Question Answering with Coattention Encoding and Answer Pointer Network

Yinghao Xu
Department of Electrical Engineering
Stanford University
ericx@stanford.edu

Abstract

Machine Comprehension (MC) and Question answering (QA) are difficult Natural Language Processing (NLP) task which have attracted ever increasing interest in recent years with the release of the Stanford Question Answering Dataset (SQuAD) [3]. This paper presents an end-to-end neural architecture for the QA problem on SQuAD, adopting techniques of Coattention Encoder by Xiong et al. [1], Answer Pointer Network by Wang et al. [2], and the idea of smart span. The architecture consists of a coattention encoder that encodes the passage and question words into mutually-aware representations, and an answer pointer decoder that decodes the encoded representation and predicts the answer span. With the smart span technique, the best single model achieves an F1 score of 71.42 and EM score of 59.62 on SQuAD development set.

1 Introduction

The question answering task is our setting is that given a question/query and a context/passage, we are required to predict an answer using an excerpt from the given passage. This kind of problems have seen vast real world applications such as online customer service, knowledge base querying and so on. The SQuAD dataset was built for exactly such problems and numerous deep neural network models have been proposed for tackling it. The dataset consists of 100k context-question-answer triples collected from Wikipedia articles. The goal is to construct a model to predict the start and end position of the answer excerpt since the answer excerpt must come from the original context.

Like in many other NLP problems, the key to a successful model is to have some attention mechanism to focus the question on a particular portion of the context and vice versa. There have been many high-performance attention mechanisms proposed for the SQuAD, and our model adopts the Coattention Encoder by Xiong et al. [1]. The pointer technique is similar to attention in that it directs the attention of the model to some specific locations of the words, filtering irrelevant information and producing better performance. The decoder part of the proposed model is a modified version of the Answer pointer architecture proposed in [2]. Finally, after analyzing the data examples, a technique to produce the answer span, called "smart span" is proposed to better predict the answer location.

The rest of the paper is organized as follows: Section 2 will introduce the background of the problem and related works; Section 3 presents the architecture of the proposed model; Section 4 presents the experiment details, the results and error analysis; Section 5 gives a short conclusion and reflection, and some future ideas for improvement.

2 Related Work

2.1 Stanford Question Answering Dataset (SQuAD)

The SQuAD dataset was proposed in 2016 by Rajpurkar et al. [3]. The dataset is composed of 100k context-question-answer triples. Among all the examples, about 80k are taken as training set, 10k as development set and the rest 10k examples are withheld from the public as the test set. All the examples come from Wikipedia articles and the answers are collected by crowd-sourcing. The answers to the questions are required to be excerpts from the given context texts. Despite this constraint, coming up the answers require rigorous logical reasoning and understanding of the texts. Since different people have different understanding of a given text, average human performance on the dataset has an F1 score of "only" 86.8. In the original SQuAD paper, the baseline models proposed by the author used a sliding window approach and logistic regression that achieved a 51 F1 score. Below is an example context-question-answer entry taken from the dataset:

- CONTEXT:
  "The crew of apollo 8 sent the first live televised pictures of the earth and the moon back to earth, and read from the creation story in the book of genesis, on christmas eve, 1968. an estimated one-quarter of the population of the world saw either live or delayed the christmas eve transmission during the ninth orbit of the moon. the mission and christmas provided an inspiring end to 1968, which had been a troubled year for the us, marked by vietnam war protests, race riots, and the assassinations of civil rights leader martin luther king, jr., and senator robert f. kennedy."
- QUESTION:
  How much of the population of earth ended up seeing the images of the earth and the moon?
- ANSWER:
  One-quarter

2.2 GloVe Word Embeddings

GloVe by Pennington et al. [4] is an unsupervised learning algorithm to encode words into dense vector representations. The GloVe embeddings are obtained by training a log-bilinear model with a weighted least-squares objective to learn the global word-word co-occurrence statistics from huge natural language corpus. The resulting word vectors accurately capture the co-occurrence relationship in human languages, and are commonly used in NLP problems as a substitute to the sparse word occurrence representation. Our project uses GloVe embeddings to preprocess the data texts before feeding them into the model.

2.3 Bidirectional LSTM

Long Short Term Memory is an advanced type of Recurrent Neural Network such that it uses memory blocks to store information and different types of gates to control information flow. This structure allows LSTM to enhance information flow over long period of time and avoid the vanishing gradient problem of RNN models.

Bidirectional LSTM further improves upon vanilla LSTM by processing inputs in both forward and backward direction, allowing the hidden representations to encompass past and future contextual information. It is the building block of the Coattention Encoder presented in Section 3.

2.4 Dynamic Coattention Network

The Dynamic Coattention Network (DCN) proposed by Xiong et al. [1] in 2016 is a successful model for the SQuAD dataset with an F1 score of 75.9 by a single DCN model. It consists of the context and question encoder, coattention encoder (which is adopted by our model), and a dynamic pointer decoder. Specifically, the coattention encoder encapsulates the interaction between the encoded context and question representations, which is the major contribution of this model as it invented an effective attention mechanism. The dynamic pointer encoder uses a highway maxout

network to iteratively predict start index and end index. This iterative process improves the performance by making use of the previous prediction to make new prediction and hence helps escaping from local maxima.

2.5 Pointer Net

The pointer net introduced in [2] allows the prediction of end position conditioning on the start position. The idea of "pointer" is similar to that of attention in that it enables the model to look at each location in the context to pick a best one.

3 Methods

3.1 Context and Question Encoders

After preprocessing, both context text and question text are represented using GloVe word embedding sequences. Then they are fed into a 1-layer Bidirectional LSTM shared between the context and the question. The bidirectional LSTM produces a sequence of forward hidden states and backward hidden states, which are concatenated to form the context hidden states and the question hidden states. This process can be represented by the equations below. Let $[x_1, ...x_n]$ and $[y_1, ...y_m]$ represents context and question word embedding sequences, then the encoded representations for context D and question Q are:

$$[\overleftarrow{c_1}, \overrightarrow{c_1}, ..., \overleftarrow{c_n}, \overrightarrow{c_n}] = biLSTM([x_1, ...x_n])$$

$$[\overleftarrow{q_1}, \overrightarrow{q_1}, ..., \overleftarrow{q_m}, \overrightarrow{q_m}] = biLSTM([y_1, ...y_m])$$

$$D = [\overleftarrow{c_1}; \overrightarrow{c_1}, ..., \overleftarrow{c_n}; \overrightarrow{c_n}]$$

$$Q = [\overleftarrow{q_1}; \overrightarrow{q_1}, ..., \overleftarrow{q_m}; \overrightarrow{q_m}]$$

3.2 Coattention Encoder

After obtaining the context and question encoding D and Q, we follow [1] to construct the coattention encoder layer. First we pass Q through a linear layer: $Q' = tanh(W^{(Q)}Q + b^{(Q)})$. Then we use Q' and D to compute the affinity matrx:

$$L = D^T Q'$$

The affinity is normalized with softmax row-wise and column-wise to obtain the attention weights for context and question encodings:

$$A^Q = softmax(L)$$

$$A^D = softmax(L^T)$$

Then, we calculate the attention summaries for the question C^Q and the context C^D. One thing to note is that the attention summaries for context C^D involves both the question encoding Q' as well as C^Q and they are concatenated before scaled by the weight matrix A^D.

$$C^Q = DA^Q$$

$$C^D = [Q'; C^Q]A^D$$

Finally, the question-aware context vectors C^D and the context encoding D are concatenated together to be passed into a bidirectional LSTM to fuse the temporal information and obtain the final output of the coattention encoder, U:

$$U = biLSTM([D; C^D])$$

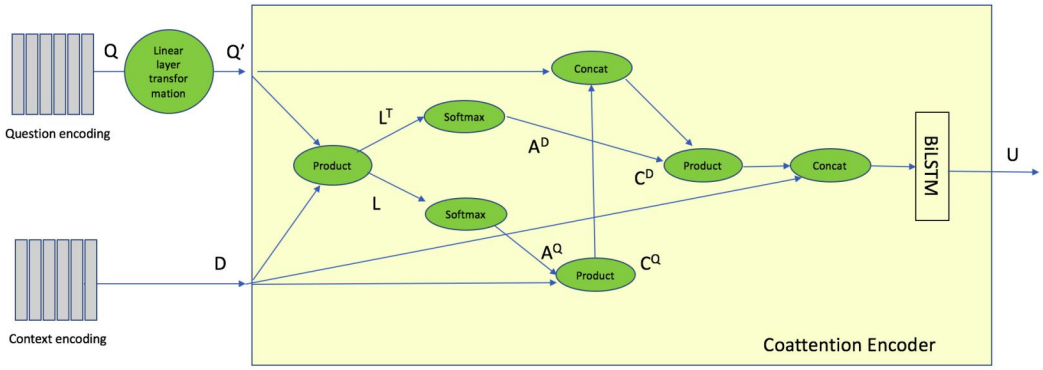

Figure 1: The Coattention Encoder

3.3 Answer Pointer

The answer pointer network in our model borrows its idea from [2], but is slightly different. In our model, the pointer network is an RNN running two time steps. It obtains its initial hidden state by passing the question encoding Q through an LSTM and taking the final hidden state:

$$h_0 = LSTM(Q)$$

As the first time step, we use this initial hidden state h_0 to attend to the coattention output U using multiplicative attention with weight matrix $W$and generate the probability distribution for the start location β_s and an attention output a_s weighted by β_s.

$$\beta_s = UW_s^U h_0$$

$$a_s = \beta_s^T U$$

On the second time step, a GRU cell will run for one time step with previous hidden state h_0 and the input a_s, producing the current hidden state h_1. h_1 will again attend to U to produce the probability distribution for the end position β_e.

$$h_1 = GRU(h_0, a_s)$$

$$\beta_e = UW_e^U h_1$$

3.4 Output layer and Smart Span

The baseline model provided by the course uses two independent softmax layer applied on the attention output to produce start and end positions. This apparently has several problems. First, since we are predicting the span of the answer, the start position should be no later than the end position, i.e. $p_s \leq p_e$. However, the baseline approach disregards this fact and simply output empty string if the end position falls before the start. Another problem is that usually context texts can be as long as a few hundred words, while the answers are usually very short. By plotting the lengths of answers we can see that the vase majority of the answers span no more than 20 words. Therefore, to address these two problems we introduce a smart span technique. The smart span restricts that the end positions must be no earlier than the start position, as well as no more than 20 words further. Respecting the smart span constraints, we can find the start and end positions from the probability distributions β_s and β_e obtained from answer pointer.

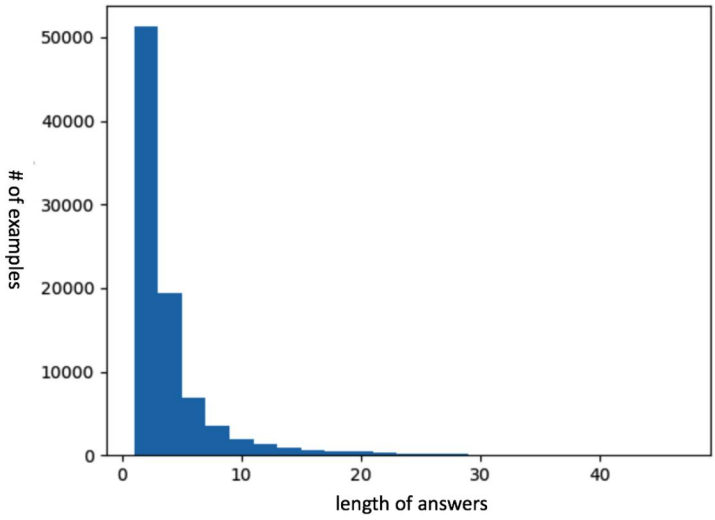

Figure 2: Answer length statistics of SQuAD

4 Experiments

4.1 Implementation details

All the words use GloVe embeddings of length 100. Different sizes of GloVe vectors (50, 100 and 200) have been experimented but not much differences have been observed. The bidirectional LSTM networks use a hidden size of 200, and batch size during training is 100. To avoid exploding gradient problem, all gradients are clipped at the maximum norm of 5. We use the Adam optimizer with a learning rate of 0.001 for training, and to regularize the model and prevent overfitting, dropout [5] is applied after each RNN network. Figure 3 shows the effect of different value of dropout, and we do observe that a larger dropout will result in a slightly slower training process, but increasing the dropout from 0.15 (as in the baseline model) to 0.25 boosts the final F1 score by 1.3%. The final model took about 15 epoch of training to reach a converged performance.

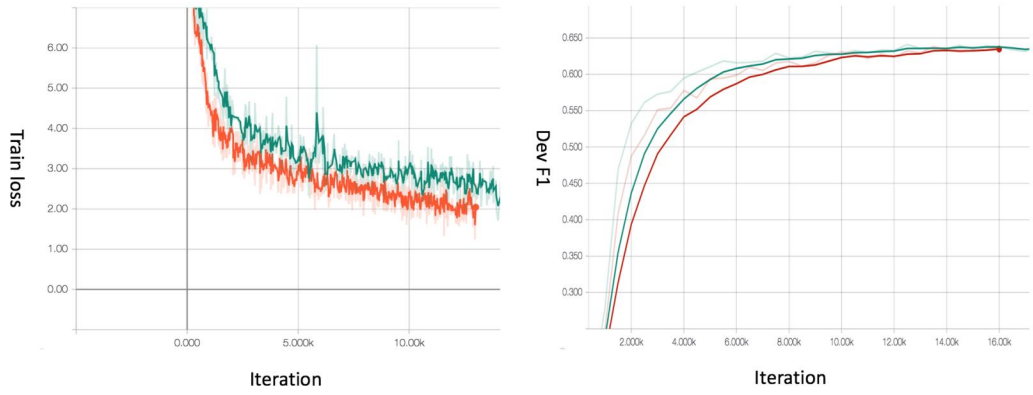

Figure 3: The train loss and Dev F1 scores obtained by dropout of 0.15 (orange) and 0.25 (green).

4.2 Results

The final model, consisting of context and question encoder, coattention encoder, answer pointer and smart span, achieved an F1 score of 71.42 and EM of 59.62 on the SQuAD development set. Different parts of the model can be considered as building blocks and we conducted experiments to compare the performance of each individual part. Below is a table showing different variants of the model we have experimented and their corresponding performance.

Due to the technical issues with CodaLab, we were not able to submit to the test set for evaluation, so only the development set results are presented.

Table 1: Model Performance Comparison (statistics on dev set)		
Model	F1 score	EM score
Baseline	43	32
Coattention	67.16	54.21
Coattention + Answer Pointer	69.18	57.38
Coattention + Answer Pointer + Smart Span (our model)	**71.42**	**59.62**

4.3 Error Analysis

The coattention mechanism is the major component of the model, and it is useful to see what kinds of errors are made by the model. After careful observation, most of the errors can be grouped into two categories: (1) Wrong/imprecise prediction of the answer span; (2) Misunderstanding of the semantics. In the first case, the model "understands" the question and passage but locates the answer span incorrectly. This type of errors is roughly estimated to account for more than 80% of the errors by looking into the prediction results. Here we provide an example of such error:

- CONTEXT:
  "Prime numbers have influenced many artists and writers . the french composer olivier messiaen used prime numbers to create ametrical music through " natural phenomena" . in works such as la nativit du seigneur (1935) and quatre tudes de rythme (194950) , he simultaneously employs motifs with lengths given by different prime numbers to create unpredictable rhythms : the primes 41 , 43 , 47 and 53 appear in **the third tude**, *neumes rythmiques" . according to messiaen this way of composing was " inspired by the movements of nature , movements of free and unequal durations "*
- QUESTION:
  In which etude of neumes rythmiques do the primes 41 , 43 , 47 and 53 appear in ?
- Predicted Answer: "**third**"
- True Answer: "**the third tude**"

The second kind of error is due to the coattention mechanism fails to direct the attention to the correct position in the context. Upon inspection, it is realized that despite the erroneous prediction, the model is able to provide an answer that has the same "part of speech" label as the true one, which can be demonstrated in the example below. This is very likely due to that the data is preprocessed using the GloVe embeddings that capture such structure.

- CONTEXT:
  "Not only are all the major british architects of the last four hundred years represented, but many european (especially italian) and american architects' drawings are held in the collection. the riba's holdings of over 330 drawings by andrea palladio are the largest in the world, other europeans well represented are jacques gentilhatre and antonio visentini. british architects whose drawings, and in some cases models of their buildings, in the collection, include: inigo jones, ***sir christopher wren****, sir john vanbrugh, nicholas hawksmoor, william kent, james gibbs, robert adam, sir william chambers, james wyatt, henry holland,*

john nash, sir john soane, sir charles barry, charles robert cockerell, augustus welby northmore pugin, sir george gilbert scott, john loughborough pearson, george edmund street, richard norman shaw, alfred waterhouse, sir edwin lutyens, charles rennie mackintosh, charles holden, frank hoar, lord richard rogers, lord norman foster, sir nicholas grimshaw, zaha hadid and alick horsnell."

- QUESTION:
  Which architect , famous for designing london 's st. paul cathedral , is represented in the riba collection?
- Predicted Answer: "**andrea palladio**"
- True Answer: "**sir christopher wren**"

Another observation is that when we implemented coattention and answer pointer along (without the smart span), the predicted end positions are still often seen to be very far away from the start position, sometimes even before, which is completely incorrect. This suggests that the answer pointer is not optimal for conditioning the end positions on start positions. One hypothesis is that the way we produce the initial hidden state of the answer pointer RNN is not optimal. Also, we used a simple multiplicative attention mechanism to produce the probability distribution of start and end positions, which does not have enough capacity for this task.

5 Conclusion

The proposed coattention model enhanced by answer pointer and smart span for the Question Answering task on SQuAD obtains an F1 score of 71.42 and EM score of 59.62 on the development set. However, since this project is done by the author individually, many ideas were not attempted due to limited time and human resources. One possible extension to the model is to develop a more sophisticated answer pointer network for predicting the start and end answer span positions, as the one in [2]. In addition, different attention mechanism such as the Bidirectional Attention Flow [6] could be implemented and integrated with the coattention encoder to produce better attention representation. Finally, an ensemble of many single models in practice always improves the performance, and therefore could be implemented as a future extension.

References

[1] C. Xiong, V. Zhong, and R. Socher, Dynamic coattention networks for question answering, arXiv preprint arXiv:1611.01604, 2016.

[2] Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. arXiv preprint arXiv:1608.07905, 2016.

[3] Rajpurkar P, Zhang J, Lopyrev K, et al. Squad: 100,000+ questions for machine comprehension of text[J]. arXiv preprint arXiv:1606.05250, 2016.

[4] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In EMNLP, volume 14, pp. 153243, 2014.

[5] Srivastava N, Hinton G E, Krizhevsky A, et al. Dropout: a simple way to prevent neural networks from overfitting[J]. Journal of Machine Learning Research, 2014,15(1): 1929-1958.

[6] Seo, Minjoon, et al. Bidirectional Attention Flow for Machine Comprehension. arXiv preprint arXiv:1611.01603 (2016).