Neural Question Answer Systems: Biased Perspectives and Joint Answer Predictions

Sam Redmond
SUNET: sredmond
CodaLab: sredmond

Lucy Wang
SUNET: wanglucy
CodaLab: wanglucy

Nick Troccoli
SUNET: troccoli
CodaLab: ntroccoli

Abstract

In the field of question answering, a fascinating subtask is the identification of contextual evidence that answers a given question. We demonstrate a deep learning model for this reading comprehension task, based on the Multi-Perspective Context Matching (MPCM) architecture that attains the reasonably high performance on the SQuAD dataset of 58.6% for F1 and 44.6% for EM. By combining ideas from MPCM with a variety of creative experimental extensions, we find that of the explored modifications, adding bias terms to perspective weights and predicting joint distributions increase performance, whereas other changes, such as changing relevancy filtering, introducing nonlinearity into the perspective layer, or predicting answer excerpts from a joint distribution, do not meaningfully improve the model.

1 Introduction

The focus of our project, a Question-Answer system, is a complex but intriguing system from an artificial intelligence perspective. Reading comprehension in general is an extremely difficult task, as it requires both understanding of the text itself as well as potentially contextual knowledge of the topic at hand. Moreover, the answer may either lie directly within the passage in question, or be paraphrased or rephrased. One proposed simplification of this problem arrived with the release of the Stanford Question-Answer Dataset, or SQuAD. SQuAD is a dataset of roughly 100K Question-Answer tuples, associated with different context paragraphs. A context paragraph is a paragraph of text taken from Wikipedia, and the questions are human-created questions for which the answer is some span within that context paragraph. For encoding purposes, this answer is represented as a pair $\langle$start index, end index$\rangle$ of its position within the context paragraph. A sample entry from the dataset is shown below (answer in bold):

> *Question:* What description was assigned to minority leader in part ?
>
> *Context:* The roles and responsibilities of the minority leader are not well-defined . To a large extent , the functions of the minority leader are defined by tradition and custom . A minority leader from 1931 to 1939 , Representative Bertrand Snell , R-N.Y. , provided this " job description " : " He is **spokesman for his party and enunciates its policies .** He is required to be alert and vigilant in defense of the minority 's rights . It is his function and duty to criticize constructively the policies and programs of the majority , and to this end employ parliamentary tactics and give close attention to all proposed legislation . "

The length of entries varies significantly in this dataset, which makes prediction more challenging. For instance, as shown in Figure 1, an answer span ranges from a single word to multiple sentences, while context paragraphs can be over 400 words long. We address this, and other challenges associated with this dataset, later on in our paper.

While not an entirely realistic model of question-answering, this nevertheless poses an interesting challenge: can we train a model to accurately determine where in a paragraph the answer to a given question lies?

Our general approach to this problem centered around experimenting with ways to encode a representation of both the context and question in such a way that we could accurately predict the most relevant parts of the context paragraph for that particular question. While starting with a simple baseline employing bidirectional LSTMs for encoding the question and context, we ended with a more complex model employing ideas from Multi-Perspective Context Matching, or MPCM, as defined in [1]. Along the way, we also experimented with a variety of alterations to the MPCM model and input/output representations to achieve reasonable results.

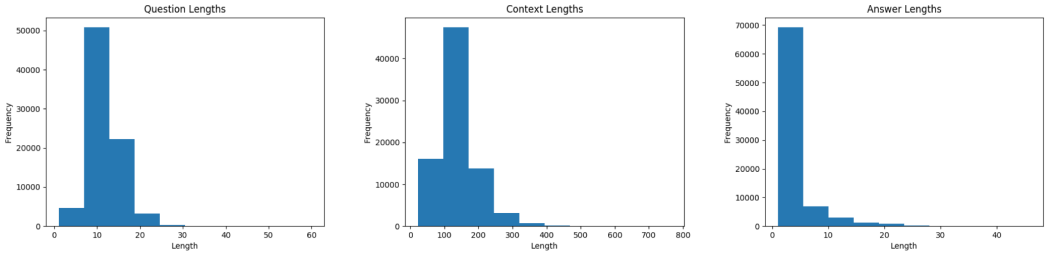

Figure 1: Distribution of question, context and answer lengths

2 Background/Related Work

Historically, reading comprehension has been much harder for machines than humans. On the SQuAD test set, a sliding window model gets an exact match score of 12.5% and a F1 score of 19.7%, a baseline logistic regression model achieves an exact match score 40.4% of and a F1 score of 51.0%, but humans can achieve exact match scores of 77.0% and F1 scores of 86.8% [2].

Though still not to the caliber of humans, many deep learning models have performed particularly well on the SQuAD dataset. For instance, the Match-LSTM model, proposed by Wang and Jiang (2016) achieves an exact match score of 67.9% and an F1 score of 77.0% [3]. This performance is achieved by using word-by-word attention to gather information about how the question relates to the context, and then using this information to choose positions from the input context as output answer predictions. Wang and Jiang consider a boundary model, where a start and end position are predicted from the context to select a contiguous subsequence answer from the context. They additionally consider a sequence model, which instead predicts a (not-necessarily contiguous) sequence of context positions as the answer. It's discovered that the boundary model noticeably outperforms the sequence model: this led us to believe that our approach should also prefer the boundary model over the sequence model.

After reading that boundary models performed well, we became particularly interested in another boundary-predicting model: the Multi-Perspective Context Matching (MPCM) model, proposed by Wang et al. (2016)[1]. This model examines the relationship between the question and the context words through the lens of different perspectives. It performs even better than the Match-LSTM model, with an exact match score of 69.4% and an F1 score of 78.6%. We were drawn to this model not only because it seemed especially successful, but also, because we saw particular potential for experimentation within it. The design of the Multi-Perspective Context Matching model was very clever, but there were several design decisions for which we wondered if we could use our own intuition about reading comprehension to improve. For example, the model introduces a concept of relevancy to filter out context words that are not relevant to a question - we wondered if our own notion of "word relevancy" could be used to improve upon the baseline. Additionally, the model introduces different mechanisms (full-matching, maxpool-matching, and meanpool-matching) to match context words to questions, and we deliberated whether we could introduce our own question-context matching method.

3 Approach

After initially implementing a simplistic baseline that employed a BiLSTM layer to encode question representation, a BiLSTM layer conditioned on that question representation to encode a context representation, and decoding the start and end distributions using two feed-forward neural networks, we migrated to an infrastructure closely modeled after MPCM [1]. However, we made a few tweaks to the MPCM model, partly for simplicity and partly to speed up the training phase.

Additionally, we explored many different creative extensions on top of MPCM; that said, only biased perspectives and joint answer predictions reasonably increased our performance.

3.1 Formal Problem

We first formulate the prediction task from a mathematical perspective. An example $X^{(i)}$ consists of a question $\mathcal{Q}^{(i)} = \left[{q_r}^{(i)}\right]_{r=1}^{m^{(i)}}$ of length $m^{(i)}$, comprised of the sequence of words ${q_r}^{(i)}$ and a context paragraph $\mathcal{C}^{(i)} = \left[{c_s}^{(i)}\right]_{s=1}^{n^{(i)}}$ of length $n^{(i)}$, comprised of the sequence of words ${c_s}^{(i)}$, such that for both the question and the context each word is represented by an index into a vocabulary list. That is, $0 \leq {q_r}^{(i)}, {c_s}^{(i)} < V$, where V is the vocabulary size. From this sample, our goal is to predict two scalar values ${\mathring{A}_s}^{(i)}, {\mathring{A}_e}^{(i)}$, representing the start and end positions of the excerpt in the passage, such that $0 \leq \mathring{A}_s, \mathring{A}_e < n^{(i)}$. That is, we predict that the passage begins at position ${\mathring{A}_s}^{(i)}$ and ends at position ${\mathring{A}_e}^{(i)}$. Note that, as a consequence, we are assuming that the context excerpt that answers the question is represented by a contiguous span of words. We also assume that we are given some collection of word embeddings $\mathcal{E}$ (such as GloVe) which encodes word meaning similarity in embedded vector similarity.

3.2 Adaptations from MPCM

Our model's primary architecture is modeled heavily on the architecture of the MPCM model, although we have made some small changes. As a reminder, the MPCM model is constructed of six layers - a word representation layer, a filter layer, a context representation layer, a multi-perspective context representation layer, an aggregation layer, and a prediction layer.

In the embedding layer, the goal is to associate with each word in the question and context some d-dimensional vector such that word similarity is captured by embedded vector similarity. Our model used GloVe word embeddings with $d = 300$, in order to capture as much semantic relationship between words as possible. We opted to avoid incorporating composed character embeddings, as in the original MPCM model, because we were concerned that the added character embeddings would confuse our model more than help it, as characters don't convey inherent meaning about words.

As with MPCM, our model first applies a relevancy filter, in order to downweight irrelevant parts of the context. Also in line with MPCM, we pass each question and each (filtered) context representation through a BiLSTM, concatenating the forwards and backwards output states at each time step to form a contextualized embedding of each position of each word in the question and paragraph.

In the multi-perspective context matching layer, however, we only implemented full-matching which compares the representation of each position in the context with the forwards or backwards representation of the question, after the question is passed through an LSTM. We did not implement maxpooling-matching or meanpooling-matching, since results in the original MPCM paper suggest that leaving out these matching strategies does not drastically affect performance. In order to compensate for this decrease in learning strength of a given perspective, we increased the number of perspectives up from 50 in the original MPCM paper.

Lastly, when computing the dimensionally weighted matchings $\hat{m}_j$, we utilized inner product distance rather than cosine similarity between the weighted vectors views. Although this strategy is not ideal (in a perfect world we'd measure vector distance the same way the underlying GloVe embeddings do - with cosine similarity), it is unlikely to significantly affect the performance of the model because both inner product and cosine similarity measure the same notion of similarity, up to scaling by the magnitudes of the vectors.

Our aggregation and prediction layers operate without changes from the MPCM paper. In particular, for prediction we use a feed-forward neural network with one hidden layer with sigmoid nodes and softmax activation.

3.3 Performance Optimizations

One of the larger challenges to designing this model was balancing performance with ability to iterate quickly and design more experiments. We sought to increase training speed without losing out on performance.

To that end, the first easy step was to batch our input, learning over minibatches instead of single examples. In each batch, the context paragraphs were padded to a consistent size, as were the questions. For a given batch, we chose as the padded context size the longest context in the batch. However, some contexts are quite long, and in order to avoid padding our data too much (and thus risk slowing down performance), we trimmed all contexts larger than 400 words to 400. Across our training dataset, this change affects only 1% of samples.

Additionally, we clustered examples with similar context lengths into batches, in order to minimize the padding used in given batches. This results in batches with fairly uniform context lengths, which cuts down on padding. In order to ensure that we are not biasing our training by always iterating from smallest context size to largest context size, the order in which we process the batches is shuffled, even though each batch contains examples with similar context length. Furthermore, we padded questions in a given batch only to the longest question size in that batch, rather than the longest question size in the overall dataset. These small changes jointly resulted in approximately a 20x speedup on train time empirically, both on CPUs and GPUs.

3.4 Successful Innovations

Biased perspectives appeared to have a positive impact on performance. In particular, we introduce a trainable bias variable (one for each perspective) that allows more flexibility in modeling an affine relationship in our perspective layer. This bias term is added to both the forward and backwards full matched perspectives. Both the forward question representation and the forward context representation share a bias term, as do the backwards representations.

A big opportunity for optimization came from our model's weakness in predicting start and end indices entirely independently. Assuming we model the location of the start and end indices independently (an admittedly flawed assumption), we can estimate the probability that an answer starts at position a_s and ends at position a_e (given probability distributions p_s and p_e over the start and end of the answer, respectively), as $p_{(s,e)}[a_s, a_e] = p_s[a_s] \cdot p_e[a_e]$. Of course, there will be some dependence between these two values, but if we model them as independent, we can still get a small gain by eliminating all possible combinations where $a_e < a_s$ (that is, 'disabling' the lower-triangular indices) and only finding the best joint answer span out of valid possibilities. In this way, we're attempting to model the joint probability of a pair of answer positions, and then dropping the half of the support that represents an invalid (start, end) pair. More broadly, we would like to learn and incorporate the true dependence between start and end positions (mutually conditioning the choice of start on the choice of end, and vice versa), but that was out of the scope of this project.

Table 1: Innovation results

CHANGES	**F1**	**EM**
None	50.8%	34%
Biased Perspectives	53.03%	43%
Joint Answer Prediction[1]	-	-
Sigmoid Matching	40.7%	27%
Ordered Training	34.3%	21%
$tanh$ Relevancy	26.8%	18%

3.5 Unsuccessful Experiments

In the process of development, we had many ideas for changes to the model. To decide which of these changes we would incorporate into our final model, we compared the results of our model with the change against our model without the change. Since comparative performance between different model instances is evident even from the first epoch, we ran all our experiments [1] on one epoch of the full validation set data. We then compared the performance results with and without each change (see Table 1), choosing to keep only the changes that benefited our model. The changes that were successful and are included in our current model (biased perspectives and joint answer prediction) are described above (section 3.4). Below is a compilation of the attempted changes that were not as successful as we had hoped:

- Sigmoid on Matching Layer

  We experimented with applying a non-linearity (sigmoid) on the matching vectors after multiplying by the perspective matrix. We believed that this would be a useful change to make, since in general introducing nonlinearity allows the model (in the local sense) to adapt to nonlinear relationship between local input and local output, which often is a useful building block in building larger models.

- Ordered Training

  We were curious how training order would affect our model. Our batches were individually bucketed by context length, so we tried both processing the examples in "true" sorted order, where all batches with shorter context sizes were processed before batches with longer context sizes, as well as shuffling the batches randomly. Unsurprisingly, the shuffled training examples led to a better performance on the validation set, because we avoided training against a fixed sequence of inputs.

- $tanh$ relevancy

  In the filter layer, each context word embedding is multiplied by some similarity, which denotes how similar the context word is to any word in the question. The goal of scaling the context word embedding by this similarity is to weight context words by how relevant they are to the question. We noticed that when we (as humans) read with the intent of answering a question, we don't process relevancy linearly with respect to similarity. We attempted to represent relevancy as a nonlinear function of similarity, choosing $tanh(\pi * similarity)$ rather than just $similarity$, to indicate that more similar words should be significantly more relevant, and very different words should be significantly less relevant. We chose $tanh$ rather than sigmoid in order to keep the range the same as cosine similarity, which returns a scalar from -1 to 1.

4 Experimental Results

With a learning rate of 0.001, a dropout of 0.85, batch size of 40, and 10 epochs, we obtained an F1 score of 58.6% and EM score of 44.6% on the hidden test set, using 300-dimension GloVe embeddings.

Our primary evaluation metrics are F1 and EM. F1 measures the harmonic mean of precision and recall between our predicted answers and the gold answer, while EM measures the exact match, or the proportion of answers that we predicted perfectly.

To investigate our final model's behavior further, we plotted F1 and EM against (golden) answer length. Unsurprisingly, we found that our model did more poorly as answer length increased, but the decay curves visually appear in line with similar graphs from the literature. As shown in Figure 4, the EM score drops dramatically immediately, while F1 stays steady before beginning to drop around length 7. This shows that our model struggles to predict longer-length answers, which may tend to be more complex (such as answers to "How" and "Why" questions, instead of "What" and "Who" questions, which are generally easier to answer)[1].

[1] We did not run the experiment for joint predictions in this manner: rather, we compared the joint prediction results directly with the independent prediction results on the validation test set.

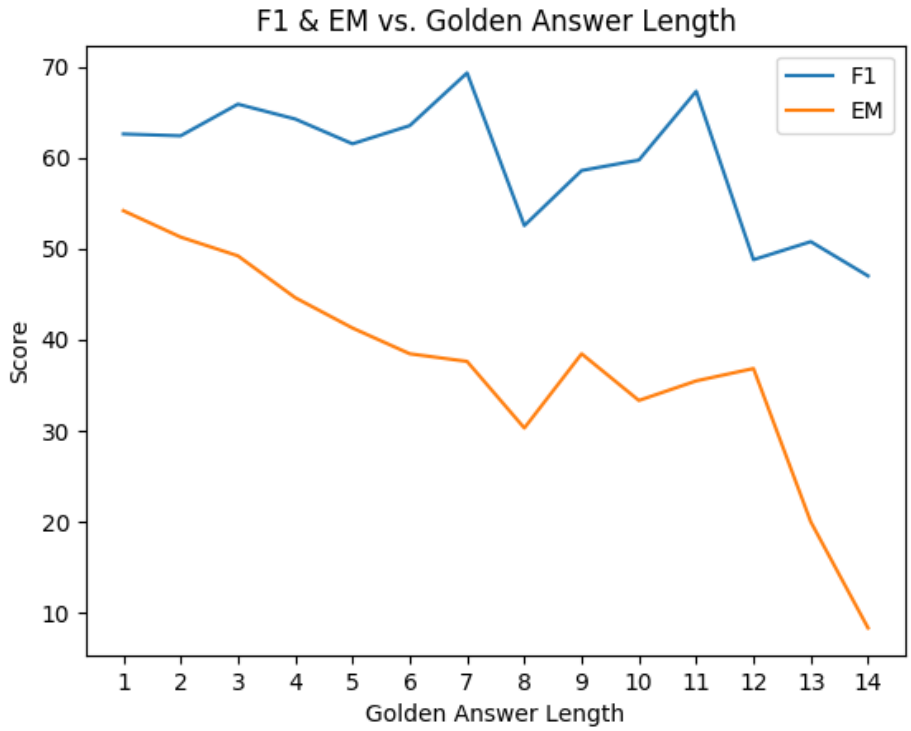

Figure 2: Relationship of answer lengths to evaluation scores

To analyze more specifically where our model excelled and where it fell short, we have chosen a few case study examples. In the examples below, the portion in bold face is our model's prediction and the portion in italics is the gold answer. The first shows an answer that we expect our model to get right; it is a short, 1-word answer to a "What" question, which typically have very direct answers:

> *Question:* What year did the university open a center in Hong Kong?
>
> *Context:* The University of Chicago also maintains facilities apart from its main campus....[truncated for brevity]...The most recent additions are a center in New Delhi, India, which opened in 2014, and a center in Hong Kong which opened in ***2015***.

The second example shows a difficult answer on which we would expect our model to fail. Specifically, it does mostly get the answer right, but in the English language there is a slight nuance between being "credited with packet switching" and "credited with coining the modern name packet switching".

> *Question:* What is Donald Davies credited with?
>
> *Context:* Starting in the late 1950s, American computer scientist Paul Baran developed the concept Distributed Adaptive Message Block Switching ...[truncated for brevity]... Davies is credited with *coining the modern name* ***packet switching*** and inspiring numerous packet switching networks in Europe in the decade following, including the incorporation of the concept in the early ARPANET in the United States.

The nuance between coining a term and being wholly responsible for inventing the process itself is something lost on our model, and would be very hard for our model to learn. One potential way to

improve our model to handle these cases is to, once it has predicted an answer span, to explore within a certain radius around that span for additional clauses that may be part of that predicted answer. In this case, this process would hopefully pick up the "coining the modern name" as a phrase that goes with "packet switching".

The third example shows a question with a short answer that we would usually expect our model to answer correctly. However, it predicts a much-too-long answer to a seemingly simple question. After further analysis, it's clear that "how" questions, which can be more ambiguous, are more difficult for our model to answer, since there are potentially a variety of answers that fit a "how" question, as opposed to a more closed question like "when".

> *Question:* How long was the longest Doctor Who Christmas Special?
>
> *Context:* 826 Doctor Who installments have been televised since 1963, ranging between 25-minute episodes (the most common format), 45-minute episodes (for Resurrection of the Daleks in the 1984 series, a single season in 1985, and the revival), two feature-length productions (1983's The Five Doctors and the 1996 television film), **eight Christmas specials (most of 60 minutes' duration, one of *72 minutes*), and four additional specials ranging from 60 to 75 minutes in 2009, 2010 and 2013. Four mini-episodes, running about eight minutes each**, were also produced for the 1993, 2005 and 2007 Children in Need charity appeals, while another mini-episode was produced in 2008 for a Doctor Who-themed edition of The Proms....[truncated for brevity]

Finally, this last example shows a question that our model unexpectedly got correct. A seemingly-extensive answer 7 words long, predicted perfectly! Upon further analysis, it seems that the "What" question phrasing may have made it slightly easier to answer vs. a "How" question.

> *Question:* What has been the main reason for the shift to the view that income inequality harms growth?
>
> *Context:* Economist Joseph Stiglitz presented evidence in 2009 that both global inequality and inequality within countries prevent growth by limiting aggregate demand...[truncated for brevity]... The main reason for this shift is the ***increasing importance of human capital in development***. When physical capital mattered most, savings and investments were key...[truncated for brevity]

Overall, while these are promising results for our model, these clearly show that we have a long way to go to further improve our QA system. On top of that, there are many intricate nuances (some of which may be language-specific) such as "coining a term' that may be hard for our model to capture.

5 Conclusion

This project has given us an appreciation for the inherent complexity of the reading comprehension task, as well as more experience with deep learning and natural language processing research in general.

In the future, we would like to explore distinctive models that do not suffer from the same pitfalls as our model: for example, the Dynamic Coattention Network model doesn't suffer from diminishing scores with longer answers like our model does [4]. Incorporating the predictions of different models (not just replicated, but with entirely different architectures) in an ensemble would likely improve performance.

Even so, there are many tunable knobs and dials we'd like to explore within the realm of our current model and experiments: for example, how many perspectives leads to the best performance? How does batch size affect training behavior? How can we alter our matching mechanism to incorporate other notions about reading comprehension? Even though $tanh$ scaling failed to succeed, can we find a different nonlinear activation function that would perform better? Can we learn the conditional dependence of the start and end locations? The list goes on. As always, we can also continue to

experiment with hyperparameters such as the hidden size of our LSTMs or the GloVe embedding size.

Overall, we've learned much about the many different ways in which the reading comprehension problem has been tackled, different methods to represent information, different ways to improve performance in deep learning tasks, how to debug deep learning models, and what kinds of representations and techniques are better than others. A more general lesson we learned was that making meaningful improvements to models is not easy - for for every step forwards, we had countless setbacks. Perhaps this is a lesson that can be applied in a broader context: for any complex task, there will be many failures that accompany success, but that is no disincentive to try.

6 Contributions

All three team members contributed heavily to this project. Nick began writing the scaffolding code, while Lucy started to implement the baseline. All $\binom{3}{2}$ combinations of pairs (Nick and Lucy, Nick and Sam, Lucy and Sam) took turns pair-programming progress on the baseline. Sam and Lucy built modifications to the initial baseline, incorporating ideas from the Multi-Perspective Context Matching paper[1]. Lucy debugged the improvements to the model, and profiled and vectorized the model code to run at a reasonable pace. Sam re-factored much of the starter code, modularizing it for readability and maintainability. He also made changes to the batching code and pre-existing code to speed up our run time. Nick handled GPU set up and CodaLab integration, as well as starter code changes. He also wrote the code to generate the visualizations in this paper (the histograms as well as the answer length/score graph).

7 Acknowledgements

We'd like to thank Chris Manning and Richard Socher for providing us with so much insight and advice during this project and throughout the whole quarter, as well as all of the TAs for their innumerable contributions to the discussion of model designs, implementation challenges, and debugging via Piazza.

8 References

[1] Zhiguo Wang, Haitao Mi, Wael Hamza and Radu Florian. 2016. Multi-Perspective Context Matching for Machine Comprehension. *arXiv preprint arXiv:1612.04211v1*

[2] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. 2016. *CoRR, abs/1606.05250*

[3] Shuohang Wang, Jing Jiang. Machine Comprehension Using Match-LSTM and Answer Pointer. 2016. *CoRR, arXiv:1608.07905v2*

[4] Caiming Xiong, Victor Zhong, Richard Socher. Dynamic Coattention Networks for Question Answering. 2017. *arXiv:1611.01604v3*