CS224N: Assignment 4 Reading Comprehension

Clare Chen
Stanford University
Stanford, CA
cchen9@stanford.edu

Kevin Luo
Stanford University
Stanford, CA
kluo8128@stanford.edu
CodaLab username: kluo8128

Abstract

The Dynamic Coattention Network [1] is an end-to-end neural network model for question answering. The model consists of a coattentive encoder that captures the interactions between the question and the context paragraph, as well as a dynamic pointing decoder that alternates between estimating the start and end of the answer span. Due to the extensive training time needed by the complex decoder in this model, this project explores the possibility of achieving comparable QA performance using the same neural network model, albeit with a simpler decoder structure. We discover that a complex decoder structure is necessary for excellent QA performance. Our model achieves an F1 score of 55.1 and an EM score of 42.4.

1 Introduction

Question answering (QA) is a central task in natural language processing. It is important to craft large and natural datasets to train and evaluate models for the QA task. Early QA datasets were initially human-annotated and small in size. Researchers then developed large-scale datasets through semi-automated techniques. However, the drawback of these datasets is that they differ from more natural, human annotated datasets.

Recently, Rajpurkar et al. [2] released the Stanford Question Answering dataset (SQuAD), which is larger than earlier datasets. SQuAD is composed of 100,000 question-answer pairs along with context paragraphs. Each answer is a span of words in the provided context paragraph, confining answers to the space of all possible spans. Thus the QA task, which is the objective of this paper, reduces to the following: given a question, the model should predict an answer span, a tuple (a_s, a_e) composed of the start index and end index, inclusive, within the given context paragraph.

2 Background/Related Work

Many deep learning based models were proposed since the release of the SQuAD dataset. Most of these models belong to one of two classes, based on how they identify the answer spans [3].

2.1 Chunking and Ranking

Chunking and ranking models extract a list of candidate chunks as answers and then rank the chunks so that the correct chunk is at the top of the list. For example, the Dynamic Chunk Reader model [4] uses part-of-speech patterns to extract candidate chunks. Then these chunks are ranked using an attention-based RNN model.

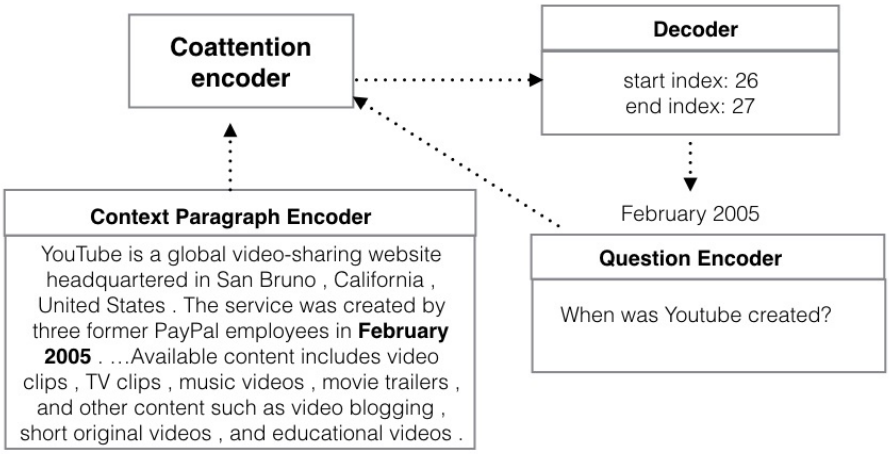


Figure 1: Overview of our model.

2.2 Boundary Identification

Boundary identification models identify the answer span directly rather than extracting a list of candidate answers. At each time step, context paragraph words are passed into a recurrent network model to encode question-aware representations. Then the start and end indexes of the answer are predicted based on the representations encoded in the states of the recurrent model.

For example, the Match-LSTM model [5] matches the context paragraph with the question and uses a Pointer Network [6] to return a list of positions from the context paragraph. On the other hand, the Dynamic Coattention Network model [1] utilizes a co-attentive encoder to capture interactions between the question and context paragraph and a dynamic pointing decoder to predict the start and end indexes. The BiDAF model [7] uses a bi-directional attention flow mechanism to produce question-aware context representations for the context paragraph; the start and end indexes are predicted based on the representations. The Multi-Perspective Context Matching model [3] matches the contextual embeddings of the context paragraph with the question from multiple perspectives to generate question-aware representations.

3 Approach

Our model is a boundary identification model consisting of the coattention encoder from the Dynamic Coattention Network model [1] and a decoder, as shown in Figure 1.

3.1 Document and Question Encoder

Let $(x_1^Q, x_2^Q, \ldots, x_n^Q)$ denote the question embeddings and $(x_1^D, x_2^D, \ldots, x_m^D)$ denote the context paragraph embeddings, where n is the question length and m is the context paragraph length. Then using an LSTM [8], the context paragraph encodings are $d_t = \text{LSTM}_D(d_{t-1}, x_t^D) \in \mathbb{R}^\ell$, for $t = 1, \ldots, m$, where ℓ is the hidden state size of the LSTM. The context paragraph encoding matrix D is the concatenation of all the context paragraph encodings: $D = [d_1, \cdots, d_m] \in \mathbb{R}^{\ell \times m}$.

Similarly, the question encodings are $q_t = \text{LSTM}_Q(q_{t-1}, x_t^Q) \in \mathbb{R}^\ell$, for $t = 1, \ldots, n$. The intermediate question encoding matrix Q' is the concatenation of all the question encodings: $Q' = [q_1, \cdots, q_m] \in \mathbb{R}^{\ell \times n}$. We apply a non-linear projection layer to the question encodings to allow for variation between the question and context paragraph encoding spaces [1]. This produces the question encoding matrix $Q = \tanh(W^{(Q)} Q' + b^{(Q)}) \in \mathbb{R}^{\ell \times n}$, where $W^{(Q)} \in \mathbb{R}^{\ell \times \ell}, b^{(Q)} \in \mathbb{R}^{\ell \times n}$ are trainable parameters.

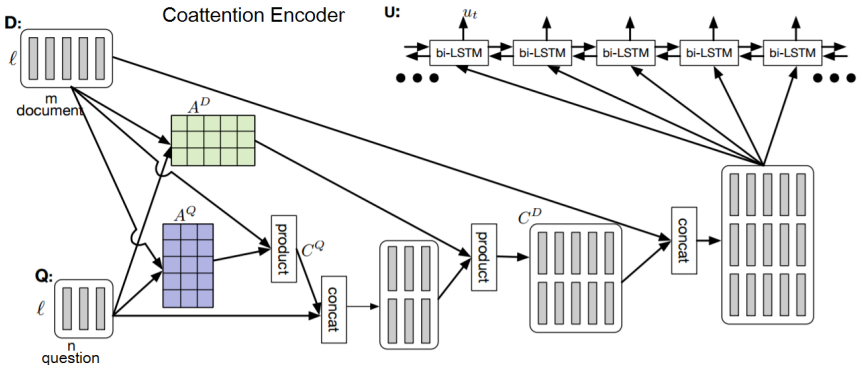


Figure 2: Coattention encoder.

3.2 Coattention Encoder

The coattention encoder attends to both the question and context paragraph at the same time, and it combines both attention contexts, as shown in Figure 2. The affinity matrix L is computed from the context paragraph and question encoding matrices: $L = D^\top Q \in \mathbb{R}^{m \times n}$. Then the affinity matrix is normalized row-wise to generate attention weights A^Q across the context paragraph for each word. Similarly, the affinity matrix is normalized column-wise to generate attention weights A^D across the question for each context paragraph word.

$$A^Q = \text{softmax}(L) \in \mathbb{R}^{m \times n}, \qquad A^D = \text{softmax}(L^\top) \in \mathbb{R}^{n \times m}. \tag{1}$$

Then the attention summaries C^Q of the context paragraphs are computed for each question word:

$$C^Q = DA^Q \in \mathbb{R}^{\ell \times n}. \tag{2}$$

The coattention context C^D is a co-dependent representation of the question and context paragraph. It contains the attention summaries QA^D of the question for each context paragraph word, as well as the summaries $C^Q A^D$ of the previous attention contexts:

$$C^D = [Q; C^Q]A^D \in \mathbb{R}^{2\ell \times m}, \tag{3}$$

where $[a; b]$ denotes vertical concatenation. Finally, the coattention context is fed into a bidirectional LSTM to include temporal information. This produces coattention encodings

$$u_t = \text{Bi-LSTM}(u_{t-1}, u_{t+1}, [d_t; c_t^D]) \in \mathbb{R}^{2\ell}, \tag{4}$$

for $t = 1, \ldots, m$. The coattention encoding matrix is the concatenation of all the coattention encodings: $U = [u_1, \ldots, u_m] \in \mathbb{R}^{2\ell \times m}$.

3.3 Decoder

3.3.1 Simple Decoder

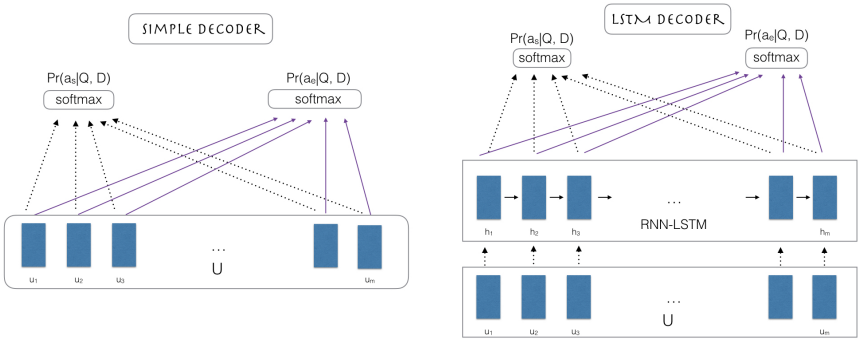
As an initial model, we employed a simple decoder to test the encoder performance, as shown in Figure 3(a). The coattention encoding matrix, U, is projected with weight vector $W^{(S)}$ and then sent through a softmax layer, to produce the probability distribution α over all possible start indexes:

$$\alpha = \text{softmax}(U^\top W^{(S)}) \in \mathbb{R}^m. \tag{5}$$

Likewise, U is projected with weight vector $W^{(E)}$ and then sent through a softmax layer, to produce the probability distribution β over all possible end indexes:

$$\beta = \text{softmax}(U^\top W^{(E)}) \in \mathbb{R}^m. \tag{6}$$

We note that the trainable parameters $W^{(S)}, W^{(E)} \in \mathbb{R}^{2\ell}$ are distinct.



(a) Simple decoder.

(b) LSTM decoder.

Figure 3: Decoders.

3.3.2 LSTM Decoder

The LSTM Decoder, shown in Figure 3(b), aggregates the coattention encodings at each timestep with a LSTM. The hidden state at each timestep of the LSTM is denoted by h_t, for time steps $t = 1, \ldots, m$ which is defined below:

$$h_t = \text{LSTM}_{dec}(h_{t-1}, u_t) \in \mathbb{R}^m. \tag{7}$$

The matrix H is the concatenation of the hidden states at all time steps: $H = [h_1, \ldots, h_m] \in \mathbb{R}^{m \times m}$. An affine transformation is applied to matrix H before the entire output is sent through a softmax layer to produce the start prediction probability distribution α:

$$\alpha = \text{softmax}(H^\top W^{(S)} + b^{(S)}) \in \mathbb{R}^m. \tag{8}$$

Similarly, H is projected through an affine transformation and then sent through a softmax layer to produce the end prediction probability distribution β:

$$\beta = \text{softmax}(H^\top W^{(E)} + b^{(E)}) \in \mathbb{R}^m. \tag{9}$$

We note that the trainable parameters in $W^{(S)}, W^{(E)}, b^{(S)}, b^{(E)} \in \mathbb{R}^m$ are all distinct.

3.3.3 Generating Answers

For both decoders, the answer span (a_s, a_e) is recovered by taking the argmax over the start and end prediction probability distributions α and β:

$$a_s = \arg\max_t(\alpha_t), \qquad a_e = \arg\max_t(\beta_t). \tag{10}$$

4 Experiments

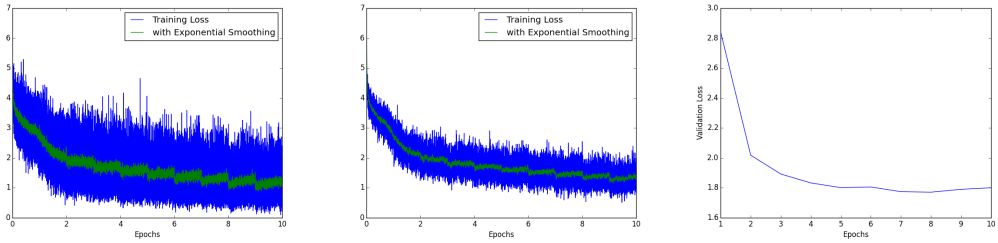
4.1 General Settings

We used the SQuAD dataset to train and evaluate our model. For our distributed word representations, we used pre-trained 100-dimensional GloVe vectors [9].

We used a maximum context paragraph length of $m = 766$ and a maximum question length of $n = 60$. For the LSTMs in the coattention encoder, we used a hidden state size of $\ell = 100$.

All of the LSTM parameters were randomly initialized using Xavier initialization, with an initial hidden state of zero. The weights $W^{(Q)}, W^{(S)}, W^{(E)}$ were also randomly initialized using Xavier initialization, while the biases $b^{(Q)}, b^{(S)}, b^{(E)}$ were initialized to zero.

To train our model, we computed the average cross entropy loss for both start and end indexes, averaged over all training examples, and then took the mean of the two losses to obtain the training loss. We used the ADAM optimizer with a learning rate of 0.001 to optimize our model, and we applied gradient clipping with a maximum gradient norm of 10. We ran our model on a Microsoft Azure Standard NV6 GPU.



(a) Simple Decoder Training Loss (b) LSTM Decoder Training Loss (c) LSTM Decoder Validation Loss

Figure 4: Loss curves.

4.1.1 Simple Decoder Settings

For the simple decoder, we used a batch size of 10 and did not apply any dropout. There were 498,000 trainable parameters.

4.1.2 LSTM Decoder Settings

For the LSTM decoder, we used a batch size of 32 and applied a dropout of 0.1 (keep probability of 0.9) to all LSTMs. There were 3,463,552 trainable parameters.

5 Results

For our evaluation metrics, we used the F1 and exact match (EM) scores. The F1 score computes the average overlap between the predicted and ground truth answers. For examples with multiple ground truth answers, we took the maximum F1 score across all ground truth answers. The EM score computes the percentage of predictions that match at least one of the ground truth answers. The F1 and EM scores of our models on the development set are displayed in Table 1.

Our Models	Dev F1	Dev EM
Simple Decoder	50.7	37.0
LSTM Decoder	54.3	41.2

Table 1: Performance of our models on the development set.

Our model with the LSTM decoder achieves higher F1 and EM scores on the development set than our model with the simple decoder. Based on the performance results shown in Table 2, the model with the LSTM decoder outperforms the baseline model on both the development and the test sets. However, its performance falls short of the performance of the Dynamic Coattention Network model. Figure 4(c) shows that the validation loss for LSTM decoder stops decreasing after reaching 1.8; additional model regularization may help reduce the validation loss.

5.1 Predictions Analysis

We include some observations on the answers generated by the model composed of the coattention encoder paired with the LSTM decoder. The average length of the predictions is 2.97 words with a standard deviation of 4.10 words. The predictions range from 0 to 44 words, with a median length of 2 words. These statistics are comparable to those of the ground truth answers reported next. The average length of the ground truth answers is 2.98 words with a standard deviation of 2.93 words. The ground truth answers range from 1 to 30 words, with a median length of 2 words.

We note that the mean F1 of the Dynamic Coattention Network model [1] exceeds those of previous systems across all question types (“What”, “Who”, “How”, “When”, “Which”, “Where”, “Why”, Other). However, the DCN, like other models, is adept at “When” questions and struggles with the more complex “Why” questions.

Model	Dev F1	Dev EM	Test F1	Test EM
DCN (single model)	75.6	65.4	75.9	66.2
Our Model	54.3	41.2	55.1	42.4
Baseline	51.0	40.0	51.0	40.4
Human	91.0	81.4	91.2	82.3

Table 2: Performance of our best model compared to other models.

Our model composed of the coattention encoder paired with the LSTM decoder learns to associate question types with answer types. For example, "When" questions are associated with numerical answers containing dates. "Who" questions are associated with proper nouns or named entities. We also observed that many answers are empty predictions that result from the end index coming before the start index. In addition, <unk> terms are usually appended to proper nouns; this occurs when the last names of named persons are not found in the vocabulary. The large number of <unk> terms reduces the EM scores. To alleviate this problem, we could pre-process the vocabulary to include more foreign words or take the corresponding words directly from the context paragraph string.

With training, the model learns to associate capitalization to increased probability of being included in the answer. For example, the question, "How was Whitehead 's theory of gravitation received ?" has the ground truth answer "It has been severely criticized". Our model may predict "It" in this case since the word "It" is capitalized but the remaining words in the ground truth answer are not.

6 Conclusion

We investigated whether simplifying the decoder in the Dynamic Coattention Network model could achieve similar performance in question answering on the SQuAD dataset. On the SQuAD test dataset, our best model achieves an F1 score of 55.1 and an EM score of 42.4.

To improve our model, we could replace our current decoder with a multilayer feed-forward neural network. Increasing complexity of decoder structure has been shown to improve performance. To further decrease validation loss, we could add regularization on the parameters and incorporate learning rate annealing, tuning the learning rate in parallel with batch size. We could experiment with GloVe word embeddings of higher dimensionality and incorporate character embeddings. Finally, we could also train an ensemble model of multiple training runs with the same hyperparameters.

References

[1] Caiming Xiong, Victor Zhong, and Richard Socher. (2016). Dynamic Coattention Networks for Question Answering. *arXiv preprint arXiv:1611.01604*.

[2] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. (2016). SQuAD: 100,000+ Questions for Machine Comprehension of Text. *arXiv preprint arXiv:1606.05250*.

[3] Zhiguo Wang, Haitao Mi, Wael Hamza, and Radu Florian. (2016) Multi-Perspective Context Matching for Machine Comprehension. *arXiv preprint arXiv:1612.04211*.

[4] Yang Yu, Wei Zhang, Kazi Hasan, Mo Yu, Bing Xiang, and Bowen Zhou. (2016). End-to-End Answer Chunk Extraction and Ranking for Reading Comprehension. *arXiv preprint arXiv:1610.09996*.

[5] Shuohang Wang and Jing Jiang. (2016). Machine Comprehension Using Match-LSTM and Answer Pointer. *arXiv preprint arXiv:1608.07905*.

[6] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. (2015). Pointer Networks. *In Advances in Neural Information Processing Systems*, pp.2692-2700.

[7] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. (2016). Bidirectional Attention Flow for Machine Comprehension. *arXiv preprint arXiv:1611.01603*.

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

[9] Jeffrey Pennington, Richard Socher, and Christopher D Manning. (2014). GloVe: Global Vectors for Word Representation. *EMNLP*, 14: pp.1532-1543.