Coattention Model for Question Answering

Tina Vachovsky
Department of Computer Science
Stanford University
Stanford, CA 94305
tvachov@stanford.edu
CodaLab Username: tvachov

Marie Vachovsky
Department of Computer Science
Stanford University
Stanford, CA 94305
mvachovs@stanford.edu

Abstract

The recently released Stanford Question Answering Dataset (SQuAD) provides a unique version of the question-answer problem that more closely relates to the complex structure of natural language, and thus lends itself to the expressive power of neural networks. We explore combining tested techniques within an encoder-decoder architecture in an attempt to achieve a model that is both accurate and efficient. We ultimately propose a model that utlizes bidirectional LSTM's fed into a coattention layer, and a fairly simple decoder consisting of an LSTM with two hidden layers. We find through our experimentation that the model performs better than combinations of coattention with both our simpler and more complex decoders. We also find that it excels at answering questions where the answer can rely on marker words or structural context rather than abstract context.

1 Introduction

We explore the problem of question answering, i.e. given a question and some context containing the answer, find that answer within the context. Question answering systems can be used for information retrieval tasks, which themselves can be applied in several fields. Of course, we have the typical Google search example. But the systems can additionally allow a company to search through client data, a doctor to search through medical records, or a movie goer to ask for a recommendation. Solving the problem efficiently and accurately has the potential to improve these tasks and the many others like them.

The problem has been solved with varying success utilizing deep learning. Most successful models employ some form of attention, and many take advantage of bidirectional LSTMs to encode and decode the data [1][2][3]. The best models achieve about an 80% F1 score and 75% EM score, though they tend to be slow to train [1][2][3].

We utlize SQuAD, introduced in [4]. SQuAD contains around 100,000 question-answer pairs, each with a corresponding context paragraph extracted from Wikipedia. SQuAD provides a particularly challenging problem because, unlike prior datasets, it does not provide a list of answer choices and instead requires the model to choose from any possible span in the context.

With this dataset, we aimed to determine the span of an answer that corresponds to a question, given a context paragraph containing the answer. We specifically focused on the common elements we saw across successful models (encoder/decoder architecture with attention), and worked to combine them to solve the question answering problem. We explored various of these combinations in an attempt to create a novel model that remains grounded in tested methods. We also aimed for our model to be less time intensive them some of the past successful models, but still perform well.

2 Related Work

As mentioned earlier, deep learning algorithms have been fairly successful at solving the problem of question answering. We specifically take direction from two specific models: Dynamic Coattention Networks and Match LSTM's with Answer Pointing [1][2]. The dynamic coattention model represents the codependent nature of the question and context by utilizing attention to merge the question and context representations. It further utilizes a dynamic pointing decoder to iterate over the answer spans. It achieves an F1 score of about 75 percent and an EM score of about 66 percent with a single model (and higher with an ensemble) on the test set. However, the multi-layer dynamic decoder is lengthy to run on large datasets [1].

The Match LSTM model similarly represents utlizes attention to represent the context's dependence on the question, but instead merges them through a match LSTM. It also similarly utilizes a pointing decoder. However, the decoder relies on fewer hidden layers and therefore is less lengthy to run [2]. The model ultimately achieves an F1 score of about 70 percent and an EM score of about 60 percent on a single model [2].

3 Approach

We first established a general architecture over which to experiment with different encoders and decoders. That is, we established the input to the encoder, the output from the decoder, and loss and training operations.

Each question and each context paragraph is represented as a variable-length vector, where each entry in the vector corresponds to a word. The question vectors are then padded to max question length (defined in Table 1 below), and the context figures are padded to max context length. The corresponding GLOVE word vector to each vector entry is used to create matrix embeddings of both the questions and the contexts. Corresponding batches of question and context paragraph matrices serve as input to the encoder.

The decoder outputs two vectors, where each entry corresponds to the probability of the answer starting (for the first vector) or ending (for the second vector) at the entry's index. We calculate the cross entropy loss between these two vectors, and the true start and end indices. We use an Adam optimizer to train the loss, and apply dropout for all encoders used.

3.1 Encoder and Coattention

3.1.1 Exploration

Within our proposed encoder/decoder architecture, we first focused on choosing an adequate encoder/attention combination. Our baseline employed a simplified version of the coattention method used in [1]. The simplified context calculated represented question summaries corresponding to each word of the document. Our first encoder followed the following flow: encode questions using a bi-LSTM. Encode context using a second bi-LSTM (with a different scope). Use the backward outputs of these LSTMS, Q and P, to calculate the attention context of the document in reference to the question. Concatenate the attention context with P, and run the concatenation through a non-hidden layer. Use the result in the decoder.

To explore different methods of attention, we then implemented the global attention described in [5] in place of the coattention described above. We found that our baseline performed better (see evaluation) with the coattention above, and thus we decided to implement the encoder/coattention in [1] as the encoding step of our encoder/decoder model.

3.1.2 Final Encoder and Coattention

Figure 1 shows the final encoder flow of our model (before coattention). After questions and paragraphs are transformed into matrices of word vectors, both pass through a bi-LSTM encoder (within the same scope). To follow [1] more closely, but still retain our biLSTM encoder, we concatenated the backward outputs with the forward outputs for both the paragraph outputs, P, and the question

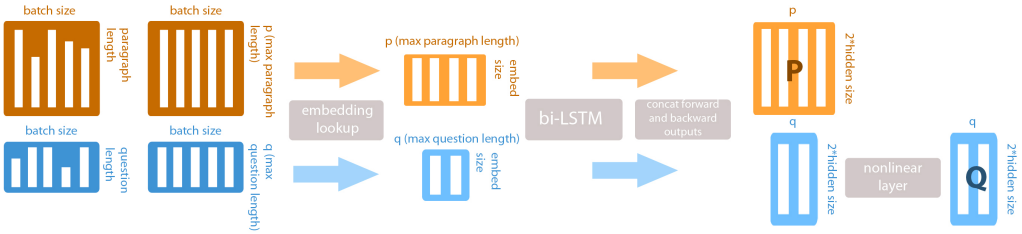

Figure 1: Encoder for Final Coattention Model

outputs, Q', instead of only using the backward outputs. We apply a nonlinear tanh layer over Q' to calculate Q.

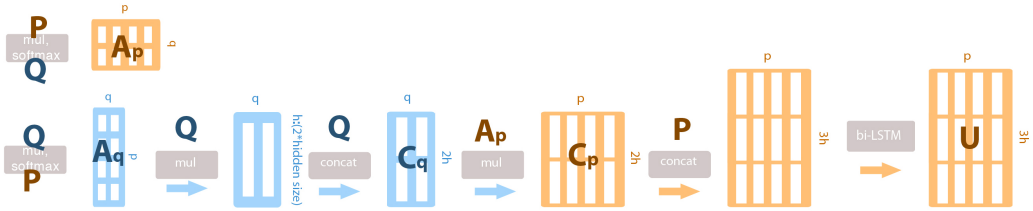

Figure 2: Coattention Encoder for Final Coattention Model

Figure 2 shows the final coattention flow of our model, where P and Q are the inputs. We again calculated the question summaries corresponding to each word of the paragraph, Cq. We then further compute the summaries of the these attention contexts Cq corresponding to each word of the document. The output is Cd, a matrix summarizing question and paragraph information. Cd is then encoded through a bi-LSTM (with different scope than the previous bi-LSTM), outputting U. Again, we concatenate the forward and backward outputs. We then feed U into our decoder.

3.2 Decoder

3.2.1 Exploration

We began with a naive decoder, where we passed our output from the encoder architecture into two recurrent neural networks of different scope. Taking the softmax of one yielded a vector of probabilities, where the probability at one index of the vector corresponded to the probability of that index being the start of the answer span. Taking the softmax of the other vector yielded the probabilities for the end of the answer span. Although we quickly abandoned the naive approach, it allowed us to establish a good framework for decoder input and output, as well as a good framework for our loss input.

From there, we explored two different decoders: a simple LSTM decoder, and an answer pointing decoder. The answer pointing decoder was taken from [2], and chosen because, in terms of complexity, it falls between the simple LSTM decoder and the dynamic pointing decoder from [1]. We hypothesized that it could function as a compromise between the accuracy of the model in [1] and the efficiency of the simple LSTM decoder. In practice, however, the simple LSTM decoder performed better. We therefore moved forward with an LSTM decoder in our final model, and used the number of hidden layers to control complexity.

3.2.2 Final Decoder

Figure 3 shows the flow of our final decoder. The decoder takes in U, the encoded coattention context, as input, and outputs probabilities for the start index and end index of the answer span (as described in exploration). U is first transformed using two hidden layers into logits for the start of

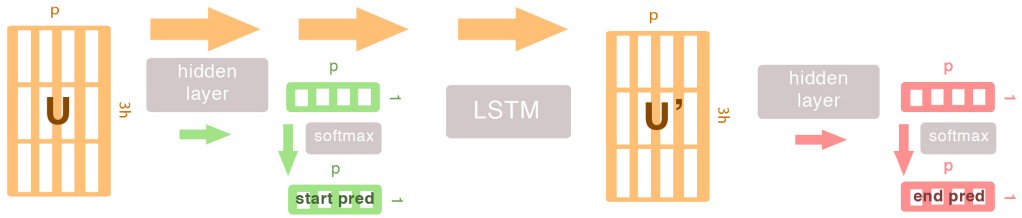

Figure 3: Decoder for Final Coattention Model

Table 1: Hyperparameters across models

	BASELINE	COATTENTION W/ ANSWER POINTING	COATTENTION W/ 1 HIDDEN LAYER	COATTENTION W/ 2 HIDDEN LAYERs
Learning Rate	.001	.001	.001	.001
Dropout	.15	.15	.15	.50
Encoder State Size	100	200	200	200
Max Question Length	30	30	30	30
Max Context Length	500	500	500	500

the answer. Softmax is applied on the output to calculate the answer start probabilities. The same input U is then run through an LSTM to produce U', which passes through a two more hidden layers to produce the logits for the end of the answer. Finally, softmax is applied on the logits to calculate the answer end probabilities. These probabilities become the input for the cross-entropy loss. We refer to this final model as Coattention with 2 Hidden Layers when reporting our results.

The first iteration of this model applied only one hidden layer betwen U and answer start, as well as between U and answer end. We refer to this version as Coattention with 1 Hidden Layer when reporting our results.

4 Experiments

4.1 Dataset

We train our model on question-answer pairs and their corresponding context paragraphs from SQuAD [4]. Questions tend to have 10 to 20 words, and rarely exceed 30 words. Paragraphs tend to have 100 to 200 words, and rarely exceed 500. Answers tend to be under 10 words. These limits influence our the set lengths of our padded question and context vectors: we set question vectors to a maximum length of 30, and context vectors to a maximum of 500.

The dataset itself consists of a wide range of question and answer types. Part of it's novelty comes from the fact that all questions are syntactically different than their corresponding answers - making the problem more closely resemble real world language and applications.

We separate the dataset into training and development sets to develop our models. We then run our final model on a hidden test set.

Table 2: Scores across models

	BASELINE	COATTENTION W/ ANSWER POINTING	COATTENTION W/ 1 HIDDEN LAYER	COATTENTION W/ 2 HIDDEN LAYERS
F1	.39	.56	.55	.67
EM	.2	.38	.36	.46
Harmonic Mean	.26	.45	.44	.55

Table 3: Final Model Performance across Question Types

	Full Dataset	"Who"	"What"	"Where"	"When"	"How"	"Why"	Other
F1	64.47	65.75	62.79	61.69	75.30	65.12	46.06	62.88
EM	51.31	56.72	48.32	48.03	66.67	52.56	22.78	49.59

4.2 Results

4.2.1 Hyperparameter Settings

In Table 1 we report the best-performing hyperparameters for each of our four models. We started the learning rate fairly low at .001 and utilized annealing across all models to prevent falling into local optima. A dropout of .15 worked well for the Baseline, Coattention with Answer Pointing, and Coattention with One Hidden Layer models, and increasing the dropout led to decreases in performance. However, we increased the dropout to .50 for the Coattention with Two Hidden Layers because the added hidden layer led the model to overfit with the lower dropout. We kept the state size at 100 for all the models with coattention. We do so to avoid overflowing the memory in the hidden states, as we concatenate the forward and backward states of the encoding LSTM to feed into the coattention, and therefore need a smaller initial state size. We did not concatenate in the baseline, and thus kept the hidden state at size 200. As we train all models on the same dataset, we keep max question length and max context length constant throughout datasets. The values were chosen by creating histograms of data lengths, and choosing values that were shorter than very few examples.

4.2.2 Performance

Table 2 shows the performance of the four models with the hyperparameters indicated in Table 1. Performance is measured over 100 samples from the validation set. We use F1, EM, and the harmonic mean of the two, as our evaluation metric, where F1 is the harmonic mean of precision and recall, EM is the count of exact matches. Scores are evaluated on correct words guessed within the answer span, that is an incorrect start and end guess can receive a nonzero score if the span it represents contains part of the correct answer. F1 For all models excluding Coattention with 1 Hidden Layer, the performance reported is the best performance. Performance reported for Coattention with 1 Hidden Layer is the average performance, as the model had a range of 20% difference between performance on different runs.

Coattention with 2 Hidden Layers scored highest for both F1 and EM, and did so by a margin of 11% for the F1 and 8% for the EM. Coattention with Answer Pointing scored the second highest scores, but was higher than Coattention with 1 Hidden Layer by only a 1-2% margin. As Coattention with Answer Pointing is much more time intensive than the Coattention with 1 Hidden Layer

After model comparison on the validation set, we evaluated our best performing model, Coattention with 2 Hidden Layers, on the development and test sets. On the development set, the model received an F1 score of 64.16% and an EM score of 51.02%. On the test set, the model received an F1 score of 64.77% and an EM score of 52.54%.

4.2.3 Question-level Analysis

We look deeper into our final model's performance by examining the types of questions it performs well on. We specifically analyze its performance on the development set, as this set represents our

Table 4: Final Model Performance Based on Length

	Answer Length <5	Answer Length >5
F1	69.80	46.74
EM	60.40	21.03

unseen data. We ultimately find that our model performs best on questions where the answers can be discovered either through marker words or structural format by analyzing by both question category and length.

The categories consist of "Who," "What," "Where," "When," "How," and "Why," and include an "Other" category for all questions lacking those key words. The results are reported in Table 3. We find that our model performs within 2 percent of the overall average F1 (and within 5 percent of the overall average EM) for all categories except "When" and "Why," where it performs 10 percent better (for both F1 and EM) and 20 percent (30 percent for EM) worse, respectively.

More broadly, we find that our model does well when it has specific word markers that indicate the start of an answer. This tends to be the case in the "When" category. For example:

"When was the third period of high viewership for the doctor who series?"

"After the series' revival in 2005"

The answer to the question is marked by both the word "After" and the year format "2005."

On the other hand, the answers to "Why" questions are much less likely to be marked by specific words or words in a specific format. Our model does correctly guess "Why" questions where the answer follows marker words; for example, words like "because." However, it has difficulty with questions given a context like the following:

"Why was the student group called "the Methodists?"

"The movement which would become The United Methodist Church began in the mid-18th century within the Church of England. A small group of students, including John Wesley, Charles Wesley and George Whitefield, met on the Oxford University campus. They focused on Bible study, methodical study of scripture and living a holy life..."

In this case, the answer should be "they focused on Bible study, methodical study of scripture, and living a holy life." However, attaining this answer relies on a logical jump on the part of the reader that is too complex for our algorithm.

When looking instead at lengths, we find that our model excels when faced with shorter answers and suffers when faced with longer ones. As an example, Table 4 shows performance when answer length is less than or greater than 5. The logic as to why is similar to the logic behind the category performance. Shorter answers tend to be more direct in their placement and context, whereas longer ones may be more abstract. For example, compare the following "How" question and answer with the "Why" question above.

"How are nuclear forces transmitted?"

"as gluons" (from the sentence "Here the strong force acts indirectly, transmitted as gluons")

In this example, the answer immediately follow a repetition of words from the question. In other words, without even knowing the context, one can guess the answer based on structure. Again, in the Methodist question, one had to make the leap of logic based on the actual context.

5 Future Work

In the future, we would like to first focus on deeper comparison between models. Specifically, we want to analyze the answer choices of all of our models using the methods we used for our final

model, and see if any of the models perform well on questions that our final model struggles with. From there, we can work on improving the weaker segments of our final model.

Additionally, we want to further experiment with the hyperparameters of our existing models. We would like to explore whether Coattention with 2 Hidden Layers is really the best performing model, or whether the other two models can reach similar scores with different hyperparameters. We would begin by running a hyperparameter search on each model.

Finally, we would like to create more combinations of encoders with attention and decoders, and test them against our existing model. We specifically would like to try combining our final coattention method with a more complex decoder, to work towards achieving higher F1 and EM scores.

6 Conclusion

We ultimately acheived the highest F1 and EM scores combining common approaches we found across previous successful models: bi-LSTM encoders and decoders with a coattention layer in between. Our most successful coattention layer took into account attention for both questions in relation to contexts, and vice versa. While we hoped a combination of this layer with a complex decoder would improve our scores, we found that it performed better with a simpler LSTM decoder.

In the future, we would like to explore other, more complex decoders, and experiment further with the hyperparameters of all our models.

7 Contributions

Marie Vachovsky wrote the code for the encoder, coattention, and decoder (both the LSTM decoder and the answer pointing decoder). She also created the poster and the visual representations of the model. Tina Vachovsky wrote the general architecture code (reading in data, training, validation, qa_answer.py, etc). She also handled all CodaLab submissions and wrote the code for the question-level analysis experiments. Both talked through all coding choices high level, and debugged both their own and the other's code. Both tuned hyperparameters, and both wrote the paper.

References

[1] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for questionanswering.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] Zhiguo Wang, Wael Hamza, and Radu Florian. Bilateral multi-perspective matching fornatural language sentences.arXiv preprint arXiv:1702.03814, 2017.

[4] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100,000+ questions for machine compre-hension of text. InEmpirical Methods in Natural Language Processing (EMNLP), 2016.

[5] Luong, Thang, Hieu Pham, and Christopher D. Manning. Effective Approaches to Attention-based Neural Machine Translation. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, 2015.

8 Appendix

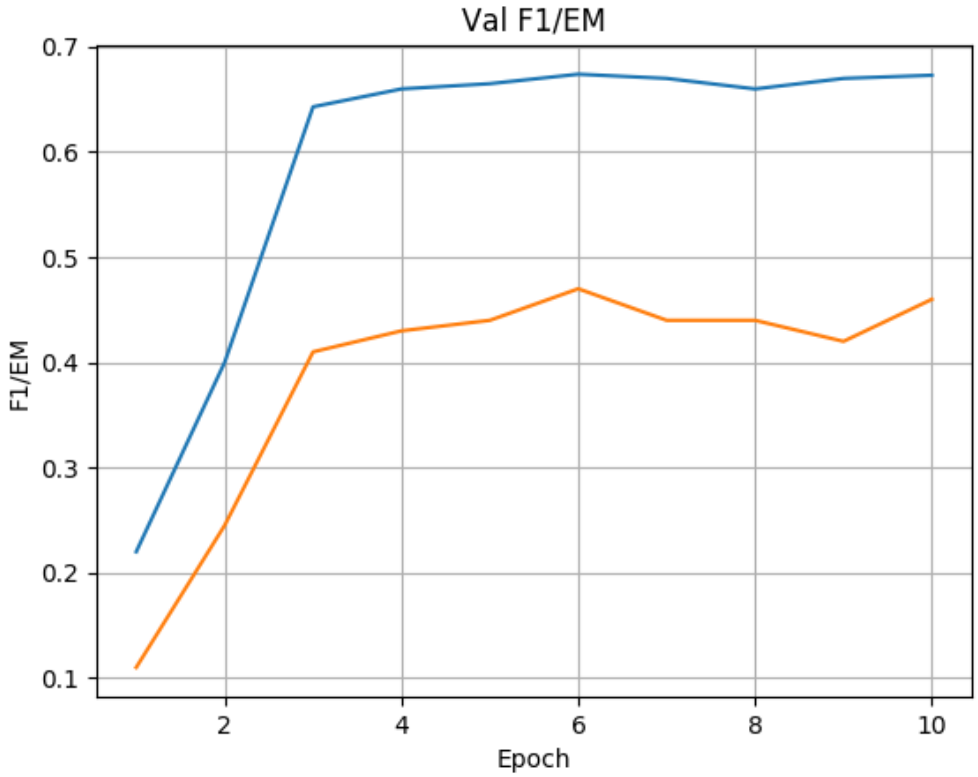

Figure 4: F1 and EM scores for coattention model with 2 hidden layer LSTM over 100 samples from the validation set for 10 epochs

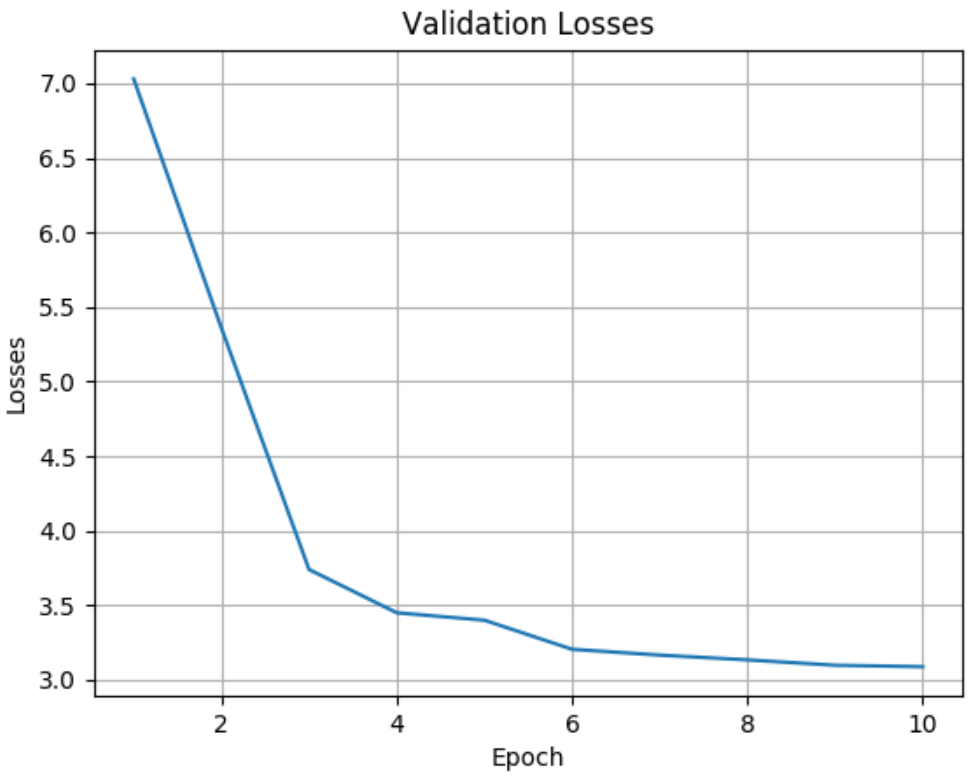

Figure 5: Validation loss for coattention model with 2 hidden layer LSTM for 10 epochs