Question Answering with Deep Learning

Hao Wu
Department of Computer Science
Stanford University
Stanford, CA 94305
wuhao20@stanford.edu

Jacqueline Yau
Department of Computer Science
Stanford University
Stanford, CA 94305
jyau@stanford.edu

Abstract

Given the Stanford Question Answering Dataset, we introduce a deep learning model that will predict a span (start position, end position) within a context paragraph that answers a question. Our model builds upon previously established models and techniques, such as Coattention [3], BiDAF [2], the R-Net model [1], and Smart Span selection [5]. We also carefully analyzed the errors that our model is making at each attention layer, by plotting the attention distribution heat map and looking at various error statistics,. We also explored approaches to mitigate these errors. Though not all of our approaches succeeded, we found interesting insights into how our model works at per layer level, which gave us many ideas for future improvement.

1 Introduction

Humans are well-known for their ability to critically think and reason to reach a conclusion. In everyday life, as people interact with each other, we deduce what actions to take from the information given.

Reading comprehension itself is an act that requires critical thinking to derive the answer from the given context. Given a paragraph and a question about the paragraph, our goal is to answer the question correctly. For this task, we are using SQuAD (Stanford Question and Answering Dataset).

We strive to create a model that is able to learn from each example it goes through, and improve on its ability to find the right places in the paragraph to focus its attention on to determine the answer. To do so, our approach is to first focus attention on the context to gather the information, then focus attention on the question to figure out where in the paragraph is the answer to the question.

2 Background/Related Work

The Natural Language Computing Group of Microsoft Research Asia introduced the R-NET, an end-to-end neural networks model for reading comprehension [1]. It first matches the question and context with gated attention-based recurrent networks to get the question-aware passage representation. Then, they use a self-matching attention mechanism to improve the representation by matching the passage against itself, encoding information in the passage. Finally they use pointer networks to locate the positions of answers from the context paragraphs. However, they were limited in trying to integrate syntax information into the model.

The results of just a single model are very good, reaching an F1 score of 80.7 and EM 72.3

on SQuAD. The ensemble performed even better.

The Allen Institute for Artificial Intelligence in the University of Washington introduced the Bi-Directional Attention Flow (BiDAF) network [2]. It represents the context at different levels of granularity and uses a bidirectional attention flow mechanism to obtain a query-aware context representation without early summarization. Single model reached F1 77.3 and EM 68.0 on SQuAD.

Salesforce research introduced the Dynamic Coattention Network (DCN), which fuses co-dependent representations of question and context paragraph to focus on the relevant parts for both [3]. A dynamic pointing decoder iterates over potential answer spans. It allows the model to recover from the local initial maxima corresponding to incorrect answers. Single model reached F1 75.9% from SQuAD.

We also used the smart span technique [5] which was mentioned in Reading Wikipedia to Answer Open-Domain Questions by Danqi Chen et al.

3 Approach

3.1 Overview of Approach

Our overall approach is to build and train a model with features suggested by past research papers, evaluate its performance, analyze where it fell short, and improve the model by adding other techniques or tuning hyper-parameters to attempt to make up for where we think the problem was. We will discuss some of the techniques that we have explored, some implementation choices we made, and how we arrived at our final model.

3.2 Baseline Model

The baseline model we used is the one given in the default project. It has an RNN encoder layer that encodes both context and question, an C2Q attention layer that correlates the context and question representations, and an output layer that applies a fully connected layer and two separate softmax layers, one to get the start location and one to get the end location of the answer span.

3.3 Bidirectional Attention Model

The first improvement we made was to implement BiDAF [2] to replace the basic attention. We followed BiDAF's original equations below:

C2Q Attention:	Q2C Attention:
$S_{ij} = w_{sim}[c_i; q_j; c_i \circ q_j]$	$m_i = max_j S_{i,j} \epsilon \forall i \epsilon 1...N$
$\alpha_i = softmax(S_{i,:}) \epsilon R^M \forall i \epsilon 1...N$	$\beta = softmax(m) \epsilon \mathbb{R}^N$
$a_i = \sum_{j=1}^{M} \alpha_j^i q_j \epsilon \mathbb{R}^{2h} \forall i \epsilon 1...N$	$c' = \sum_{i=1}^{N} \beta_i c_i \epsilon \mathbb{R}^{2h}$

Final output for each context location i is b_i:
$b_i = [c_i; a_i; c_i \circ a_i; c_i \circ c'] \forall i \epsilon 1...N$

We saw some improvements in all of our evaluation metrics after implementing BiDAF (F1 & EM for both Training and Dev), although the improvements were not too significant, which led us to explore other techniques such as the Coattention.

3.4 Coattention Model

The next improvement we tried was implementing coattention [3]. We also followed the original equation.

$q'_j = tanh(W q_j + b) \epsilon \mathbb{R}^l \forall i \epsilon 1...M$

*We discovered and used the Einstein Sum, tf.einsum, to accomplish broadcasted matrix multiplication, so the above step were done in batch.

The model also appends sentinel to the context and the transformed question representations, giving the attention mechanism a choice to pay attention to nothing.

C2Q Attention:	Q2C Attention:
$L_{ij} = c_i^T q'_j \epsilon \mathbb{R}$	$\beta^j = softmax(L_{:,j}) \epsilon \mathbb{R}^{N+1}$
$\alpha_i = softmax(L_{i,:}) \epsilon \mathbb{R}^{M+1}$	$b_j = \sum_{i=1}^{N+1} \beta_i^j c_i \epsilon \mathbb{R}^l$
$a_i = \sum_{j=1}^{M+1} \alpha_j^i q'_j \epsilon \mathbb{R}^l$	$s_i = \sum_{j=1}^{M+1} \alpha_j^i b_j \epsilon \mathbb{R}^l \forall i \epsilon 1...N$

Finally, we combine the Q2C and C2Q attention and fed them to a biLSTM: $u_1, ..., u_N = biLSTM([s_1; a_1], ..., [s_N, a_N])$
$u_1, ..., u_N$ will be the output of this attention layer.

Coattention performs very well when we didn't use q', and used the raw question representations passed from the previous RNN encoder layer. Once we used q', the end (plateau) performance actually slightly decreased, however the loss drops much sharper in the beginning, we made an assumption that values of Matrix W is probably too large and the model is overfitting to the first dozens of training batches. Indeed, after we clipped values of W to between 0 and 1.5, the performance got much better.

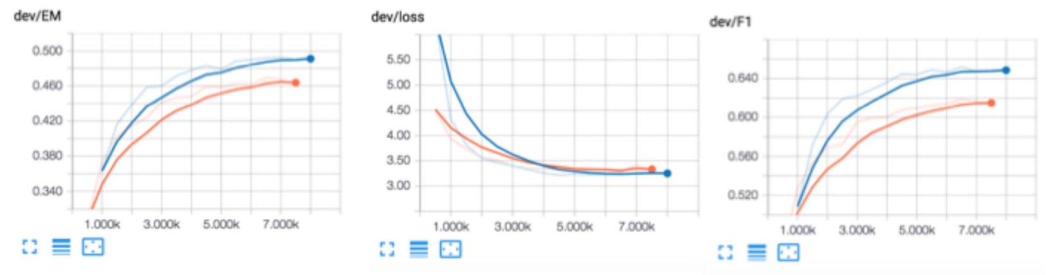

Figure 1: Orange: coattention model with dot product. Blue: coattention model with 1.5 W parameter value clipping for multiplicative attention

3.5 BiDAF and Coattention Model

After seeing how powerful the coattention and BiDAF models were, we wanted to try an ensemble and put both models in parallel by concatenating their attention output.

$b_{final} = [b_i; u_i]$ where b_i is the attention output of biDAF and u_i is the attention output of Coattention

This approach performed very well for dev F1/EM, although it still overfits to the training set like the previous models. To remedy the issue, we tried L2 regularization and even Word Dropout [7], which was mentioned in lecture. These techniques did prevent overfitting to some degree, but the model did not perform as well as before for both training set and dev set. Moreover, since we don't have enough time and resources to tune these regularizations, we decided to not use them.

3.6 Coattention and Self-attention Model

From our previous models, we observed that Coattention alone was performing very well already, and that BiDAF in parallel with Coattention model performed only slightly better. We looked at some of the errors that we were making at this point and came to the conclusion that maybe the model isn't deep enough to fully understand questions and contexts with complex syntactical structure and meaning.

After some research, we came upon R-Net [1], a model that has a Context/Passage Self Attention layer after the Question and Context/Passage matching layer. This immediately made sense to us, as the self matching layer is performing some sort of iterative reasoning in a sense. The first Question and Context/Passage matching layer primarily focused on word-level, isolated correlations between words in the question and words in the context.

The Self Attention layer further connected words in the passage together, achieving a deeper understanding of the context. And indeed, this model performed better.

3.7 Coattention and Self-attention with Smart Span Model

We observed that some of our models' mistakes were when the answer show up multiple times in the passage (in exact words or not), and the model choose one appearance's start with another appearance's end, which can include many words in the middle (length of span is very large).

To prevent this type of mistake and further improve F1 and EM scores, we used a smarter Span Selection mechanism, which is suggested in the DrQA[5] paper, we choose the start, end location pair (i, j) that maximizes the probability $p^{start}(i)p^{end}(j)$ while constraining $i \leq j \leq i+15$. The threshold 15 is chosen because only 2.379% of the training answer spans has length above 15, so we are not excluding answer lengths that appear very often:

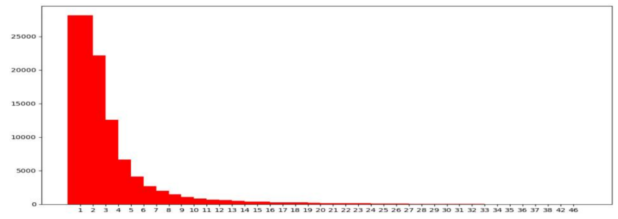

Figure 2: Histogram of train answer length frequencies

Moreover, **we discovered the short comings of the Self Attention mechanism in R-Net**. For a piece a passage where the wrong answer appear multiple times, even when the wrong answers aren't exactly the same, the model will eventually lean towards such wrong answers as it will still assign high attention score to them and their representations will dominate the sequence [See example in Figure 9].
This led us to propose the Mindful Self Attention, which is Self Attention on the context with the question in mind.

3.8 Coattention, BiDAF and Mindful Self-attention plus Smart Span

The idea of adding BiDAF as an additional layer in between Coattention and Self-Attention stems from our observation of human behaviors.

When people perform this kind of question answering tasks, they sometimes re-read the question and tries to get a better sense of it after the first pass of reading the question and the context, especially when the context or question is complicated or recursive. This behavior is "simulated" by our model in the following ways:

1. Coattention layer builds correlation (cross-references) among context positions and question positions, the attention output is "context representations with the question in mind"
2. BiDAF layer allows the model to re-read the question and see which parts of the new context representation, which incorporated the part of the question that it paid attention too, is more relevant to the question.
3. **Mindful Self Attention** does context/passage self matching again, achieve better understanding of the passage with the question "in mind".

$relevance_{i,j} = c_i^T q_j \epsilon$ relevance is of size (batch_size, context_lens, question_lens)
$m_i = max_j relevance_{i,j}$
$\beta = softmax(m)$, take softmax along i
$attention_score_{i,k} = \beta_i c_i^T c_k$

This turns out to be our best and final model.

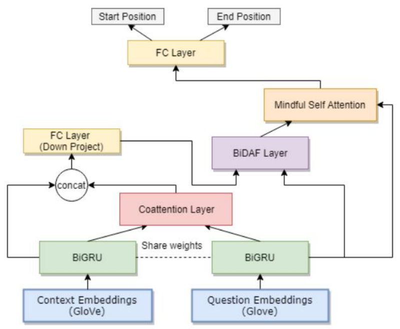

Figure 3: Final Model Diagram

4 Experiments

4.1 Dataset

We evaluated our models on the SQuAD dataset. This is a reading comprehension dataset consisting of questions posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage. It has 100,000+ question-answer pairs on 500+ articles.

4.2 Experiment Setup

We implemented all of our models with TensorFlow. Word embeddings are based on the pre-trained GloVe embeddings of dimension 100. We set our number of epochs to 0 to train indefinitely. We use Adam optimizer with learning rate of 10^{-3}. We set batch size to be 100. Hidden size is set to 200. We reduced the context length to 350 to be able to run our model on 1 GPU, 350 is determined by plotting a histogram of the frequency of all context lengths and realizing that there are only **0.586%** of the training contexts with length above 350. Embedding size is 100. We apply standard dropout with 0.15 regularization.

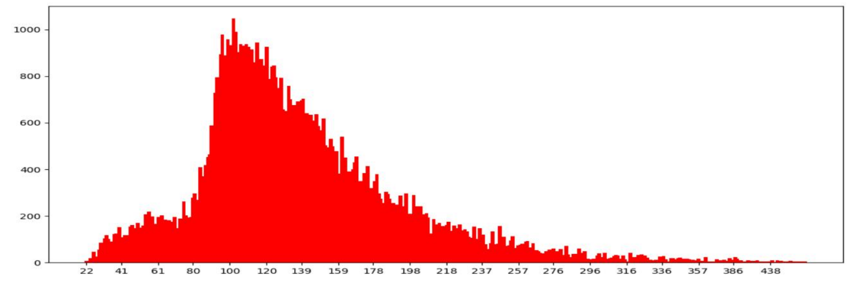

Figure 4: Histogram of context length frequencies of train data. x-axis: context length, y-axis: frequency

4.3 Evaluation Metric

The evaluation metrics used are the F1 score and Exact Match (EM) score. The F1 score is the harmonic mean of precision and recall. Precision is the number of correct positive results divided by the number of all positive results. Recall is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive). The EM score is a binary measure of whether model's predicted answer matches the ground truth exactly.

4.4 Experiment Results

BiDAF Model: It did perform better than our baseline, results for BiDAF in Table 1, but we wanted to try other techniques to see if it would help improve/do better. *In implementation, we actually broke W_{sim} which is originally a vector of size $\mathbb{R}^{6h}$ to three $\mathbb{R}^{2h}$ vectors, and applied *tf.matmul* individually with C, Q and CQ^T, then expand certain dimensions of each result and made a broadcasted addition. This is done to conserve memory consumption of the model.

Coattention Model: We ran experiments with 2 different attention formulas (dot product attention and multiplicative attention). We also explored various W parameter value clipping for multiplicative attention, and found that the optimal value is 1.5. Coattention performed significantly better, for Coattention in Table 1.

BiDAF and Coattention Model: After the previous two models, we wanted to test out how well having both BiDAF and coattention in parallel would perform. It performed slightly better than coattention by itself as seen in Table 1.

Coattention and Self-attention plus Smart Span Model: Since BiDAF and Coattention in parallel did not perform significantly better, we took out BiDAF and instead added self-attention and Smart Span because it was proposed in the R-Net[1]. As the results show in Table 1, this model performed significantly better than our previous models.

Techniques that didn't bring improvements so far: Answer Pointer [6], Iterative reasoning [3], Additive Attention. Based on our error analysis, we also created some original techniques like "BiDAF with beta from alpha" and "Self Attention weighted by Question" which didn't make improvements so far (didn't really do worse either) which we will talk about later.

4.4.1 Final Model: Coattention, BiDAF, and Self-Attention plus Smart Span

Our final and best model added BiDAF back again to the coattention and self-attention plus Smart Span model because it can give the model a second chance to go over the question. This model performed a little better than the previous model as indicated in Table 1.

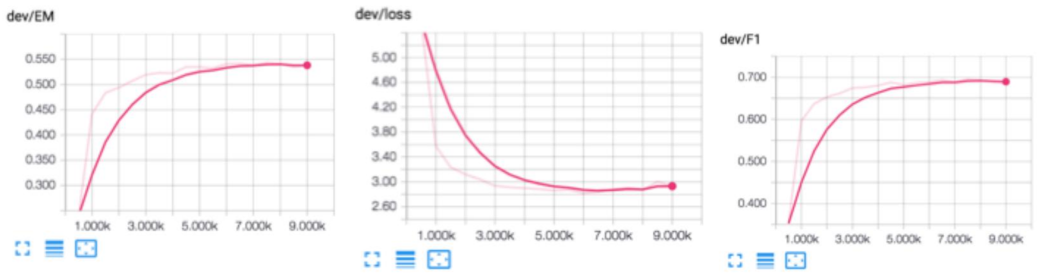

Figure 5: TensorBoard of Coattention, BiDAF, and Self-attention plus Smart Span Model

Table 1: Dev Evaluation Results of some of the models we tried

Model	EM	F1
BiDAF	0.2927	0.4316
Coattention	0.4932	0.6497
BiDAF and Coattention in parallel	0.4770	0.6255
Coattention and Self-attention plus Smart Span	0.6325	0.7406
Coattention, BiDAF, and Self-attention plus Smart Span	0.6417	0.7500

4.5 Error Analysis

There are in general 2 main approaches we took in error analysis: 1. Statistical approach: we collected statistics on what type of errors were made and how often these errors appear. For example, here are the statistics we collected for our final model:
For 100 examples:

start right end wrong: 11 = 0.11%	end right start wrong: 16 = 0.16%
totally wrong: 25 = 0.25%	start within true span: 14 = 0.14%
end within true span: 11 = 0.11%	prediction span encapsulates truth: 2 = 0.02%

This approach informed out attempts to implement the Answer Pointer[6], which is to condition the span end on the start, since there are a lot of wrong predictions had the right start but wrong end or vice versa. We believed that Answer Pointer will help us better predict both the start and the end as the end's loss will also propagate to the start. However, we could not achieve better performance till the date of this report, but it is something we would like to further explore.

2. Example approach: To understand where each of our layers fell short and how they might contributed to the errors our model make overall, we analyzed the attention distribution that each layer produces with concrete examples (we added the 'eval' mode and code to lot attention distribution heatmap). One of the examples is the following:

CONTEXT: (green text is true answer, magenta text is predicted start, red text is predicted end) southern california is home to many major business districts . central business districts (cbd) include downtown los angeles , downtown san diego , downtown san bernardino , downtown bakersfield , south coast metro and downtown riverside . QUESTION: what is the only district in the cbd to not have " downtown " in it 's name ?
TRUE ANSWER: south coast metro
PREDICTED ANSWER: downtown los angeles

** In above example, the question is asking for the district whose name does not contain "downtown", but the coattention layer focused on the ones with "downtown" in their name, due to the word-level high attention score of "downtown" in question and "downtown" in context. This phenomenon is demonstrated with the Coattention's C2Q attention score (α) plot here:*

**The darker the color, the higher the attention score*

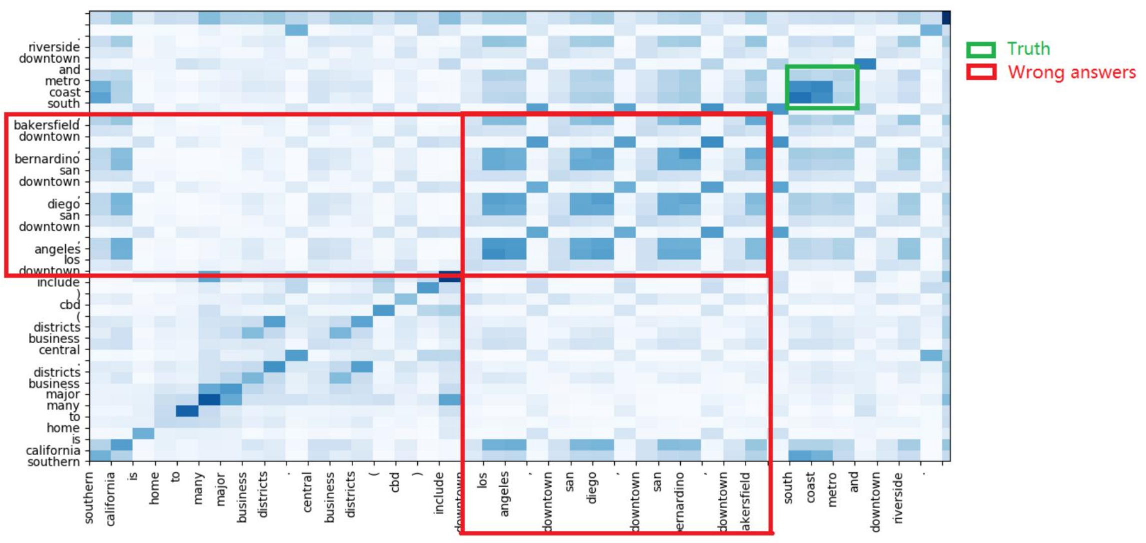

Figure 6: Self Attention Score Heatmap

This type of error inspired us to implement the "Self Attention Weighted by Question", and "Self Attention Maxpooling".
The first aims to bring the question back into self attention so that the appearance of multiple similar wrong answers in the context will not result in a large misleading attention output summation in the end. The equation is:
$Score_ij = c'_i c_j$ where c'_i is calculated with the beta distribution in BiDAF: $c'_i = \beta_i c_i \epsilon \mathbb{R}^{2h}$, however it didn't make the model better. The second also tried to tackle the same phenomenon by not replacing the summation: $a_i = \sum_j score_i, jc_j$ with a max pooling: $a_i = max_j score_i, jc_j$, but it didn't work either at the moment.

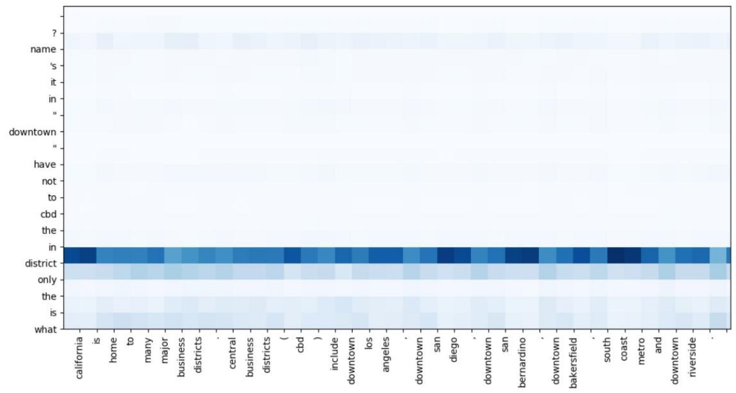

Figure 7: Coattention Q2C matrix
This graph, accompanying the heat map of the raw attention score: L, inspired us to try "BiDAF with beta from alpha" since alpha seems to be a more focused representation of how much each answer candidate relates to the actual question. which is to replace the raw attention score L in $\alpha_i = softmax(L_{i,:}) \epsilon \mathbb{R}^{M+1}$ with alpha, which is the attention distribution taken over the question representations. However, still no improvements were seen.

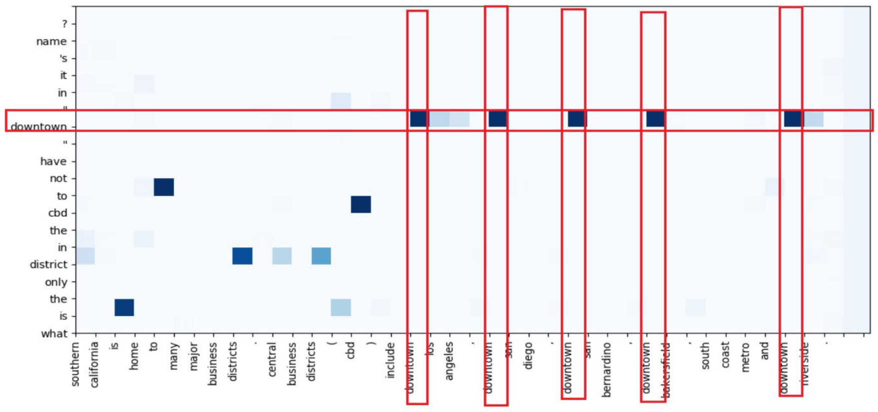

Figure 8: Coattention C2Q matrix
In the end, we suspected that out model is fundamentally not powerful enough to understand complex questions (with negation, dependent clauses etc.), as demonstrated by the Coattention C2Q distribution above. There are also questions that required **world knowledge** to answer, which is impossible for our model.

5 Conclusion

We learned that combining different techniques often require further tuning and modification. Moreover, proper error analysis during the development of a model is very helpful. We have tried to implement Answer Pointers but it didnt work with the rest of the model, which could be something that we want to further explore and integrate.

References

[1] Microsoft Asia Natural Language Computing Group. R-net: Machine Reading Comprehension with Self-Matching Networks. 2017.

[2] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. arXiv preprint arXiv:1611.01603, 2016.

[3] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. arXiv preprint arXiv:1611.01604, 2016.

[4] Yang Yu, Wei Zhang, Kazi Hasan, Mo Yu, Bing Xiang, and Bowen Zhou. End-to-end answer chunk extraction and ranking for reading comprehension. arXiv preprint arXiv:1610.09996, 2016.

[5] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. arXiv preprint arXiv:1704.00051, 2017.

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

[7] Kevin Clark, Christopher D. Manning. Improving Coreference Resolution by Learning Entity-Level Distributed Representations. arXiv preprint arXiv:1606.01323, 2016.