Codalab Username: *yanshuh*

Coattention Answer-Pointer Networks for Question Answering

Yanshu Hong, Yiju Hou, Tian Zhao
Department of Computer Science
Stanford University
Stanford, CA 94305
{yanshuh, yijuhou, tianzhao}@cs.stanford.edu

Abstract

Machine comprehension (MC) and question answering (QA) are crucial tasks in natural language understanding. Training deep neural network-based QA models has become practical upon the recent release of the Stanford Question Answering Dataset (SQuAD), a significantly larger dataset of question-answer pairs created by humans on a set of Wikipedia articles [1]. In this paper, we propose an end-to-end neural architecture for this task. The architecture consists of a Dynamic Coattention Network (DCN) encoder and a Match-LSTM decoder. On the hidden SQuAD test set, our model achieves 68.92% F1 and 57.56% EM.

1 Introduction

Machine Comprehension (MC) is an unsolved challenge in natural language processing. In recent years, several benchmark datasets have been developed to measure and accelerate the progress of MC architecture. However, these datasets do not provide sufficient human-labeled data to train expressive models. To address the deficiency of the previous datasets, Rajpurkar et al. developed the SQuAD dataset. The SQuAD task is to predict an arbitrary answer span within the given context paragraph.

The SQuAD dataset is significantly more challenging than previous datasets. For example, SQuAD requires deep language understanding, logical reasoning, and multi-sentence reasoning. The fact that the answer can be an arbitrary span within the context requires modeling complex interactions between the context and the question. Human reading comprehension interprets the context with the question in mind. Equivalently, a competent MC model should be able to use attention mechanism to focus on a certain portion of the context and abstract the information.

In this paper, we assess the strength and weakness of models proposed in previous researches and create new models that combine the advantages of these models.

2 Related Work

The model proposed in the original paper from Rajpurkar et al. on the SQuAD dataset first generates a list of candidate answer spans and then uses a linear model with carefully crafted lexical features to select the best answer. In later researches, deep neural network-based architectures with attention mechanism are most commonly used on the SQuAD task. After reviewing a variety of recent researches that have pushed forward the frontiers of MC, we decided to keep exploring the potential of the following four deep neural network architectures: Dynamic Coattention Network (DCN), Answer Pointer (Ans-Ptr), Match-LSTM and LSTM.

2.1 Dynamic Coattention Network

The DCN model is an end-to-end neural network for question answering [2]. The model consists of a coattention encoder attending to the question and the context paragraph simultaneously. For complex context and questions, human readers often revisit the given context and question to perform deeper inference. The coattention encoder captures the interactions between the question and the context by simulating this revisiting behavior. This model also consists of a dynamic pointing decoder that alternates between assessing the most likely start and end position of the answer span.

2.2 Match-LSTM

Another model that has demonstrated high performance on the SQuAD task is Match-LSTM, consisting of a Match-LSTM encoder and a Ans-Ptr decoder [3]. The Match-LSTM model sequentially aggregates the matching between context and the question with weighted attention. Then the Ans-Ptr selects a list of positions from the context paragraph as the final answer. Additionally, we have adopted an assortment of deep learning optimization strategies that could improve the performance of Machine Comprehension tasks in practice, such as dropout, sentinel vectors, and flexible RNN sequence length.

2.3 Dropout

Dropout is a technique that randomly drops units from the neural network during training, preventing the units from overly co-adapting to training data [4]. Dropout also provides a more efficient alternative to approximately combining many different neural network architectures to improve the overall model performance.

2.4 Sentinel and Trainable Null

Sentinel vectors are designed for handling rare words in NLP [5]. A hidden state has limited capacity in retrieving information from relevant previous hidden states. Even with attention mechanism, most classifiers do not have good performance in tasks involving predicting rare or previously unknown

words. Therefore, the sentinel is proposed to increase hidden state capacity by providing a trainable vehicle that is not tied to time steps.

Inspired by previous work, we have developed our MC model using a Coattention encoder and an Ans-Ptr decoder.

3 Approach

Our end-to-end architecture includes a Coattention encoder and an Ans-Ptr decoder. See Figure 1. In section 3.1, we will give an overview on the encoder and then on the decoder in Section 3.2.

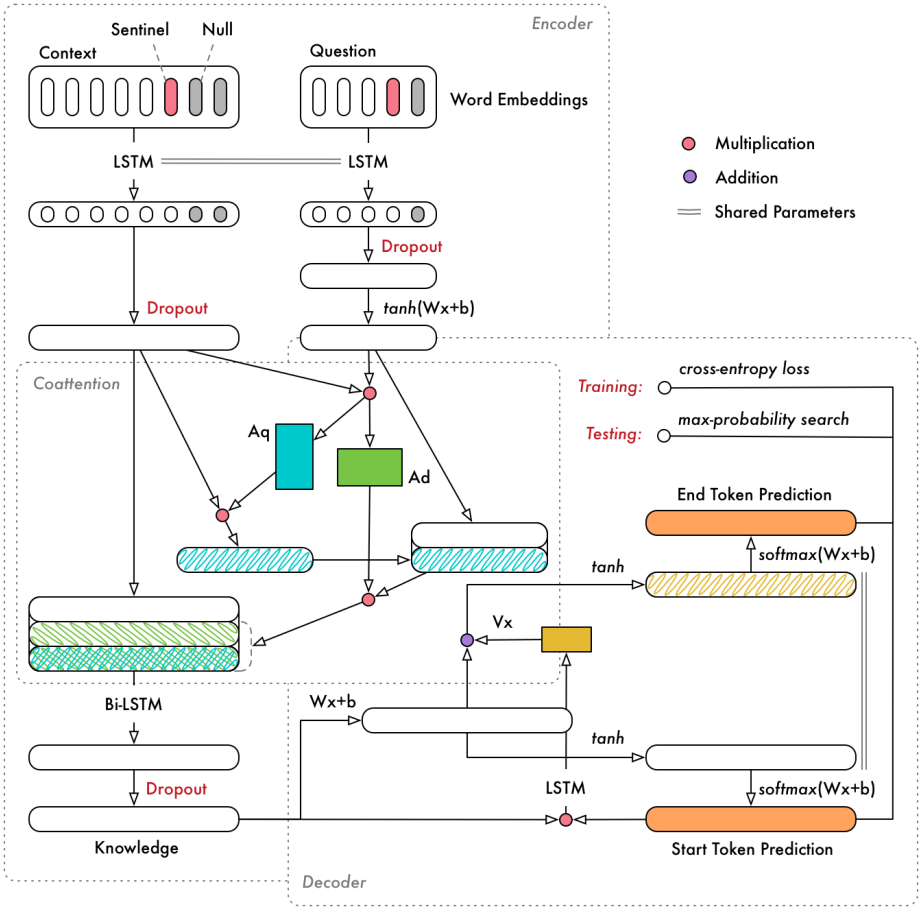

Figure 1: Coatt-Ans-Ptr Model

3.1 Coattention Encoder

Let $(x_1^C, x_2^C, \cdots, x_m^C)$ denote the context vector. In a Coattention encoder, a context paragraph is passed into an LSTM encoder to generate the context encoding: $c_t = LSTM_{enc}(c_{t-1}, x_t^C)$. The context encoding matrix C is defined as $C = [c_1, c_2, \cdots, c_m, c_\emptyset] \in \mathbb{R}^{l \times (m+1)}$. The sentinel vector $c_\emptyset$ prevents the model from overfitting on a particular word from the context. In the output of $LSTM$, we add an additional dropout layer to prevent overfitting.

Let $\left(x_1^Q, x_2^Q, \cdots, x_n^Q\right)$ denote the question word vector. The question encodings are computed with the same $LSTM_{enc}$: $Q' = LSTM_{enc}\left(q_{t-1}, x_t^Q\right)$. Similarly, we concatenate this representation with a sentinel vector. We also add a dropout layer at the output of LSTM.

To separate the projection space of context and question representations, an affine transformation is performed on the question encoding matrix. The final question representation is: $Q = \tanh(W^Q Q' + b^Q) \in \mathbb{R}^{l \times (n+1)}$.

To compute the coattention matrix, we first generate an affinity matrix corresponding to all context-question pairs: $L = C^T Q \in \mathbb{R}^{(m+1)\times(n+1)}$. Then we normalize the affinity matrix row-wise to produce an attention matrix across the document for each word in the question. Similarly, we generate attention matrix across the question for each word in the document by column-wise normalization: $A^Q = softmax(L),\ A^C = softmax(L^T)$.

Next, we compute the following three summary matrices:

1. Summary of the context with respect to the question: $M^Q = CA^Q \in \mathbb{R}^{l\times(n+1)}$.
2. Summary of the question with respect to the context: $M^C = QA^C \in \mathbb{R}^{l\times(m+1)}$.
3. Summary of the previous attention context incorporating each word of the document: $M^Q A^C$. This summary maps encoding of question into the space of context encodings.

We fuse the last two summaries to generate a co-dependent representation of the question and the context:

$M^C = [Q;\ M^Q]A^C \in \mathbb{R}^{2l\times(m+1)}$. After obtaining the fused representation, we pass it through a bidirectional LSTM to obtain a sequence of encoding state h, defined as: $h_t = BiLSTM\ (h_{t-1}, h_{t+1}, [c_t; m_t^C])$. We concatenate h_t at each LSTM propagation to generate our final encoding $H = [h_1, \cdots, h_m] \in \mathbb{R}^{2l\times m}$. We then apply a dropout at the final encoding to prepare the knowledge representation for the decoder.

3.2 Answer Pointer Decoder

First, we will give an overview on the sequence model to provide some context on the Answer Pointer decoder. In a sequence model, the answer is generated as a sequence of indices: $\boldsymbol{a} = (a_1, a_2, \cdots)$,

$a \in [1, P+1]$. Each index corresponds to the position of a selected token in the original paragraph. $P+1$ is a special position that indicates the end of the answer.

We want to obtain an attention weight vector, β, where $\beta_{k,j}$ is the

probability of selecting the j^{th} token from the context as the k^{th} token in the answer. $\beta_{k,P+1}$ gives the probability of stopping answer generation at position k. We model β as follows:

We first predict the distribution of the start position by computing: $F_{start} = \tanh(HV + b_a), \beta_{start} = F_{start} v_{beta} + b_{beta}$, where $V \in \mathbb{R}^{2l \times l}, b_a, v_{beta}, b_{beta}$ are parameters to be learned. $\beta_{sp} = softmax(\beta_{start})$ gives the normalized distribution for the start position.

We then pass β_{start} to the following $LSTM$: $h_{start} = LSTM(\beta_{start})$ for one step. To calculate the distribution of the end position, we use: $F_{end} = \tanh(HV + h_{start} W_a + b_a)$, $\beta_{end} = F_{end} v_{beta} + b_{beta}$, where $W_a \in \mathbb{R}^{l \times l}, b_a, b_{beta}$, are trainable variables. $\beta_{ep} = softmax(\beta_{end})$ gives the normalized distribution of the end position. To train the model, we minimize the cross-entropy loss.

4 Experiments

We implemented and tested an assortment of model variants and then chose the most robust one to improve by fine tuning model parameters. We implemented the following 7 models, and further optimized a subset of them. After evaluating the performance on these models, we decided to move forward with the Coatt-LSTM and Coatt-Ans-Ptr models. See Table 1.

Table 1: Experiment Results

Encoder	**Decoder**	**Optimization**	**Performance %**	**Conclusion**
Coattention	HMN	-	-	Too expensive to train
Coattention	Affine	-	F1 = 11.51, EM = 8.59	Decoder is not expressive enough
LSTM (baseline)	LSTM (baseline)	-	F1 = 30.06, EM = 21.09	Competent model. We need to update the implementation of the bias term
Coattention	LSTM (baseline)	-	F1 = 46.80, EM = 35.16	Competent model. We need to implement dropout to solve the overfitting issue
Match-LSTM	Ans-Ptr	-	F1 = 39.65, EM = 27.34	Training is very slow. Not enough time for design space exploration
Coattention	LSTM (baseline)	Dropout, Sentinel Vector, Flexible RNN sequence length	F1 = 59.57, EM = 48.44	Dropout is effective
Coattention	**Ans-Ptr**	Dropout, Sentinel Vector, Flexible RNN sequence length	**F1 = 68.92, EM = 57.56**	**Best performance**

We expected the LSTM baseline model to have a better performance after training on all 20 epochs, so we went back to diagnose the system and we found a problem with the bias term. We were initially using a bias term of shape [1, max_paragraph_length, state_size], which gave the model excessive degree of freedom during training by removing the state-specific contrast from the bias term. Therefore, we updated the bias term to [1, 1, state_size]. In retrospect, the earlier attempt with the affine decoder might also have been impacted by this issue.

After updating the implementation of the bias term, we also introduced the sentinel vector to the encoder. From observing the data set, we found the start and end words could potentially be one of the low-frequency words, and the sentinel will allow the model to address rare words more effectively from relevant previous hidden states. We also added trainable null paddings to the end of the question and context. Unlike the untrainable embeddings, these paddings managed to capture information about the unseen words during training. This optimization increased the training speed and performance remarkable.

We also observed that the question and the context can vary significantly in length. To further reduce the training time, we adopted flexible RNN sequence length, which allows the runtime of encoding to be linearly correlated with the length of the question or context.

Moreover, we noticed the evidence of overfitting after epoch 10 when the divergence between training result and test result started increasing. We suspected the overfitting problem can be resolved by introducing a 0.2 dropout into the model.

We also observed a linear decrease in the loss curve, which indicated a no optimal learning rate. We later found that our current learning rate of 1e-3 is not optimal for Adam Optimizer, and therefore we should adjust it to 1e-4. Consequently, we decided to revisit the two models that had the potential to reach desirable performance with the updated learning rate: (1) Coatt-LSTM, and (2) Coatt-Ans-Ptr.

Figure 2 shows the training progress of the two models. Both models have been optimized with the sentinel vector, dropout (at 0.2) and flexible RNN sequence length.

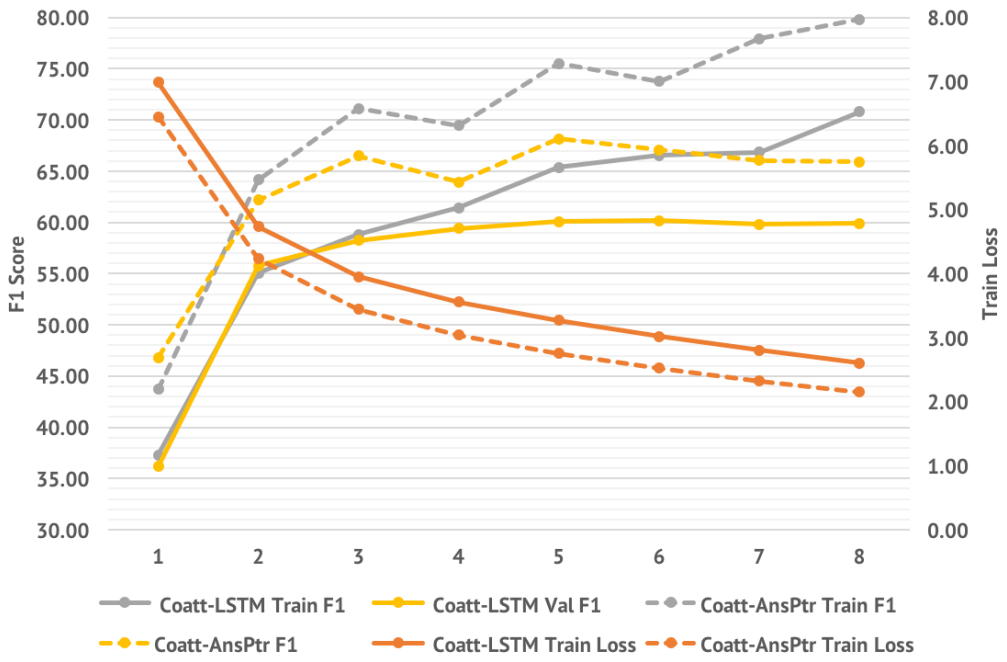

Figure 2: Coatt-Ans-Ptr and Coatt-LSTM Model Training Curve

The most notable change was that the loss showed exponential decay over training, which validated our previous assumption on the non-optimal learning rate. Moreover, in both models gaps between the F1 scores on the validation set and the development set are smaller comparing with the previous training. By comparing the training statistics of the two models, we concluded that the Coatt-Ans-Ptr has a better performance on the SQuAD task.

5 Results

Our model achieves 68.92% F1 and 57.56% EM on the SQuAD dataset, which indicates the robustness of model quantitatively. We would like to give a brief analysis on the predicted answers vs. the ground truth. We discovered that our model performs the worst on the following three categories of question-answer pairs:

1. Numbers. For example, our model completely failed answering a question about percentage of bachelor degrees in US. We conclude that our model is failing because our word embeddings do not handle percentage numbers.
2. People's names positioned closely in the context. For example, our model is not able to tell the differences between "Zachary Taylor, Ulysses S. Grant, William Howard Taft". We hypothesize that our model is confused by the similar correlations the three names resemble to the rest of the context.
3. Answer with ambiguous boundaries. For example, in one QA pair, our model predicted "5th century CE" while the ground truth is "5th

century". The tiny difference still indicates the robustness of our model, but it also shows that our model is not very sensitive on deciding when to stop during decoding.

6 Conclusion

In this paper, we present an end-to-end MC system, in which we identify the key components crucial to compressing and decompressing the information in contexts and questions. Moreover, we manage to verify the potential of our model through well-defined experiments. We observe that the gap between our training and validation sets are reasonably narrow, which indicates the robustness of our model.

In the future, we would like to improve our implementation from three perspectives. First, we plan to explore more options with the architecture, such as more expressive attention mechanisms, as well as decoders from other sources. Second, we are also constrained by time in design parameters exploration. We would like to experiment with more combinations of hyperparameters to quantify their impacts individually on the training process. Third, we would like to develop a more descriptive preprocessing script. We observe that many of the provided words are rarely used, but they do exist in the vocabulary and in the GloVe embeddings. We would like to prune away these redundancies to improve the efficiency of our model.

References

[1] Rajpurkar, P., Zhang, J., Lopyrev, K., Liang, P. (2016) SQuAD: 100, 000+ Questions for Machine Comprehension of Text. *CoRR* abs/1606.05250.

[2] Xiong, C., Zhong, V., Socher, R. (2016) Dynamic Coattention Networks For Question Answering. *CoRR* abs/1611.01604.

[3] Wang, S., Jiang, J. (2016) Machine Comprehension Using Match-LSTM and Answer Pointer. *CoRR* abs/1608.07905.

[4] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R. (2014) Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15 (2014) 1929-1958

[5] Merity, S., Xiong, C., Bradbury, J., Socher, R. (2016) Pointer Sentinel Mixture Models. *CoRR* abs/1609.07843