An End-to-End Neural Architecture for Reading Comprehension

Meredith Burkle
Department of EE
Stanford University
mnburkle

Miguel Camacho
Department of CS
Stanford University
somosmas

Ned Danyliw
Department of EE
Stanford University
edanyliw

Codalab username: ndanyliw

Abstract

Machine reading comprehension is an on-going area of research in both industry and academia. In this paper, we present an end-to-end model combining elements of state-of-the-art question answering systems trained on the Stanford Question-Answering Dataset (SQuAD). Using this architecture, we achieve an **F1 of 61.5% and an EM of 48.3%**.

1 Introduction

With various applications in news aggregation and summarization, text analysis, and of course natural language understanding, machine reading comprehension has long been a prominent area of research in natural language processing. Although 'machine reading comprehension' encompasses a broad range of tasks, in our work we will focus specifically on question-answering. In particular, the challenge we undertook was to construct a reading comprehension agent capable of answering questions specifically relevant to a given body of text such as an article or paragraph.

Previous approaches to this problem have typically relied heavily on complex linguistic analyses and feature engineering using domain-expertise. However, in recent years the increased computational power and availability of substantial question-answer datasets have allowed neural network-based models to greatly surpass earlier benchmarks for success.

In this paper, we will describe our efforts to build on state-of-the-art models such as Dynamic Coattention Network [2], Ptr-Net [3], and match-LSTM with Answer-Pointer [4]. Ultimately, we will compare the F1 and exact match (EM) scores of our hybrid reading comprehension agent with existing cutting-edge models to get a metric for our model's performance.

2 Related Work

Before detailing on our own hybrid model, let us review the existing models from which we drew inspiration and to whom we owe much credit.

Dynamic Coattention Networks

In 2016, Xiong et al. put forward a model that brings together a coattention encoder with a dynamic pointer decoder that iterates over potential answer spans to identify the answer within the given context.

Specifically, their encoding brings together question and context paragraph representations into attention matrices (A^Q and A^D in Figure 6 below) which then come together to form a co-dependent representation of the question and context (C^D) that is ultimately fed into a bi-directional LSTM (biLSTM) to fuse temporal information into their final coattention (U).

We soon switched to this encoding mechanism after implementing our Baseline++ model, as the coattention architecture is slightly more intricate in the encoding stage and underpins multiple high-performing models.

Of course, Xiong et al. go on to decode their question-context coattention using a dynamic decoder which employs a Highway Maxout Network (HMN) based on the Highway Networks work of Srivastava et al. [5] and the Maxout Networks work of Goodfellow et al. [6], however we did not focus on their decoding architecture so we will refer the reader to their paper for further details.

Answer-Pointer / Ptr-Net

In their 2017 work on machine comprehension, Wang et al. employed a match-LSTM encoder together with an answer-pointer decoder.

In their (boundary) decoder they use multiple non-linearities on top of fully-connected layers that use the hidden layers of an LSTM as input to predict the start and answer tokens all while passing along the transformed hidden layers as input into the next time-step. This framework is based on the Ptr-Net word done by Vinyals et al. shown in Figure 7.

We improved our model by borrowing this framework for our decoder (refer to Figure 1 for a much more clear description of this process). As for their matchLSTM-layer encoding, again we will refer the reader to their paper.

3 Model

When designing the model we took an iterative approach, implementing simpler models and then building towards our final model. In the following subsections we will describe our initial baseline (Baseline Model), our improved baseline (Baseline++ Model), our intermediate model which adds an answer-pointer decoder to Baseline++ (Boundary Model), and our current model which uses a coattention encoder and the answer-pointer decoder (Final Model).

Baseline

The original baseline is built off of two BiLSTM RNNs used to get our question and (context) paragraph representations from the question/paragraph word vectors. The paragraph LSTM is "conditioned" on the question data by initializing the paragraph LSTM's hidden state on the final hidden state of the BiLSTM. The output of the paragraph LSTM is then sent to a decoding layer that is simply two fully connected layers where the first attempts to predict the probabilities of being the start and end indices of the answer. The predicted probabilities for the start are concatenated to the paragraph encoding to allow the end decoder to condition on the start prediction to predict the end index. See Figure 8 for a pictorial representation.

Let $Q = [q_1, q_2, ..., q_{|Q|}]$ and $P = [p_1, p_2, ..., p_{|P|}]$ where $|P|$ and $|Q|$ are the max paragraph and question lengths respectively. Then letting i_k^Q be the k-th question word and i_k^P be the k-th context word, q_k and p_k are as follows (h is the hidden state size):

$$\overrightarrow{q_k} = \overrightarrow{\text{LSTM}}(i_k^Q, \overrightarrow{q_{k-1}}) \tag{1}$$

$$\overleftarrow{q_k} = \overleftarrow{\text{LSTM}}(i_k^Q, \overleftarrow{q_{k+1}}) \tag{2}$$

$$q_k = [\overrightarrow{q_k}; \overleftarrow{q_k}] \in \mathbb{R}^{2h} \tag{3}$$

$$
\begin{aligned}
\overrightarrow{p_0} &= \overrightarrow{q_{|Q|}} && (4)\\
\overleftarrow{p_0} &= \overleftarrow{q_{|Q|}} && (5)\\
\overrightarrow{p_k} &= \overrightarrow{\text{LSTM}}(i_k^P, \overrightarrow{p_{k-1}}) && (6)\\
\overleftarrow{p_k} &= \overleftarrow{\text{LSTM}}(i_k^P, \overleftarrow{p_{k+1}}) && (7)\\
p_k &= [\overrightarrow{p_k}; \overleftarrow{p_k}] \in \mathbb{R}^{2h} && (8)
\end{aligned}
$$

Then the probability that a given paragraph position is the start of an answer, h_s, is calculated through a fully-connected layer

$$h_s = PW_1 + b_1 \quad (9)$$

Next h_s is concatenated to the encoded paragraph representation and fed into a final fully connected layer to calculate the answer end position probabilities.

$$h_e = [h_s; P]W_2 + b_2 \quad (10)$$

The simple baseline model is shown in Figure 8.

Baseline++

To enhance the performance of the baseline we restructured the model based on the sequence attention mix model (also as suggested by the course staff as a better baseline). In this model, the paragraph and question are passed through two independent BiLSTMs which create question and paragraph representations P and Q respectively. For this model the input word embeddings were taken from the 100-dimensional GloVe 6B pretrained embeddings. The model equations and pictorial representation of the architecture can be found below.

Let P, Q, and the inputs be defined the same as the simple baseline except that the paragraph BiLSTM is not initialized with the final state of the question BiLSTM such that each row of P and Q are of length h. The question and paragraph representations are then mixed:

$$
\begin{aligned}
A &= \text{softmax}(PQ^T) \quad \in \mathbb{R}^{|P|\times|Q|} && (11)\\
C_P &= AQ \quad \in \mathbb{R}^{|P|\times 2h} && (12)\\
H_r &= [C_P; P]W + b \quad \in \mathbb{R}^{|P|\times 2h} && (13)
\end{aligned}
$$

After encoding the paragraph representation, the probability of each paragraph position being categorized as the start of the answer span is set by

$$h_s = PW_1 \in \mathbb{R}^{|P|} \quad (14)$$

The contextualized paragraph representation is then passed through a final BiLSTM

$$
\begin{aligned}
\overrightarrow{p'_k} &= \overrightarrow{\text{LSTM}}(p_k, \overrightarrow{p'_{k-1}}) && (15)\\
\overleftarrow{p'_k} &= \overleftarrow{\text{LSTM}}(p_k, \overleftarrow{p'_{k+1}}) && (16)\\
p'_k &= [\overrightarrow{p'_k}; \overleftarrow{p'_k}] \quad \in \mathbb{R}^{2h} && (17)
\end{aligned}
$$

Then the end token probabilities are calculated by

$$h_e = P'W_2 \quad \in \mathbb{R}^{|P|} \quad (18)$$

The Baseline++ architecture is shown in Figure 9

Boundary (A-P) Model

The next addition to the model was replacing the simple decoder with an answer-pointer (A-P) layer as described in Wang et al [4]. We implemented the boundary decoder which calculates the probabilities for the start and end answer span positions. This is done with a two cell LSTM with attention mechanisms for the input representation. The result is a decoder that can more confidently select the span of the answer.

If H_r is the input representation then for the k-th cell of the LSTM (2 total) we make the following calculations

$$\begin{aligned} F_k &= \tanh(H_r V + (W^a h^a_{k-1} + b_a)) \quad \in \mathbb{R}^{|P| \times h} && (19) \\ \beta_k &= \text{softmax}(F_k v + c) \quad \in \mathbb{R}^{|P|} && (20) \\ h^a_k &= \overrightarrow{\text{LSTM}}(\beta_k^T H_r, h^a_{k-1}) \quad \in \mathbb{R}^h && (21) \end{aligned}$$

The Boundary model is illustrated in Figure 10

Final (A-P + Coattention) Model

The final addition we made to our model was adding in a dynamic coattention encoder as described in Xiong et al [2]. From our pre-processed context paragraph and question encodings (i.e. the biLSTM hidden vectors), P and Q we first append randomly initialized, trainable "sentinel" vectors so that the model can attend to no particular word in the input. After this, we computed the affinity matrix

$$L = P^T Q$$

We then normalized row-wise and column-wise to get the attention weights, $A^Q = softmax(L)$ and $A^P = softmax(L^T)$, for the question and paragraph respectively. Then we computed the attention contexts for the question and paragraph:

$$C^Q = DA^Q$$

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

where C^D is our co-dependent representation of both question and paragraph. Finally, we ran a biLSTM for a 'fusion of temporal information' [2] to get our final coattention encoding $U = [u_1, ..., u_{|P|}]$ for

$$u_t = BiLSTM(u_{t-1}, u_{t+1}, [d_t; c^D_t])$$

The final model architecture is shown in Figure 1.

4 Experiments

Data

To train our model we used the recently released Stanford Question Answering Dataset (SQuAD) [1] which contains more than 100,000 *context, question, answer* triplets (see Example 1) consisting of questions from Wikipedia articles and their crowd- sourced answers found in the related context reading.

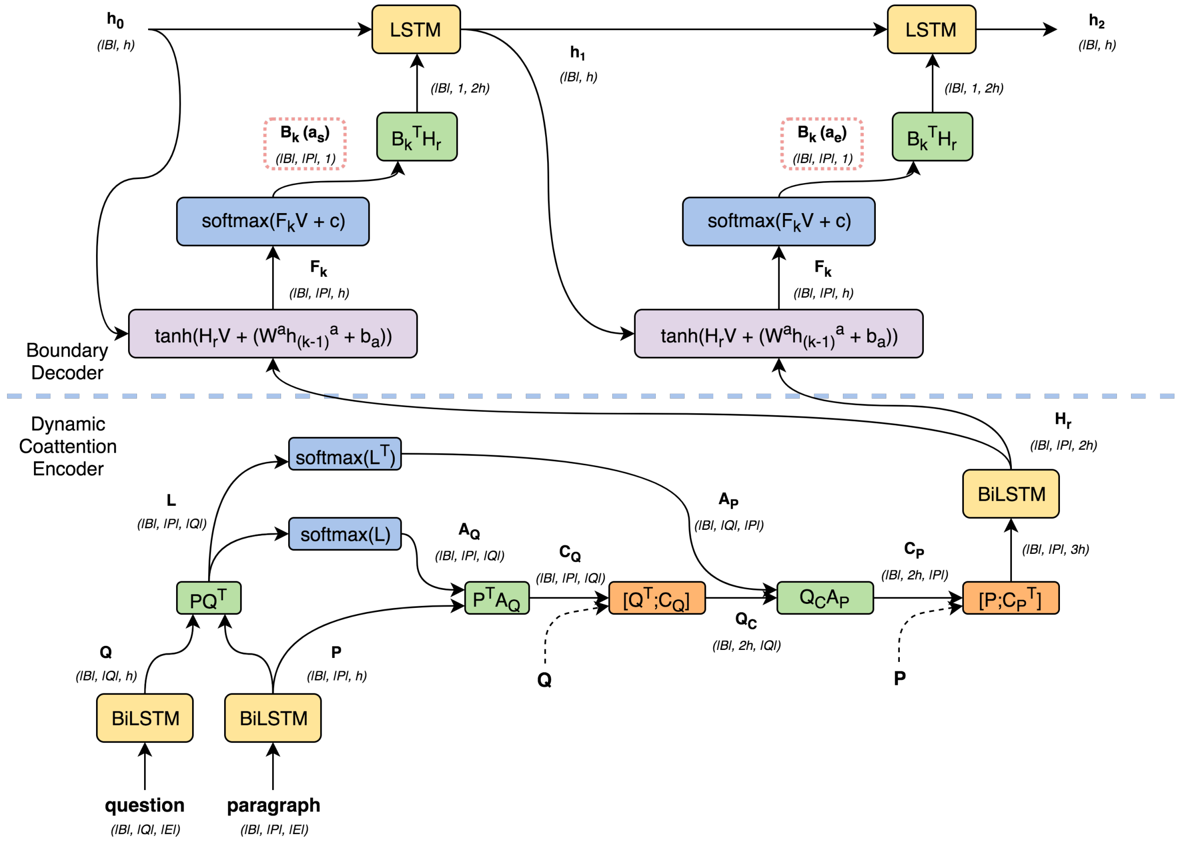

Figure 1: Final (Coattention + A-P) Model Architecture.

Question: What are the principal cash crops in Kenya?
Context paragraph: Agriculture is the second largest contributor to Kenya's gross domestic product (GDP), after the service sector. in 2005 agriculture, including forestry and fishing, accounted for 24% of employment and 50% of revenue from exports. The principal cash crops are **tea, horticultural produce, and coffee**. Horticultural produce and tea are the main growth sectors and the two most valuable of all of Kenya's exports. The production of major food staples such as corn is subject to sharp weather-related fluctuations. Production downturns periodically necessitate food aid – for example, in 2004 aid for 1.8 million people because of one of Kenya's intermittent droughts.
Answer: 'tea, horticultural produce, and coffee'

Example 1: SQuAD Triplet.

Though SQuAD is assembled from only 536 Wikipedia articles generally broken up into paragraph contexts of $\sim$5 sentences each, SQuAD is unique in that multiple crowdworkers provide different answers for each question. This is especially useful for questions that have multiple correct answers or where answers can be partially correct. Additionally, Rajpurkar et al. have shown that despite its size, SQuAD necessitates multiple forms of reasoning for its diverse answers. As such, many of the cutting-edge models for reading comprehension in the past few years have, like us, chosen to train their models on SQuAD.

Experiment settings

In our work, we split the available data as per Table 1.

Description	Count
Training Set	$\sim$81K
Validation Set	$\sim$5K
Development Set	$\sim$10K
Training Set	(Hidden)

Table 1: SQuAD Splits.

The data is read into the model by tokenizing the context paragraphs and questions into key IDs referencing a fixed sized vocabulary using GloVe word embeddings from the Wikipedia 6B word dataset or the CommonCrawl 840B word dataset. The vocabulary is limited to the words present in the seen dataset and unseen words are mapped to a random vector. The preprocessing code also calculates the true length of each example to pass to the dynamic RNN elements in the model. It then will truncate or zero pad the sequences to a fixed max paragraph and question length (e.g. 250 words for the paragraph and 20 words for the question, as we used in the final model) and mask the loss for the padded additions. The final fixed lengths were chosen after analyzing the training dataset and determining where a certain percentile of the data remained un-truncated.

For our model hyperparameters, we tried many various values for a spectrum of fields though certainly we found that some impacted our results more than others. In the end, we chose not to regularize our loss nor clip our gradients. We set the dimensionality l of the hidden layers to be 100 and computed each update using a minibatch length of 50 instances without much variation. We tested both the ADAM [7] and AdaDelta optimizers, and found that though the ADAM optimizer is used by many of the cutting-edge models, the AdaDelta performed and generalized better. In previous models we had witnessed severe over-fitting (see "valid_loss" plot in Fig 2) and as such we ran some of our models with dropout rates of up to 30% (i.e. a keep probability of 0.7). However, we then reduced our dropout rate to 15% for our final runs to speed training after verifying that our final model was not overfitting. Another hyperparameter we adjusted was annealing our learning rate, because we found that our initial models saw somewhat volatile losses between epochs (see "loss" plot in Fig 2). Our final annealing schedule began with a learning rate $lr = 0.005$ and decayed by a factor of 10 per epoch (decay rate $dr = 0.1$). In other words, $lr' = lr \cdot dr^{1/n_{\text{steps}}}$, thus n_{steps} is $\sim$800. Finally, we looked into using different pre-trained word embeddings. Although many of our initial runs were done on GloVe 100-dimensional word embeddings pre-trained on the Wikipedia corpus of $\sim$6 billion words, we moved to the 300-dimensional word embeddings trained on the Common Crawl vocabulary of $\sim$840 billion words in order to ensure we do not lower performance due to unseen vocabulary.

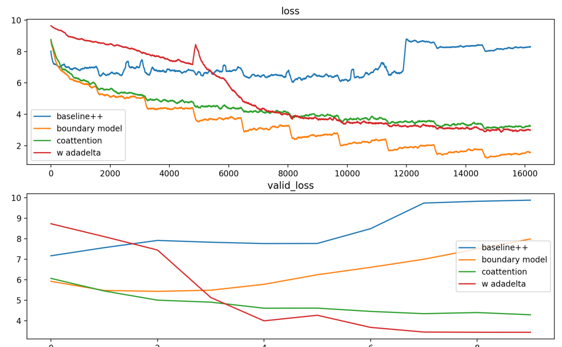

Figure 2: Training loss and validation loss for intermediate and current models. We note that the red line uses the same model structure but changed to the Adadelta optimizer. The jump in training loss is due to restarting training with a higher learning rate.

Results

As with the majority of the state-of-the-art models we looked into, our model will be is evaluated using the F1 and EM (exact match) metrics. F1 score corresponds to a word-level metric comparing tokens in the predicted answers with tokens in the ground truth answer and EM calculates the percentage of answers that match up perfectly with the ground truth answers. (We also note that each question usually has three ground truth answers, and we use F1 scores with the best matching answers to compute the overall F1 score).

The results of our model are shown in below in Table 2. We can see that our model achieves performance similar to the Logistic Regression model proposed by Rajpurkar et al. (2016), which is dependent on meticulously designed features.

	F1 / EM Percentages			
Model	Train	Val	Dev	Test
Baseline	- / -	- / -	- / -	- / -
Baseline++	53.9 / 42	25.6 / 18	20.0 / 11	- / -
Boundary (+ A-P)	90.0 / 87	40.3 / 28	28.1 / 18	- / -
Final (Coattention + A-P)	66.5 / 54	59.1 / 44	61.0 / 47	61.5 / 48.3

Table 2: F1 & EM Results.

Analysis

In order to get an intuition as to where, when, and why our model was failing, visualizing our predictions proved incredibly useful. One such prediction example is shown for our intermediate (Baseline++ and Boundary) and final models in Figure 3. On the x-axis we have the paragraph tokens, representing the token index that the span would output, and on the y-axis we have the estimated probability. Though we considered post-processing alternatives, our models simply chose the start and end indexes that maximized this probability, so this plot shows which places in the context paragraph the model considered choosing.

For this example, the question was: *What is the second largest contributor to Kenya's GDP?* and the answer was: *Agriculture*, according to the paragraph in Example 1 SQuAD Triplet. We see that the left-most plot, Baseline++, has relatively noisy estimates, with many different areas of interest, or "clusters," throughout the paragraph. Moreover, this model predicts the wrong answer, *'corn'*. The center plot, which uses the Boundary model, is much more confident, with its highest probability close to 0.8, and only two different clusters near the beginning and near the middle/end where it predicts the answer. Despite this confidence, it also picks the wrong answer, *'downturns'*. The right-most plot depicts the final model, which uses coattention to capture interactions between the question and the context. This plot also has the same two clusters as in the Boundary model, but this time it has predicted correctly and with reasonable confidence that the answer is *'Agriculture'*.

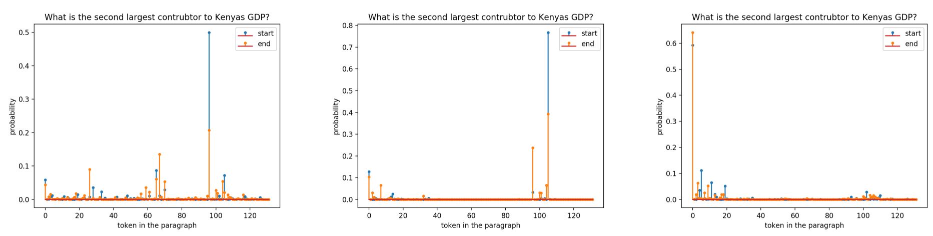

Figure 3: Ex 1. Baseline++, Boundary, and Final model performance

The second example in Figure 4 shows the question: *In what year was Henry Ware elected to chair?* with an answer of *1805* according to the context paragraph below

Throughout the 18th century, Enlightenment ideas of the power of reason and free will became widespread among Congregationalist ministers, putting those ministers and their congregations in tension with more traditionalist, Calvinist parties. When the Hollis Professor of Divinity David Tappan died in 1803 and the president of Harvard Joseph Willard died a year later, in 1804, a struggle broke out over their replacements. Henry Ware was elected to the chair in 1805, and the liberal Samuel Webber was appointed to the presidency of Harvard two years later, which signaled the changing of the tide from the dominance of traditional ideas at Harvard to the dominance of liberal, Arminian ideas (defined by traditionalists as Unitarian ideas).

This example is interesting because all three models were able to identify that the question is looking for a year, but the Baseline++ model predicts *'1804'*, the Boundary model picks *'1803 and the president of Harvard Joseph Willard died a year later, in 1804'* likely because it could not predict definitively between '1803' and '1804'. The final model correctly identified *'1805'* with high confidence (though it did appear to identify the tokens corresponding to 1803 and 1804 as tokens of interest). This boost in performance could be due to a few reasons. The first two intermediate models may have had more trouble performing because 1) it is unlikely that years like '1805' are in these models' vocabulary and thus it is depending solely on 2) accurately capturing interaction between the question and context so that it can fill in the blanks. In our final model, we not only implemented coattention but we also used Common Crawl which has a much larger vocabulary (2.2m words in Common Crawl vs 400k words in GloVe).

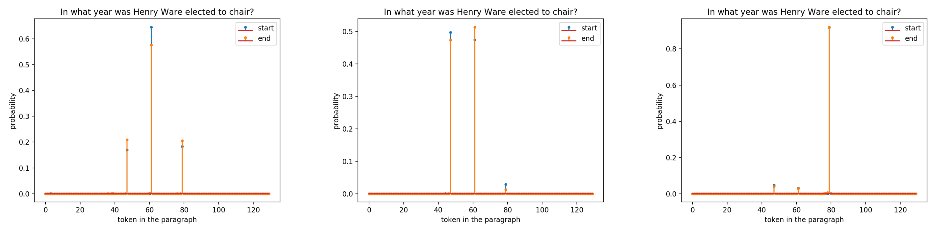

Figure 4: Ex 1. Baseline++, Boundary, and Final model performance

We also delved a bit deeper into our F1 score performance, breaking it down into a histogram representing the number of examples that received F1 scores in a given range to better understand where our overall score came from. In Figure 5, we see that there is a bimodal distribution in our F1 scores, with a lot of perfect or almost-perfect F1 scores but also a lot of F1 scores of zero. This could be an accentuated artifact of coattention, as the Dynamic Coattention Network paper [2] also notes that the distribution is bimodal (although with a much higher proportion of 1.0 F1 scores). It is possible our distribution is more severely bimodal because we implemented the coattention aspect of DCN but did not implement their decoder which was specifically constructed to recover from incorrect local maxima that the model might identify. We also plotted the length of the predicted answer versus the F1 score, which shows that for predicted spans of length $\sim$50 or greater, we usually have very low F1 scores. In other words, when we predict extremely long spans, we are almost certainly wrong (this is corroborated by the fact that a vast majority of ground truth answers are less than 20 tokens long).

5 Conclusion and Future Work

While we did succeed in getting a model to perform on the SQuAD dataset, there are still many improvements that can be made to improve our F1 and EM metrics to be more competitive. From our results – specifically the drop observed between validation and development scores – we could see that the model suffered when it encountered unknown vocabularies. This is because unknown words were simply mapped to the same "unknown" word vector. A potential solution for this would be to add dependency parsing to the model which would make the performance less susceptible to unknown words. Another approach would be to preprocess the data and assign unique identifiers for

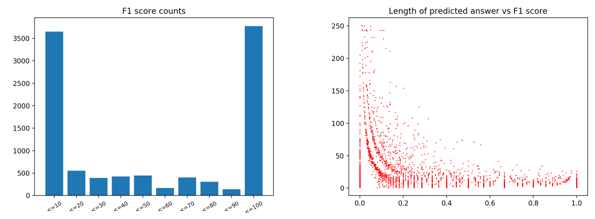

Figure 5: A. Histogram of F1 scores. B. Length of predicted answer vs. F1 score

unseen entities allowing unknown vocabularies to be matched between questions and paragraphs instead of all being mapped to the same embedding.

Another area that could have helped the performance of the model would be more complex decoding to avoid predicting initial incorrect local maxima in the answer span probabilities. In our decoder, we simply identified the indexes of highest probability for the start and end tokens and returned that span. However, when our model does not have a clear consensus of the correct "area of interest" in the paragraph, it can accidentally predict quite long spans which are very likely to be wrong. Perhaps if we instead picked the highest start/end probabilities within an area of interest (where we define an area of interest by looking at probabilistic area under a window of a certain length), we may have more success. Indeed, Wang et al. [4] made this same connection in their paper noting that most SQuAD answers are less than 15 tokens in length. After "clipping" their answers to be within 15 tokens and using span searching, they achieved an F1 score increase of 1.5%. We might be able to achieve similar performance boosts with more complex post-processing. Another idea could be to implement a similar decoder architecture to the DCN paper, as their iterative approach is more robust to situations where the model is torn between multiple potential answers [2].

6 Acknowledgments

Many thanks to Professor Manning and Profesor Socher as well as the entire CS224N teaching staff for all of their support on this project. Additionally, thank you to Microsoft for graciously allowing us access to their Azure GPUs.

7 Contributions

- Ned: wrote data pre-processing methods and helped write baseline++, boundary model, and final coattention model.
- Meredith: wrote data visualization methods and helped write baseline++ and boundary model.
- Miguel: wrote coattention model, wrote match-LSTM encoder (not in final encoder), helped write boundary model.

References

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

[2] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering.arXiv:1611.01604v3, 2017.

[3] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In Proceedings of the Con- ference on Advances in Neural Information Processing Systems, 2015.

[4] Shuohang Wang and Jing Jiang. Machine comprehension using match-LSTM and answer pointer. arXiv preprint arXiv:1608.07905, 2016b.

[5] Rupesh K Srivastava, Klaus Greff, and Juergen Schmidhuber. Training very deep networks. In Advances in Neural Information Processing Systems 28, pp. 23772385, 2015.

[6] Ian J Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron C Courville, and Yoshua Bengio. Max- out networks. ICML (3), 28:13191327, 2013.

[7] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Proceedings of the International Conference on Learning Representations, 2015.

Appendix

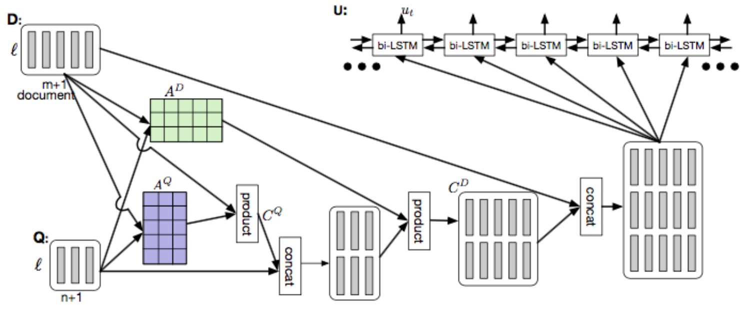

Figure 6: DCN Coattention Architecture.

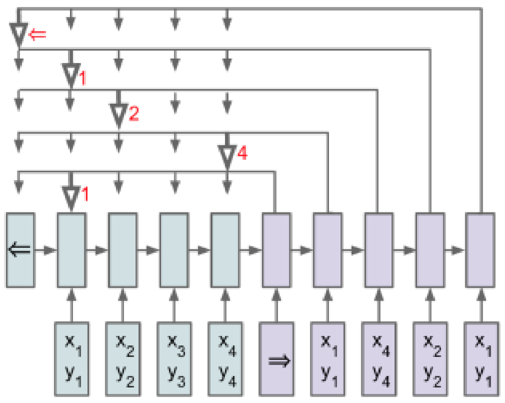

Figure 7: Ptr-Net Framework.

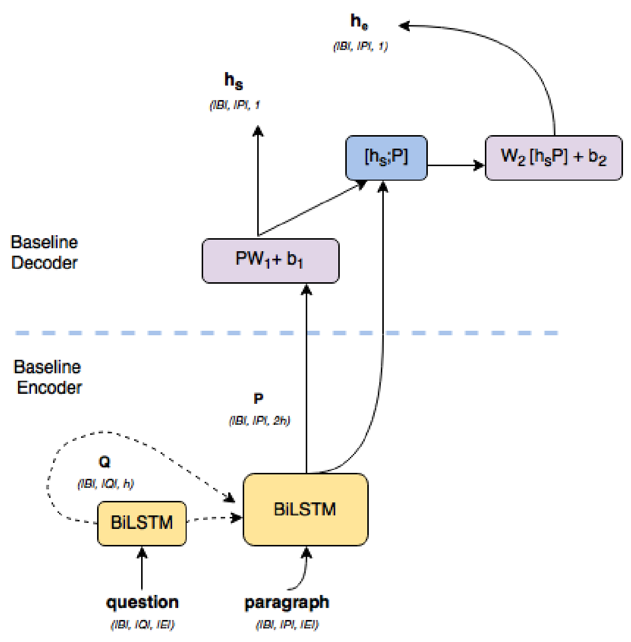

Figure 8: Baseline Model Architecture.

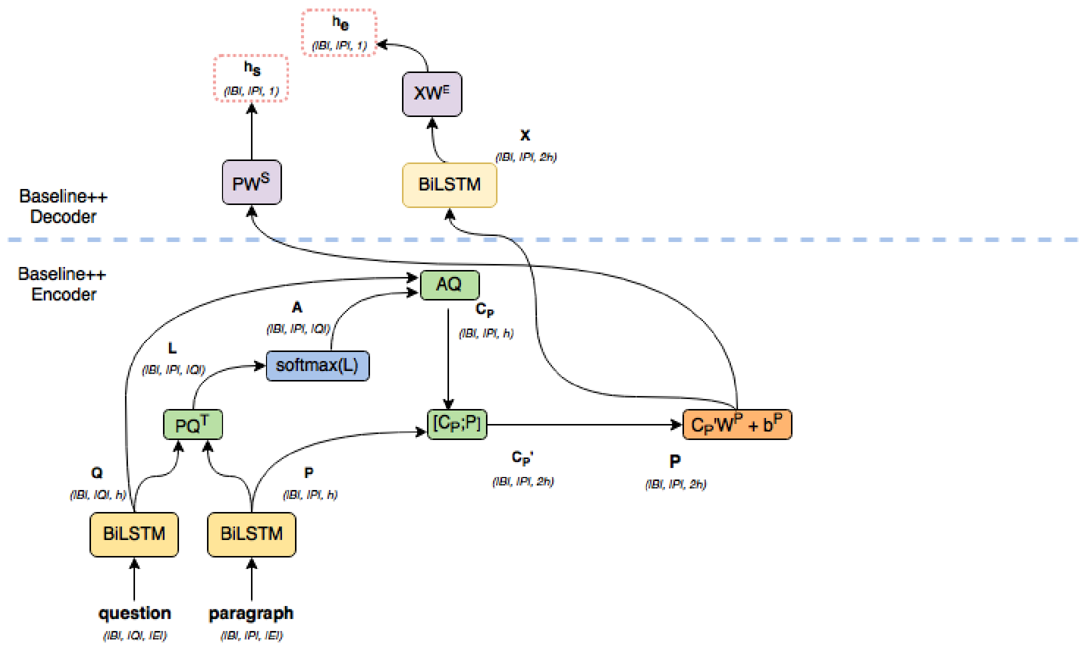

Figure 9: Baseline++ Model Architecture.

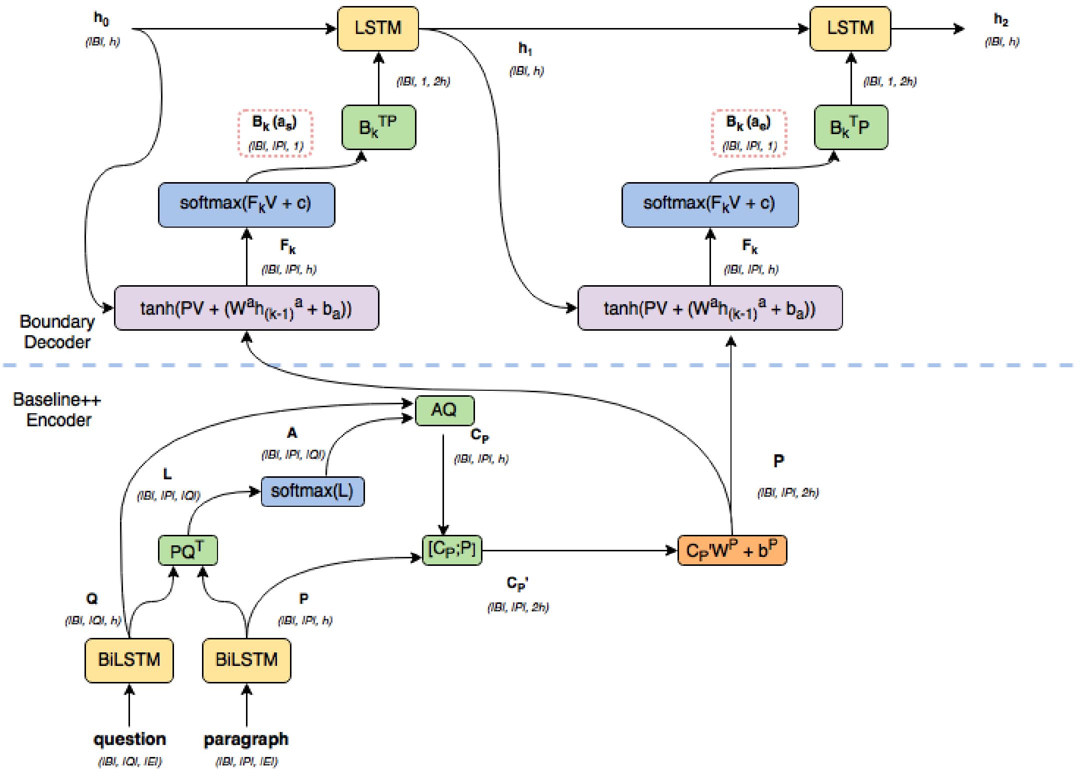

Figure 10: Boundary (A-P) Model Architecture.