	cumulative softmax-CE	0.532	0.387
10	200	200	4	0.01	0.3	cumulative softmax-CE	0.521	0.373
11	200	200	4	0.01	0.1	cumulative softmax-CE	0.534	0.391
12	200	200	4	0.005	0.5	cumulative softmax-CE	0.559	0.415
13	50	50	4	0.005	0	cumulative softmax-CE	0.552	0.401

After the experiments, we selected the top 3 parameter settings from experiments and use them to train our model.

5.2 Main results

After 10-hour training with more than 10 epochs, the best one among all selected parameter settings achieved an F1 score of **63.8%** and exact match (EM) of **52.3%**, with its learning curve in Figure 5. Notice that our model shows signal of over-fitting after 4K iterations, which was not well detected in our parameter-selection phase as it only tests the parameters on a 5-epoch training.

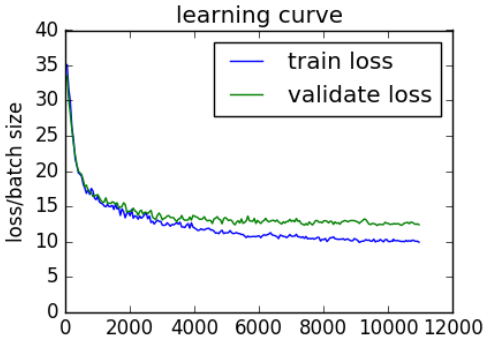

Figure 5: Learning curve.

5.3 Performance on different question types and analysis

We split the questions into different groups including “what”, “how”, “who”, “when”, “which”, “where” and “why”, and analyze our model by examining its performance across question types, as shown in Figure 6; where the height of each bar represents the mean F1 for the given question type and the lower number denotes how many instances in the dev set are of the corresponding question type. We note that our model works best for “when” questions. This may suggest that our model is best at recognizing temporal expressions. Other groups of questions whose answers are noun phrases, such as “who”, “where”, “which” and “how” questions, also get relatively better results. On the other hand, “why” questions are the hardest to answer. This is somewhat expected because the answers to “why” questions can be very diverse and complex, and they are not restricted to any certain type of phrases.

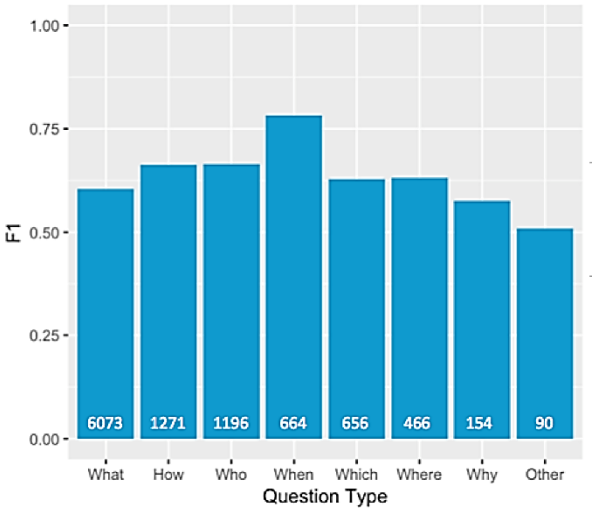

Figure 6: Performance across question types.

6 Conclusion and future work

In conclusion, we successfully introduced filter-context encoder to Dynamic Coattention Networks model, achieved 63.8% F1 and 52.3% EM, which shows a significant improvement over the logistic regression baseline model with 51% F1 and 40.4% EM [11].

Based on the performance analysis over different question types in Section 5.3, we plan to train sub-models based on the type of question type, and ensemble them to acquire the overall model. This may further improve the performance as each type of question has its own characteristic which might be learned better separately. In addition, one obvious improvement should be made in the future is to tune the dropout rate to avoid overfitting.

Work division

Yangxin Zhong: took charge of all model design, programed the data flow for training and prediction, helped in test different set of hyper-parameters, wrote final report.

Peng Yuan: took charge of all the model implementation, tested the model for different set of hyper-parameters, made analysis on the model, wrote and refined final report.

Jian Huang: took charge of data pre-processing and helper function implementation, made analysis on the model, made our poster and wrote final report.

References

[1] C. Xiong, V. Zhong, and R. Socher, "Dynamic Coattention Networks For Question Answering," *arXiv preprint arXiv:1611.01604,* 2016.

[2] Z. Wang, W. Hamza, and R. Florian, "Bilateral Multi-Perspective Matching for Natural Language Sentences," *arXiv preprint arXiv:1702.03814,* 2017.

[3] D. A. Ferrucci, "Introduction to "This is Watson"," *IBM Journal of Research and Development,* vol. 56, no. 3.4, pp. 1:1-1:15, 2012.

[4] J. Berant *et al.*, "Modeling Biological Processes for Reading Comprehension," in *EMNLP*, 2014.

[5] F. Hill, A. Bordes, S. Chopra, and J. Weston, "The Goldilocks Principle: Reading Children's Books with Explicit Memory Representations," *arXiv preprint arXiv:1511.02301,* 2015.

[6] K. M. Hermann *et al.*, "Teaching machines to read and comprehend," in *Advances in Neural Information Processing Systems*, 2015, pp. 1693-1701.

[7] D. Chen, J. Bolton, and C. D. Manning, "A thorough examination of the cnn/daily mail reading comprehension task," *arXiv preprint arXiv:1606.02858,* 2016.

[8] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," *arXiv preprint arXiv:1606.05250,* 2016.

[9] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global Vectors for Word Representation," in *EMNLP*, 2014, vol. 14, pp. 1532-1543.

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation,* vol. 9, no. 8, pp. 1735-1780, 1997.

[11] CS224N, "CS 224N: Assignment #4: Reading Comprehension," 2017.