CS224N Assignment 4: Reading Comprehension

Christina Kao - chris18@stanford.edu - 05650213
Jason Liu - liujas00@stanford.edu - 05933552
Christopher Vo - cvo9@stanford.edu - 05938491
CodaLab Username: liujas00

Abstract

Building question answering systems with deep learning is a significant application of solving the complex natural language problem of reading comprehension. In our approach, we have analyzed literature on previous work, implemented and improved on specific models described, and compared the various models to analyze the effects of certain aspects of models on performance on the question answering task.

1 Introduction

Reading comprehension is a complex natural language processing task with applications to building knowledge representation and question answering systems. Previous approaches involving deep learning have applied various methods for representing the context and questions and obtaining answer spans. While many of these methods have performed well (around 84 F1, 77 EM on the SQuAD test set), deep learning systems have yet to achieve human levels of accuracy (91 F1, 82 EM).

Given a question and a context paragraph, the goal is to predict the span within the context which contains the answer to the question using a deep learning architecture. Our general approach is to analyze literature on previous work, implement and improve on specific models described, and compare the various models. The models are evaluated by calculating the F1 and Exact Match (EM) scores of the models on a common test set. We have implemented and evaluated three models: a simple attention baseline model, a dynamic coattention model inspired by Xiong et al [3], and a multiple perspective context matching model by Wang et al [2].

2 Background and Related Work

Since the release of the Stanford Question Answering Dataset (SQuAD), a comprehensive dataset containing realistic and challenging questions, many deep learning approaches to machine comprehension have performed well on question answering. Our approaches have been inspired by two models in particular, Dynamic Coattention Networks [3] and Multi-Perspective Context Matching [2].

The Dynamic Coattention Networks approach, published by Xiong, Zhong, and Socher from Salesforce Research, introduces a model that consists of a coattentive encoder to capture question document interaction and a dynamic pointing decoder to estimate the start and end points of the answer span [3]. Within the model, the question and context are encoded using Long Short Term Memory (LSTM) neural networks. An affinity matrix is then computed to capture the relationships between the question and context from which new question and context representations are formed. These representations are then passed to a bidirectional LSTM to capture temporal information. To obtain the answer span, the dynamic decoder applies a Highway Maxout Network model to predict the start and end points of the answer span. The method performs well at an F1 score of 80.4 and an EM score of 71.2 on the SQuAD test set using an ensemble of models. We were motivated to work with this model due to the unique approach to coattention to capture question-context interaction and the strong performance of the model on the question answering task.

In a different approach, the Multi-Perspective Context Matching model, published by Wang, Mi, Hamza, and Florian from IBM Research, presents a model that directly predicts the start and end answer span points by matching the context with question from multiple perspectives [2]. This model calculates a relevancy weight for each context word representing its similarity to each question word and scales the context by these weights. The question and weighted context representations are then passed through bidirectional LSTMs to capture temporal information and obtain the final representations. The context is then compared with the question using different matching strategies from multiple perspectives to form matching vectors. These matching vectors are then passed to bidirectional LSTMs to produce an aggregated vector for each time step. To obtain answer spans, the aggregated vectors are fed through separate feed-forward neural networks to predict the probability distributions of the span start and end points. The model achieved strong performance on the SQuAD test set with 81.3 F1 score and 73.8 EM score. Our approach was inspired by this model because of the interesting mulit-perspective approach to matching context and question and the significant performance of the model on question answering.

3 Approach

3.1 Baseline Model

3.1.1 Baseline Encoder

Our baseline encoder incorporates a simple representation of the question and context using basic attention. To begin, we encode the question Q and context C representations using a bidirectional LSTM to obtain new representations, $Q_{rep} \in \mathbb{R}^{n \times l}$ and $C_{rep} \in \mathbb{R}^{m \times l}$ respectively where n is the max length of a question, m is the max length of a context paragraph, and l is the size of the word embeddings. An attention matrix $\alpha \in \mathbb{R}^{m \times n}$ is then computed by multiplying C_{rep} and Q_{rep} together as a simple measure of similarity between the context and question. The question representation is then scaled by the affinity matrix to produce $\alpha' \in \mathbb{R}^{m \times l}$ that incorporates question-context interaction, and our final output is the concatenation of the context representation C_{rep} and α'.

$$Q, C \rightarrow Bi-LSTM \rightarrow Q_{rep}, C_{rep}$$

$$\alpha = C_{rep} Q_{rep}^T$$

$$\alpha' = \alpha Q_{rep}$$

$$output = [C_{rep}; \alpha']$$

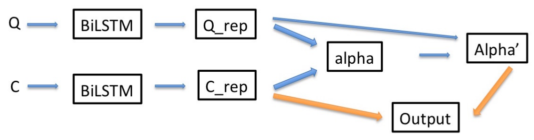

Figure 1: Baseline Model Encoder

3.1.2 Baseline Decoder

To obtain the answer span, our baseline decoder calculates the probability distribution for the start and end indexes in the context. The final output x from the encoder is passed through a single dense layer transformation with weight W and bias b parameters initialized by the Xavier initializer. We use dropout with a keep probability of 0.85. Our final decoder output $\in \mathbb{R}^{m \times l \times 2}$ consists of two separate (unscaled) probability distributions that score each context word index on how likely it is to be the start index and how likely it is to be the end index. To obtain an answer span, we simply pick the context indexes that have the highest scores for the start and end distributions.

$$output = dropout(Wx + b)$$

3.2 Dynamic Coattention Model

3.2.1 Dynamic Coattention Model Encoder

Our implementation of the dynamic coattention encoder follows what is proposed in the Dynamic Coattention Network paper [3], with a minor change being the exclusion of the sentinel vectors for the document and question representations.

The coattention encoder architecture can be summarized by the following graph from [3]: the documentation and question are encoded using the same LSTM, and the final question representation is created through an additional tanh layer. The two attention matrices, one across the document for each word in the question (A^Q), and the other across the question for each word in the document (A^D), are generated through normalizing the product of the document and question representations. These two attention matrices are then combined with the document and question representations through multiplications and concatenations, and passed through a bidirectional LSTM to create the final coattention output.

$$Document, Question \rightarrow LSTM \rightarrow D \in \mathbb{R}^{l \times (m+1)}, Q'$$

$$Q = tanh(WQ' + b) \in \mathbb{R}^{l \times (n+1)}$$

$$L = D^T Q$$

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

$$C^Q = DA^Q$$

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

$$[D; C^D] \rightarrow Bi - LSTM \rightarrow U$$

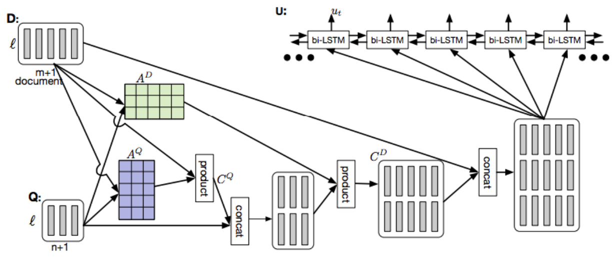

Figure 2: Coattention Encoder from Xiong et.al (2017)

3.2.2 Baseline Decoder

We used the same decoder as that in the baseline model. See section 3.1.2 for details.

3.3 Multi-Perspective Context Matching Model

3.3.1 Multi-Perspective Context Matching Model Encoder

This model has five major components.

Word Representation Layer: In the paper, Wang et al chose to use pre-trained GloVE vectors concatenated with character-composed embeddings for each question and context word [2]. In our experiment, we decided not to include the character-composed embeddings, as the authors' ablation test reveals that not doing so results in only the slightest penalty to F1 and EM scores. This is fairly routine step in deep learning models that allows us to start with a general understanding of our data.

Filter Layer: In this layer, we calculate the cosine similarity between each pair of question and context embeddings. Then, we scale each context embedding by the maximum of all similarity values between that specific context embedding and all question embeddings. This layer exists to filter out any context embeddings that have absolutely no relation to any of the question words; if the max cosine similarity is 0, then the embedding is zeroed out.

Context-Representation Layer: In this layer, we run a BiLSTM separately over the question embeddings and our scaled context embeddings. This gives us forward and backward outputs for both questions and contexts. As the name of the layer implies, running a bidirectional LSTM allows us to capture the overall context of the individual words in both question and context. So we have $\overrightarrow{H}^c, \overleftarrow{H}^c \in R^{N \times L}$ and $\overrightarrow{H}^q, \overleftarrow{H}^q \in R^{M \times L}$

Multi-Perspective Context Matching Layer: This is the most important part of the model. As stated by Wang et. al, "The goal of this layer is to compare each contextual embedding of the passage with the question with multi-perspectives." There are two directions in which matching is made. The first is dimensional weighting matching with:

$$m = f_m(v_1, v_2; W) \tag{1}$$

Here, W is a matrix of size $R^{P \times d}$, where l is the number of perspectives and d is the dimension of the hidden state from the previous layer. v_1 and v_2 are both d-dimensional vectors. The following is the derivation for m:

$$m_k = cosine(W_k \odot v_1, W_k \odot v_2) \tag{2}$$

As the paper suggests, we use three matching strategies to compare each contextual passage embedding with the question.

$$\overrightarrow{m}_j^{full} = f_m(\overrightarrow{h}_j^c, \overrightarrow{h}_M^q; W^1) \tag{3}$$

$$\overleftarrow{m}_j^{full} = f_m(\overleftarrow{h}_j^c, \overleftarrow{h}_1^q; W^2) \tag{4}$$

$$\overrightarrow{m}_j^{max} = \max_{i \in (1...M)} f_m(\overrightarrow{h}_j^c, \overrightarrow{h}_i^q; W^3) \tag{5}$$

$$\overleftarrow{m}_j^{max} = \max_{i \in (1...M)} f_m(\overleftarrow{h}_j^c, \overleftarrow{h}_i^q; W^4) \tag{6}$$

$$\overrightarrow{m}_j^{mean} = \frac{1}{M} \sum_{i=1}^{M} f_m(\overrightarrow{h}_j^c, \overrightarrow{h}_i^q; W^1) \tag{7}$$

$$\overleftarrow{m}_j^{mean} = \frac{1}{M} \sum_{i=1}^{M} f_m(\overleftarrow{h}_j^c, \overleftarrow{h}_i^q; W^2) \tag{8}$$

$$m_j = [\overrightarrow{m}_j^{full}; \overleftarrow{m}_j^{full}; \overrightarrow{m}_j^{max}; \overleftarrow{m}_j^{max}; \overrightarrow{m}_j^{mean}; \overleftarrow{m}_j^{mean}] \tag{9}$$

Aggregation and Prediction Layer: The resulting matching vectors for each contextual paragraph embedding are then fed into a BiLSTM. The forward and backwards outputs are concatenated.

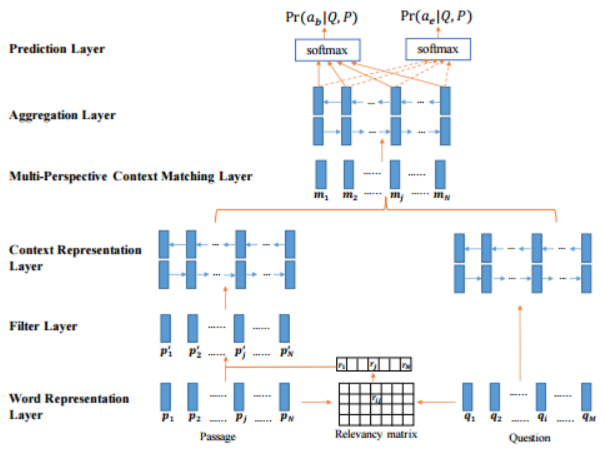

Figure 3: Multi-Perspective Matching Architecture from Wang et.al (2016)

3.3.2 Baseline Decoder

We used the same decoder as that in the baseline model. See section 3.1.2 for details. This decoder is basically encapsulated by our Aggregation and Prediction Layer.

4 Experiments

The three models we implemented were run on the same validation set to compute and compare performance. The main dataset used for both training and testing is the Stanford Question Answering Dataset (SQuAD), a large dataset containing realistic and challenging questions. The format of the answers are a span in the context which is desirable for our approach. The dataset contains a diverse set of questions requiring different kinds of logical reasoning to deduce the answers and is widely used in other deep learning approaches for machine comprehension.

In terms of training our models, we calculated loss by taking the softmax cross entropy of our unscaled probability distributions calculated by the decoder, and matched the results with the correct labels.

To evaluate our models, we computed and compared F1 and Exact Match (EM) scores for each model on the validation set, two widely used metrics for SQuAD proposed by Rajpurkar, Zhang, Lopyrev, and Liang from Stanford University in the original SQuAD paper [3]. As described in the literature, F1 score measures the average overlap between the prediction and ground truth answer. The prediction and ground truth are treated as bags of tokens, and the F1 score is computed. The maximum F1 over all ground truths for a question is taken, and the final F1 score is the average over all questions. On the other hand, exact match measures the percentage of predictions that match any of the ground truths exactly.

4.1 Baseline Model

Our baseline model was implemented and evaluated on the validation set to obtain a base performance threshold. The goal was to compare the baseline model with more advanced models to analyze the effects of more sophisticated attention features on performance of the question answering task. Our simple baseline model achieved an F1 score of 51.8 and an EM score of 21.6 on the validation set used to evaluate our models. Performance for our baseline model plateaued after the third epoch. Additionally, training loss decreased over each epoch and batch iteration while the normalized gradient decreased slightly. Dropout with a keep probability of 0.85 was applied at the decoding layer. A learning rate of 0.001 was used, chosen based on empirical testing.

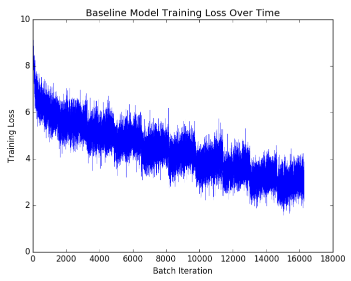

Figure 4: The training loss over each batch iteration and epoch for the Baseline model.

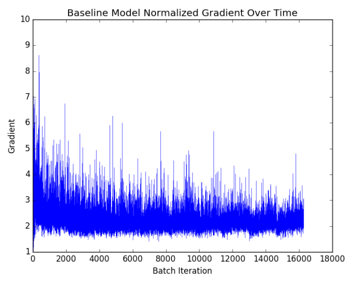

Figure 5: The normalized gradient over each batch iteration and epoch for the Baseline model.

4.2 Dynamic Coattention Model

There are four main differences between the coattention encoder and our baseline encoder: (1) coattention model uses single direction LSTM and has an additional non-linear projection layer for question representation, whereas the baseline uses bi-directional LSTM to encode (2) coattention model adds an additional sentinel vector to the original documentation and question representations (3) instead of just calculating the representation for the question in light of each word of the document (baseline), the coattention model also computes the reverse (4) coattention model passes the final representation through a bi-directional LSTM.

We divided the coattention encoder into separate components based on the differences mentioned above, and tested out various combinations by adding components to the baseline encoder or subtracting components from the coattention encoder one by one. We keep (1), (3) and (4), as there are no noticeable improvements of BiLSTM encodings over LSTM, and the combination of (3) and (4) provide more fine-grained attention representations, and are also the main differentiators between our baseline encoder and the coattention encoder; whereas the addition of sentinel vectors to our model harmed performance, resulting in little to no learning and hence a stagnant loss, which could be due to the simplicity of our decoder, therefore we decided not to include it.

For our final run of the coattention model on the full dataset, we apply dropout with a keep probability of 0.85, and choose a decay rate of 0.3 over 1600 steps for the learning rate, because each of our epoch is 1600 steps, and based on previous runs, the model stopped learning pass the 1st epoch. We used Adam Optimizer

Given the experiments above, the model achieved an F1 score of 53.9 and EM score of 21.0. The training loss converged after about 6000 batch iterations despite an increase in the normalized gradient, which could be due to us using too low of a decay rate.

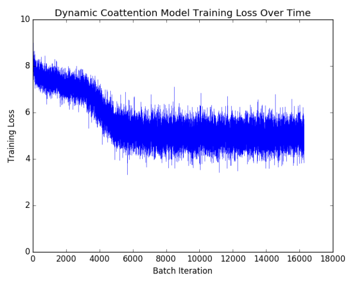

Figure 6: The training loss over each batch iteration and epoch for the Dynamic Coattention model.

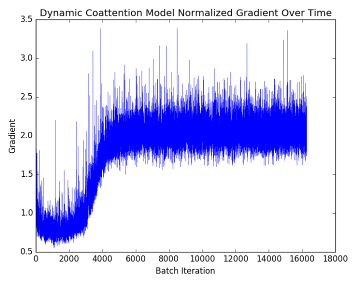

Figure 7: The normalized gradient over each batch iteration and epoch for the Dynamic Coattention model.

4.3 Multi-Perspective Context Matching Model

For the multi-perspective model, the only implementation difference, as previously discussed, was that we did not include character embeddings. Otherwise, our model followed the Wang paper fairly closely. We did differ in hyper parameters, however. In the interest of cutting down training time and not exceeding GPU space, we used only 1 perspective instead of 50 perspectives, and our GloVE vectors were only 100-dimensional instead of 300-dimensional ones[2]. The authors' benchmark tests indicated that the difference between 50 perspectives and 1 perspectives was about a reduction of 4 percent in EM score[2]; given more time, we would like to train with that many perspectives and observe the boost ourselves. Similarly to the authors did, we also implemented dropout at every layer. However, while the authors used a ratio of 0.2, we used a ratio of 0.1 because we had fewer parameters to work with. We used a learning rate of 0.01 with a decay rate of .5 every epoch, as past testing indicated that the model learned a lot within the first epoch, and would need significantly lower learning rates in later stages of training. We used Adam Optimizer.

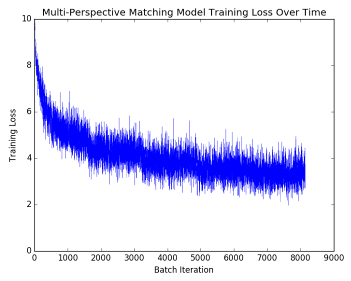

Figure 8: The training loss over each batch iteration and epoch for the Multi-Perspective Matching model.

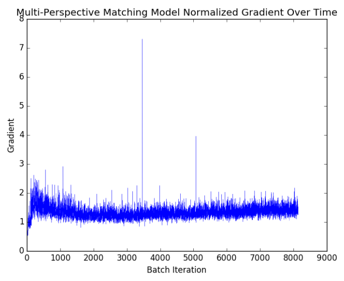

Figure 9: The normalized gradient over each batch iteration and epoch for the Multi-Perspective Matching model.

4.4 Analysis and Comparison

For our models, the Multi-Perspective Matching model had the highest performance with 68.2 F1, 36.0 EM while the Dynamic Coattention did second best with 53.9 F1, 21.0 EM and our Baseline model performed the worse with 51.8 F1, 21.6 EM on the val set. In addition, our best model, Multi-Perspective Matching, achieved 46.7 F1 and 33.8 EM on the test set used as the final submission to the leaderboard. This suggests that more complex attention correlates with better performance; however, models with complex attention often take longer to learn as their initial F1 and EM scores are much lower in the first few epochs. Additionally, all of our models have a large gap between F1 and EM scores which may potentially be due to the simple decoder function that each model uses to predict span start and end points.

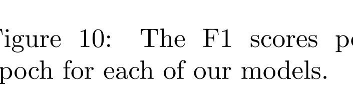

Figure 10: The F1 scores per epoch for each of our models.

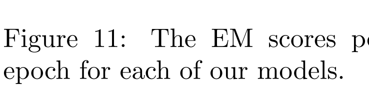

Figure 11: The EM scores per epoch for each of our models.

Additionally, we analyzed the performance of our best model, the Multi-Perspective model, on different types and lengths of questions. The model seems to perform well on questions beginning with the words "When" and "In" because these questions generally have a simple answer such as a single date or location. For example, on the question "When did Robert Crispin go up against the Turks?", our model predicted the simple answer "1060s" which is part of the valid answer set of "1060s" and "In the 1060s". On the other hand, our model performs poorly on questions beginning with "Why" as these questions can be more complex and have variable length answers. On the question "Why are ctenophores extremely rare as fossils?", our model predicts the answer "Because of their soft , <unk> bodies , <unk> are extremely rare as fossils , and fossils that have been interpreted as <unk> have been found only in lagersttten , places where the environment was exceptionally suited to preservation of soft tissue" where "<unk>" is an unknown word that does not appear in the vocabulary. The valid answer set for the question consists of "Because of their soft, gelatinous bodies" and "their soft, gelantinous bodies" which shows that while our answer contains the relevant information, it produced an answer that was too long.

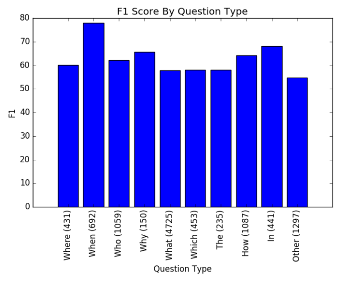

Figure 12: The F1 scores by question type for our best model.

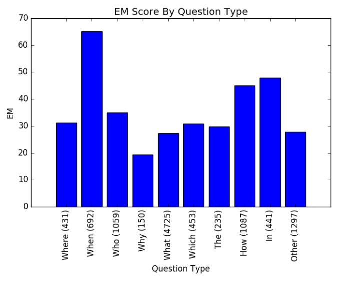

Figure 13: The EM scores by question type for our best model.

When analyzing question length, we found that generally, performance decreases as question length increases except for really short questions of length one to five words. This may be that for sentences of this size, our model has a hard time fully capturing the context representation of the question and relating the paragraph to it. For example, on the short question "What was Galileo Ferraris?", our model predicted the answer "Italian physicist Galileo Ferraris , but decided Tesla 's patent would probably control the market" when the valid answer set contained "physicist" and "Italian physicist". In this instance, our model has the correct idea, but struggles to map a more specific section of the paragraph back to the question.

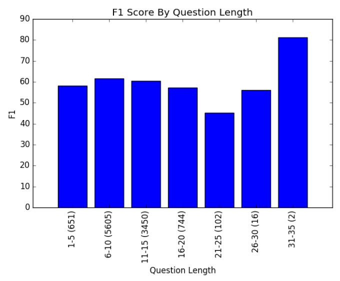

Figure 14: The F1 scores by question length for our best model.

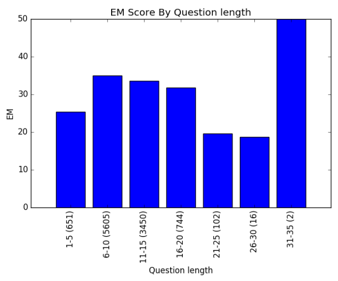

Figure 15: The EM scores by question length for our best model.

5 Conclusion and Future Work

From our approach, we learned that most of the improvements in performance for question answering models come from better word embedding vectors and stronger capture of question-context interaction as our models with the most sophisticated attention functions performed best. For future work, we would like to implement a more complex decoder to improve span prediction. One of the major differences between the original Dynamic

Coattention model and ours was using the Highway Maxout Network decoder which may be a key reason for the lack of relative performance of our Dynamic Coattention implementation. Better decoders may also help with the gap between our F1 and EM scores for our models. Additionally, future work could include more hyperparameter tuning and training ensembles of our models to further boost performance. Finally, combining different features of the models we compared and analyzed may also lead to a better performing model than any individual model we implemented.

6 Team Contributions

As a team, we worked very well together. In addition to group programming, Christina and Jason worked mainly on implementing the Baseline, Dynamic Coattention, and Multi-Perspective Matching models while Christopher worked mainly on implementing the qa-answer script and visualization code, graphs, and poster. Everyone also contributed equally to the paper.

7 References

[1] Rajpurkar, Pranav, et al. ”Squad: 100,000+ questions for machine comprehension of text.” arXiv preprint arXiv:1606.05250 (2016).

[2] Wang, Zhiguo, et al. ”Multi-Perspective Context Matching for Machine Comprehension.” arXiv preprint arXiv:1612.04211 (2016).

[3] Xiong, Caiming, Victor Zhong, and Richard Socher. ”Dynamic Coattention Networks For Question Answering.” arXiv preprint arXiv:1611.01604 (2016).