Random Coattention Forest for Question Answering

Jheng-Hao Chen
Stanford University
jhenghao@stanford.edu

Ting-Po Lee
Stanford University
tingpo@stanford.edu

Yi-Chun Chen
Stanford University
yichunc@stanford.edu

CodaLab ID: tingpo

CodaLab Group: cs224n-tingpo_jhenghao_yichun

Abstract

We build a Random Coattention Forest Network that achieved 55% F1 score and 41% EM score on the on-line test set of question answering dataset SQuAD. We use Coattention encoder and LSTM Classification decoder to build our basic encoder-decoder pair. Multiple loosely correlated encoder-decoder pairs are trained in parallel to form a random-forest-style model. Our implementation provides efficiency in training, which allowed us to train multiple encoder-decoder pairs and run multiple experiments to tune hyperparameters.

1 Introduction

Teaching machines to read natural language documents and answer corresponding questions is a central, yet unsolved goal of NLP. Such reading comprehension and question answering abilities have a variety of applications e.g. information retrieval from commercial/legal contracts, dialog systems and chatbots designed to simulate human conversation. General neural network based sequence-to-sequence model suffers from the limitations that contexts are much longer than questions but are of equal importance and the model has to be able to produce different output from a fixed context given different questions. Those limitations make it hard to apply Machine-Translation-like models to solve question answering problems.

Given context paragraph $P = \{p_1, p_2, \ldots, p_m\}$ of length m, and a question $Q = \{q_1, q_2, \ldots, q_n\}$ of length n, the answer $\{p_{j_1}, p_{j_2}, \ldots, p_{j_l}\}$ is a contiguous part of context p with length $l \leq m$. The problem can be formulated as given a context question pair (p, q), learn a function to predict tuple $A = \{a_s, a_e\}$ where a_s and a_e indicates the starting and ending position of the answer in context paragraph p.

Stanford Question Answering Dataset (SQuAD) is the dataset we train and evaluate our model with. SQuAD is comprised of context paragraphs and around 100K question-answer pairs. The context paragraphs were extracted from a set of Wikipedia articles. The ground truth answers are genrated by human who highlight part of context paragraph as the answer to the given question. SQuAD is a challenging testbed for evaluating machine comprehension because the answers do not come from a small set of candidate answers and they have variable lengths.

2 Random Coattention Forest Networks

Our general approach is based on the conditional recurrent language model – like what a human would do, we encoded the "question" and "paragraph" into a continuous representation with RNN-based approach. We used GloVe as our distributed word representation. Coattention networks was applied to learn and encode context-question interactions. An RNN-based decoder is then used to predict the

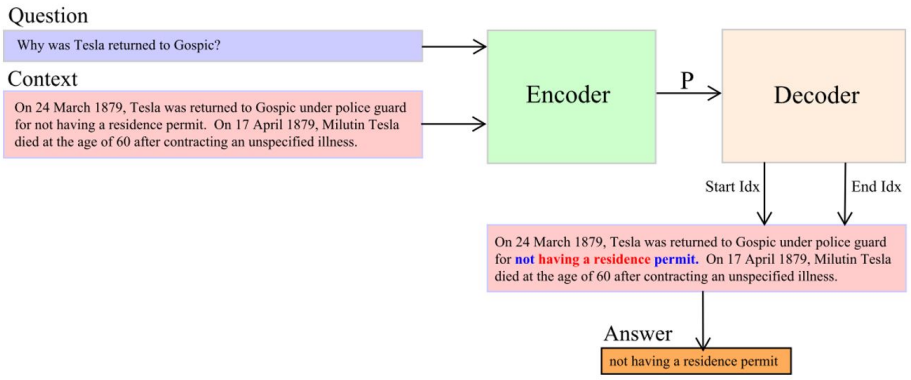

Figure 1: System Architecture.

probability distributions of starting point and ending point over the context paragraph. The encoder-decoder architecture is visualized in Fig.1. Finally multiple endoer-decoder pairs were combined to construct a random-forest style predictor to produce the predicted answer tuple $\hat{A} = \{\hat{a}_s, \hat{a}_e\}$.

2.1 Encoder Model

2.1.1 Sequence Attention Mix Encoder

Context and question word embeddings are first passed through two separate one-directional LSTMs to produce context and question representations H_P and H_Q. Then we consider the attention mechanisms by multiplying H_P and H_Q to encode context-question interactions.

$$
\begin{aligned}
H_P &= \text{LSTM}(P) = \left[h_1^P, \ldots, h_m^P\right] \\
H_Q &= \text{LSTM}(Q) = \left[h_1^Q, \ldots, h_n^Q\right] \\
A &= \text{softmax}\left(H_Q^T H_P\right) \\
C_p &= H_Q A
\end{aligned}
$$

Where $H_P \in R^{d \times p}$ and $H_Q \in R^{d \times q}$, d is the output dimension of the LSTM above. The matrix A is the attention weights of context over questions. Then the attention enhanced context representation C_P can be obtained by applying weights A to question representation H_Q.

Finally, context vector C_P and the original context representation H_P are concatenated and passed through a single neural layer to create the encoded output P' as follow

$$
P' = \text{LSTM}\left(W\left[C_P, H_P\right] + b\right)
$$

Where $W \in R^{d \times 2d}$ is the linear weights applied to mix C_P and H_P.

2.1.2 Coattention Encoder

Coattention encoder is described in [2]. Here we implement a slightly different version. Similar to the sequence attention mix encoder, context-question interactions are captured through matrix multiplications. And in coattention encoder we tried to enforce more context-question interactions through more carefully designed matrix multiplications, operations and concatenations. First, we performed a matrix multiplication and then used column and row-based softmax on context and question hidden states H_P, H_Q separately.

$$
\begin{aligned}
A_P &= \text{softmax}\left(H_Q^T H_P, dim = 1\right) \\
A_Q &= \text{softmax}\left(H_Q^T H_P, dim = 2\right)
\end{aligned}
$$

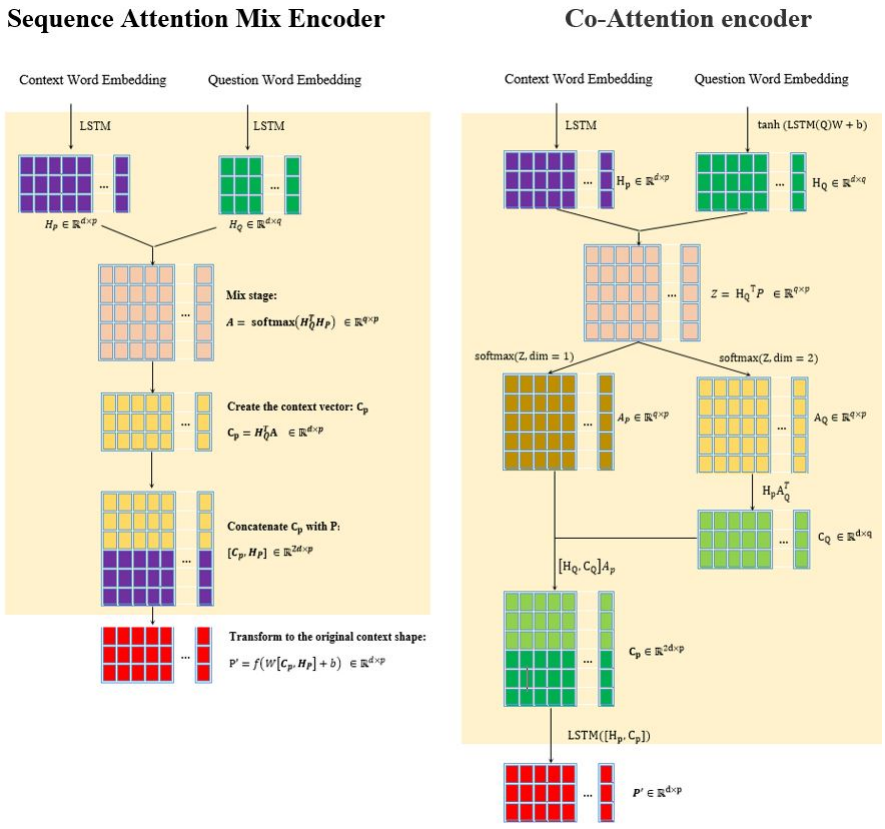

Figure 2: Attention Encoder.

Where $A_P \in R^{q \times p}$ and $A_Q \in R^{q \times p}$ are the intermediate matrices that represent attention weights. Then we first computed the question attention vector, and concatenated it with the original question hidden states H_Q to create a new context vector C_P.

$$C_Q = H_P A_Q^T$$
$$C_P = [H_Q, C_Q]\, A_P$$

Where $C_Q \in R^{d \times q}$ and $C_P \in R^{2d \times p}$ are the context vectors for question and context. Finally, the concatenated vector is passed to an one-direction LSTM to compute the encoded output P', which remains the same dimension with the original hidden state H_P

$$P' = \text{LSTM}\left(W\left[H_P, C_P\right] + b\right)$$

Figure 2 shows sequence attention mix and coattention encoder that we discussed.

2.2 LSTM Classification Decoder

Encoder input P' is used to predict the probability distribution $f_{a_s}\left(i|P'\right)$, where i indicates a position in P. P' is then passed through a one-directional LSTM to produce transformed encoded information P'' and the probability distribution of ending position $f_{a_e}\left(j|P''\right)$ is computed accordingly. Where j is a position in P.

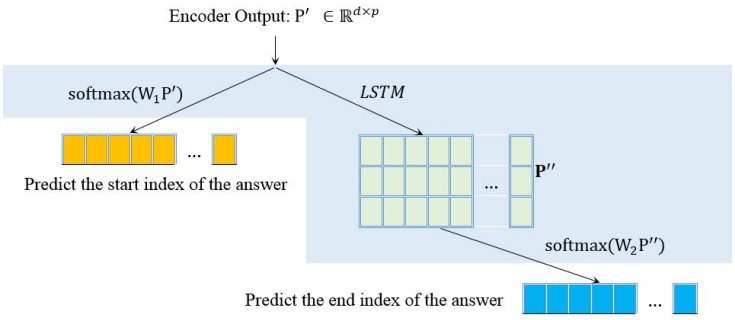

Figure 3: LSTM Classification Decoder.

Single Coattention Network			Random Coattention Forest		
	F1	EM		F1	EM
Training	69.7	57.0	Training	94.4	90.0
Validation	45.2	35.0	Validation	65.3	50.0
Development	31.2	21.7	Development	53.9	39.7
Test	31.3	20.5	Test	**54.6**	**41.0**

Table 1: Experiment results of single Coattention Network and Random Coattention Forest with 3 networks.

$$
\begin{aligned}
f_{a_s}(i|P') &= \text{softmax}(W_1 P') \\
P'' &= \text{LSTM}(P') \\
f_{a_e}(j|P'') &= \text{softmax}(W_2 P'') \\
\hat{a}_s &= \arg\max_i f_{a_s}(i|P') \\
\hat{a}_e &= \arg\max_j f_{a_e}(j|P')
\end{aligned}
$$

Where $W_1 \in R^{1\times d}$, $W_2 \in R^{1\times d}$, $v_{start} \in R^{1\times p}$. The decoder operation is visualized in Fig.3.

2.3 Random Forest Network

Our implementation of coattention encoder is actually a weaker version than described in [2]. The design choice is made so that the encoder-decoder pairs fit better as building blocks of a random-forest-style structure. Random forest predictors are formed by ensembling several loosely correlated weak predictors. Our coattention encoder and Classification decoder uses one-directional LSTMs, which provides better computational efficiency. It also helps decreases model complexity to prevent overfitting, which, in our experiment results later, is shown to be a major challenge to generalize our model.

During the training phase, all encoder-decoder pairs are given the same input, however, each encoder-decoder pair sees different input due to random dropout. Also, different random start was applied. The two sources randomness introduced above makes our encoder-decoder pairs only loosely correlated with each other, which is idea for building random forest networks.

3 Experiments

Our model was trained and tested with SQuAD dataset. The 105K question-answer pairs in SQuAD was randomly divided into four parts: training (81K), validation (4K), development (10K) and test (10K). Where the test set was never seen by the model throughout the model training process and was used for final evaluation only. Validation set was used to tune hyperparameters and development

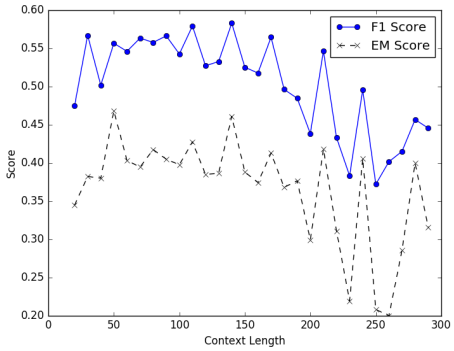

Figure 4: Performance for different context length.

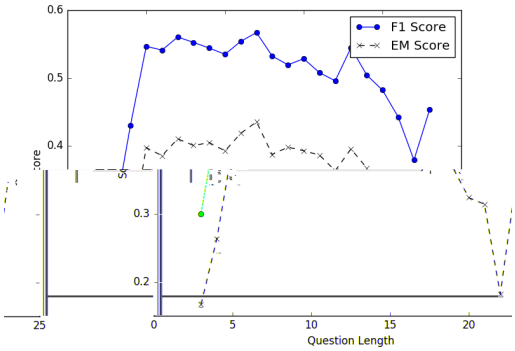

Figure 5: Performance for different question length.

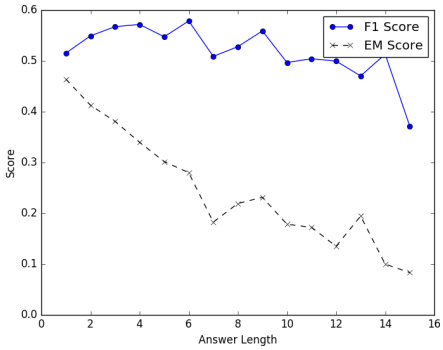

Figure 6: Performance for different answer length.

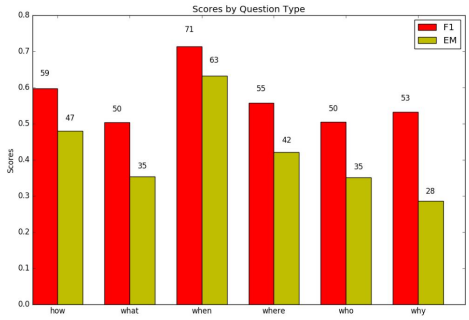

Figure 7: Performance for question types.

set was used to estimate model performance during the training process. The model was trained with dropout and exponential learning rate decay using Adam optimizer.

3.1 Evaluation Metrics

We use exact match (EM) and F1 score to evaluate our experimental results. F1 score is a metric that loosely measures the average overlap between the prediction and ground truth answer. Exact match measures the percentage of predictions that match one of the ground truth answers exactly. We treat the prediction and ground truth as bags of tokens, and compute their F1. We take the maximum F1 over all of the ground truth answers for a given question, and then average over all of the questions.

3.2 Performance

Our experiment results are shown in Table.1. The Random Coattention Forest we implemented consists of 3 Coattention Networks. The benefit of ensembling loosely correlated coattention networks is obvious, near 20% boost in F1 and EM scores are observed when compared to our implementation of single Coattention Network. This shows the correlation between different encoder-decoder pairs is small as we designed and how weak predictors are suitable for building a random forest network.

It took around 120 seconds to train one full epoch (81K training samples) on one *Tesla M60* GPU with 8GB memory. This computational efficiency was due to our design choice of building weaker version of coattention network encoder and a simple decoder design. Training one epoch on our Random Coattention Forest with 3 encoder-decoder networks took around 480 seconds, which is still a desirable computational performance.

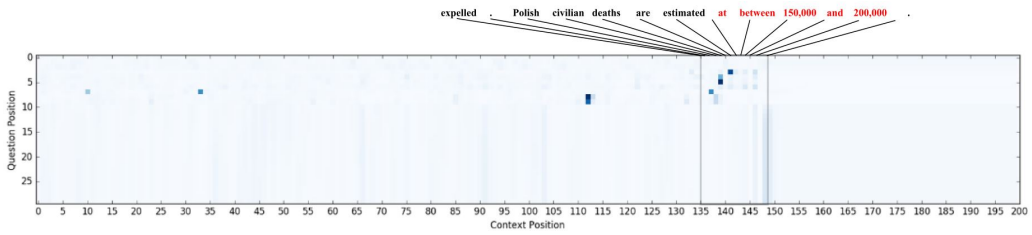

Figure 8: A_P matrix visualization. Input context was *"..... expelled. Polish civilian deaths are estimated at between 150,000 and 200,000."* Input question was *"What is the estimated death toll for Polish civilians?"* And the corresponding answer was *"between 150,000 and 200,000"*.

3.3 Coattention Mechanism

The coattention encoder captures context-question interactions through matrix multiplications. Fig.8 provides a visualization of how this attention mechanism works. The visualization demonstrated a transposed A_P matrix which was obtained by multiplying context and question representation then *softmax* over context words. It shows A_P activated in context area before the answer. Actually, since A_P is derived through context-question multiplication, when common words between context and question meet, their *cosine* distance is small (context and question word embeddings went through different LSTMs, so they will not be exactly the same) and inner product is likely to be large. This results in a highlighted area in A_P. As the example in Fig.8 shows, answer range is generally in the context near where the question words appear, thus highlighted area in context gives a hint for where the true answer may reside and will be helpful in predicting answer positions.

3.4 Length Analysis

Fig.4, 5 and 6 plot model performance with respect to the length of context paragraphs, questions and ground truth answers, respectively. The number of samples we have for question length < 4 is too small so shall be excluded from analysis. Generally, F1 and EM performance gradually decrease as the length of context paragraphs, questions and ground truth answers increases. E.g. for contexts containing more than 250 characters, the F1 score of our model drops to around 40% and the exact match score drops to around 20%, compared to the F1 score and exact match score of close to 57% and 35%, for context with shorter length. For question containing more than 23 characters, the F1 score drops to around 37% and the exact match score drops to around 10%, compared to the F1 score and exact match score close to 55% and 40% for question with shorter length.

It's worth pointing out as the length of ground truth answers grow, the corresponding F1 and EM scores drop with different slope. While F1 scores drop gradually, EM drops to lower than 10% as answer length exceeds 24. As the length of target answer grows, it's more likely for our model to make a "mistake" when predicting starting and ending points, therefore the sharp drop in EM scores. But this also shows our model can still make a reasonable prediction of answer range, thus the F1 score drops much slower.

3.5 Question Types Analysis

We analyze the performance of our model on different types of questions. We basically split the questions into different groups including "how," "what," "when," "where," "who," and "why. These different words refer to questions with different types of answers. According to the F1 and EM scores on the development set, our model performs the best for "when" questions. This may be because temporal expressions is much more distinguishable in the context. "How" questions come as the second best performing type of questions, "how" type includes questions like "how much" or "how many" which are generally refer to numbers, making them easier to answer.

It can be observed that "why" types of question has average F1 performance but the worst EM performance. The is the direct result from average answer length with respect to different type of questions. The average answer length of different types of questions are: 6.75 for "why", 3.04 for "what", 2.90 for "where", 2.59 for "who", 2.52 for "how" and 2.31 for "when". As we've seen in

Fig.6 that EM score drops sharply with answer length, there's no surprise the "why" type questions have the worst EM performance.

4 Conclusion

We build a Random Coattention Forest model that solves question answering problem to a satisfactory level. With the nature of loosely correlated encoder-decoder pairs, ensembling multiple encoder-decoder pairs gave significant boost in F1 and EM performance. Our design choice of building weaker encoders and decoders enabled us to build a random-forest-style model both theoretically and computationally.

Due to computing power limitation, we did not experiment with larger number of ensembled encoder-decoder pairs and larger state sizes which may possibly boost model performance even more. We leave these as future directions. Also, it's worth exploring the trade-offs between encoder/decoder complexity and the performance of random-forest-style setting. Though random-forest-style settings prefer weaker individual predictors, a more complex encoder/decoder design can be expected to have better individual level performance.

We had a 40% gap between F1 scores of training and test set, indicating a room for improvement if we can further eliminate overfitting. A finer tuned hyperparameter set may be a possible future solution. Dropout rate, learning rate decay rate, number of encoder-decoder pairs in random forest network can all be tuned to find a better regularization. Experimental design techniques like Kriging may provide a systematic way to search through the parameter space. Given the fast-training nature of our model, such systematic hyperparameter search is possible.

References

[1] Minh-Thang Luong, Hieu Pham, and Christopher D. Manning. Effective Approaches to Attention-based Neural Machine Translation. *arXiv preprint arXiv:1508.04025*, 2015.

[2] Wang, S. & Jiang, J. (2017) Machine Comprehension Using Match-LSTM and Answer Pointer. *arXiv preprint arXiv:1608.07905v2*, 2016.

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

[4] Dzmitry Bahdanau, KyungHyun Cho, Yoshua Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. *arXiv preprint arXiv:1409.0473*, 2016.

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

[6] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100,000+ questions for machine comprehension of text. In Empirical Methods in Natural Language Processing (EMNLP), 2016.