CS 224N Assignment 4: Question Answering on SQuAD

R. Aykan Ozturk
Stanford University
aykanoz@stanford.edu

Huseyin Atahan Inan
Stanford University
*Codalab: hinan
hinan1@stanford.edu

Kevin Garbe
Stanford University
kgarbe@stanford.edu

Abstract

In this project, we are interested in building an end-to-end neural network architecture for the Question Answering task on the well-known Stanford Question Answering Dataset (SQuAD). Our implementation is motivated from a recent high-performance achieving method that combines coattention encoder with a dynamic pointing decoder known as Dynamic Coattention Network. We explored different ensemble and test decoding techniques that we believe might improve the performance of such systems.

1 Introduction

Reading comprehension has traditionally been an important problem in Artificial Intelligence. Building a machine that can read text and answer questions about it would be a major milestone in AI and could potentially be used in many applications, e.g., finding automatic answers to people's query in an online fashion. In this project, we build an end-to-end deep neural network model that could understand a given context and answer questions related to the context. In the following section, we formally define the problem tackled in this project.

1.1 Problem Statement

The objective in this project is to build a neural network architecture for the reading comprehension task using the recently published Stanford Question Answering Dataset (SQuAD) [1].

In this problem we are given a set of questions, passages containing the answers, and the correct answer span within the passage. The goal is to understand the question and identify the answer in the passage:

- We have a set of tuples (Q, P, A)
- $Q = (q_1, \ldots, q_m)$ is the question of size m
- $P = (p_1, \ldots, p_n)$ is the question of size n
- $A = (a_s, a_e)$ is the answer where a_s is the start and a_e is the end index in the paragraph
- The model takes (Q, P) and returns $\hat{A} = (\hat{a}_s, \hat{a}_e)$.

1.2 Related Work

Even though SQuAD is a very new dataset, it has generated a lot of interest in the NLP community, and there is a lot of research going on the reading comprehension task. In fact, the current state-of-the-art model on the SQuAD leaderboard was submitted only 9 days ago. In this assignment, we

*This is the group Codalab username.

focused mainly on two of the top encoder-decoder architecture models on the SQuAD leaderboard. The first one is Dynamic Coattention Networks [2]. This model combines a coattention encoder with a dynamic pointing decoder, and achieves an F1 score of 75.896 and EM score of 66.233 with a single model (80.383 and 71.625 with an ensemble) [3]. The second architecture we implemented is Multi-perspective Context Matching [4] who introduces a multi-perspective approach and compares each contextual embedding of the passage with the question using multi-perspectives. It achieves an F1 score of 68.877 and EM score of 77.771 with a single model (73.765 and 81.257 with an ensemble). Both of these models and our implementations will be described in detail throughout this paper.

2 Approaches

2.1 Dynamic Coattention Model

Dynamic Coattention Networks (DCN) [2] consist of a coattention encoder coupled with a dynamic pointing decoder. In this section, we will briefly describe the encoder and decoder described in [2], before moving on to explain the differences in our model.

The encoder of DCN has an LSTM that is shared for both question and paragraph. This allows the model to share representation power between the question and paragraph. The input the this LSTM are GloVe [5] embeddings. There is an additional tanh layer for the question to allow variation between the question encoding space and the paragraph encoding space.

The encoded question and document representations are then passed into a coattention encoder. This attention model is summarized in Figure 1.

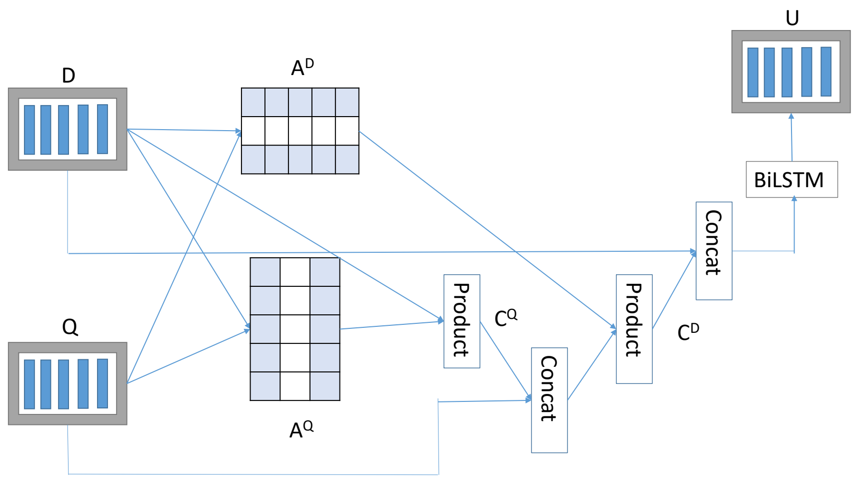


Figure 1: Dynamic Coattention Model

Here $D \in \mathcal{R}^{dxn}$ is the document representation, and $Q \in \mathcal{R}^{dxm}$ is the question representation. We use these and construct the affinity matrix as

$$L = D^T Q. \tag{1}$$

The affinity matrix is normalized row-wise to produce attention weights A^Q across the document for each word in question, and column-wise to produce attention weights A^D across the question for each word in document:

$$\begin{aligned} A^Q &= softmax(L) \\ A^D &= softmax(L^T). \end{aligned} \tag{2}$$

Next step is to use these weights to compute C^Q and C^D, summaries of the question and the document:

$$\begin{aligned} C^Q &= DA^Q \\ C^D &= [Q; C^Q]A^D \end{aligned} \tag{3}$$

C^D is a co-dependent representation of the question and the document, and it can be defined to be the coattention context. The final step is to concatenate this coattention context with D, the temporal encoding of the document and pass it into a BiLSTM:

$$u_t = BiLSTM(u_{t-1}, u_{t+1}, [d_t; C_t^D]) \tag{4}$$

After this step, $U \in \mathcal{R}^{(2d)xm}$ is obtained. This U is then passed to the dynamic pointing decoder. The idea behind dynamic pointing decoder is that there may exist several intuitive start and end word pairs in the paragraph for a given question. These correspond to local optima. Dynamic pointing decoder tries to iteratively update its guess for the start and end words using to avoid getting stuck at these local optima. The decoder is similar to a state machine where the state is maintained by an LSTM-based sequential model. Given $a_{s,i}$ and $a_{e,i}$ (estimates of start and end words at iteration i), this LSTM has the following update equation:

$$h_{i+1} = LSTM(h_i, [u_{a_{s,i}}; u_{a_{e,i}}]) \tag{5}$$

h, $a_{s,i}$ and $a_{e,i}$ are then used to produce vectors α and β. This is achieved through what is called a highway maxout network (HMN), but we will not go into the details of that architecture here. Final step of the iteration is to use α and β to update $a_{s,i}$ and $a_{e,i}$ as follows:

$$\begin{aligned} \alpha_t &= HMN_{start}(u_t, h_i, u_{a_{s,i}}, u_{a_{e,i}}) \\ \beta_t &= HMN_{beta}(u_t, h_i, u_{a_{s,i}}, u_{a_{e,i}}) \end{aligned} \tag{6}$$

In our implementation, the encoder model is very similar. The only difference is that we chose to use a secondary LSTM instead of a tanh layer in an attempt to further improve the variation capability of question encoding. Rest of the encoder model, including the coattention encoder, is fully implemented.

However, we chose to use a much simpler decoder in our model. Even though the dynamic pointing decoder makes a lot of sense in the context of this project, unfortunately it is not easy to implement and it trains a long time to train as well. Given the difficulty of the assignment and the time constraint, we decided to build a simpler decoder. Our decoder consists of a BiLSTM followed by two separate softmax layers: one to produce a_s, one to produce a_e. We will see in the following sections that even though we used a relatively simple decoder, we still achieved very good performance with this model.

Here we make a brief digression.***We would like to emphasize that our best performance is achieved by this single model DCN***. However, the initial plan that we had in this project was the following: We wanted to implement these two top-performance methods that use quite different techniques to tackle this problem and we thought we could obtain a much more powerful system if we take the ensemble of these two implementations and further explore additional tricks to make it even better. It was intuitive to guess that the two models will learn different features and combining them in a smart way would produce a very powerful system. However, until recently, we were unable to obtain a good performance from both methods, though different ensemble methods did give us better results as we will shortly see. Recently, we observed a small error in our implementation of the DCN model and fixing that provided us a very good performance. Unfortunately, we could not locate the possible error in our implementation of second model (M-P CM). For the sake of completeness, we decided to still mention the two models and our ensemble techniques in the following sections. For the best performance results, one may skip the following two subsections.

2.2 Multi Perspective Matching

Multi-Perspective Context Matching (M-P CM) [4] introduces a multi-perspective approach and compares each contextual embedding of the passage with the question using multi-perspectives. Figure 2 depicts the model of this system.

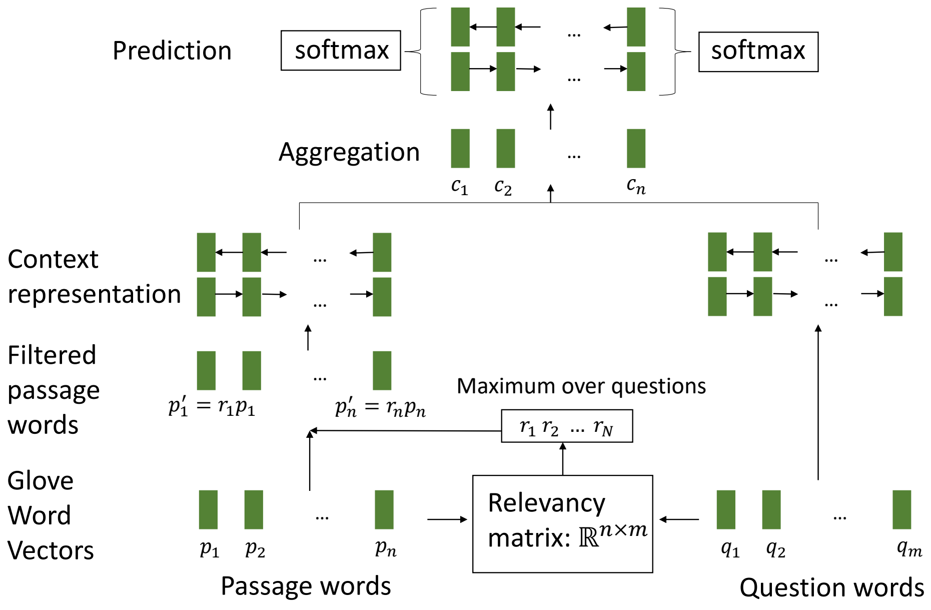


Figure 2: Multi Perspective Matching Model

The first operation is to filter context words that are not relevant to the question words. To do so, the cosine similarity between each question word $\mathbf{q}_i \in Q$ and $\mathbf{p}_j \in P$ is calculated as $r_{i,j} = \frac{\mathbf{q}_i^T \mathbf{p}_j}{||\mathbf{q}_i|| \cdot ||\mathbf{p}_j||}$ and the relevancy degree is obtained by $r_j = \max_{i \in M} r_{i,j}$. Then, each context work is filtered $\mathbf{p'}_j = r_j \mathbf{p}_j$.

In the next layer, same two bidirectional LSTM's are used to blend the contextual information into the words of the context and question, i.e.,

$$\overrightarrow{\mathbf{h}}_i^q = \overrightarrow{LSTM}(\overrightarrow{\mathbf{h}}_{i-1}^q, \mathbf{q}_i) \quad i = 1, \ldots, m$$
$$\overleftarrow{\mathbf{h}}_i^q = \overleftarrow{LSTM}(\overleftarrow{\mathbf{h}}_{i+1}^q, \mathbf{q}_i) \quad i = m, \ldots, 1$$
$$\overrightarrow{\mathbf{h}}_i^p = \overrightarrow{LSTM}(\overrightarrow{\mathbf{h}}_{i-1}^p, \mathbf{p'}_j) \quad i = 1, \ldots, n$$
$$\overleftarrow{\mathbf{h}}_i^p = \overleftarrow{LSTM}(\overleftarrow{\mathbf{h}}_{i+1}^p, \mathbf{p'}_j) \quad i = n, \ldots, 1$$

The core layer compares each contextual embedding of the passage with the question with multi-perspectives.

- Full matching:

$$\overrightarrow{\mathbf{c}}_j^{full} = cosine(\mathbf{W}_1 \circ \overrightarrow{\mathbf{h}}_j^p, \mathbf{W}_1 \circ \overrightarrow{\mathbf{h}}_m^q)$$
$$\overleftarrow{\mathbf{c}}_j^{full} = cosine(\mathbf{W}_2 \circ \overleftarrow{\mathbf{h}}_j^p, \mathbf{W}_2 \circ \overleftarrow{\mathbf{h}}_1^q)$$

- Max-pooling matching:

$$\overrightarrow{\mathbf{c}}_j^{max} = \max_{i \in (1,\ldots,m)} cosine(\mathbf{W}_3 \circ \overrightarrow{\mathbf{h}}_j^p, \mathbf{W}_3 \circ \overrightarrow{\mathbf{h}}_i^q)$$
$$\overleftarrow{\mathbf{c}}_j^{max} = \max_{i \in (1,\ldots,m)} cosine(\mathbf{W}_4 \circ \overleftarrow{\mathbf{h}}_j^p, \mathbf{W}_4 \circ \overleftarrow{\mathbf{h}}_i^q)$$

- Mean-pooling matching:

$$\overrightarrow{\mathbf{c}}_j^{mean} = \frac{1}{m} \sum_{i=1}^{m} cosine(\mathbf{W}_5 \circ \overrightarrow{\mathbf{h}}_j^p, \mathbf{W}_5 \circ \overrightarrow{\mathbf{h}}_i^q)$$

$$\overleftarrow{\mathbf{c}}_j^{mean} = \frac{1}{m}\sum_{i=1}^{m} cosine(\mathbf{W}_6 \circ \overleftarrow{\mathbf{h}}_j^p, \mathbf{W}_6 \circ \overleftarrow{\mathbf{h}}_i^q)$$

The overall matching vector for each word in the context is the concatenation of all the matching vectors, i.e., $\mathbf{c}_j = [\overrightarrow{\mathbf{c}}_j^{full}; \overleftarrow{\mathbf{c}}_j^{full}; \overrightarrow{\mathbf{c}}_j^{max}; \overleftarrow{\mathbf{c}}_j^{max}; \overrightarrow{\mathbf{c}}_j^{mean}; \overleftarrow{\mathbf{c}}_j^{mean}]$. Each matching technique is beneficial for different kind of question/passage sets. For instance, full matching is useful when only some words of the question is important to know, whereas, Max-pooling and Mean-pooling matchings are important when the information is uniform over the question words.

The last part of the system consists of a bidirectional LSTM to aggregate the matching vectors for the purpose of blending each word in the passage to its surrounding words and produce the probability distributions of start and end indexes using two different feed-forward neural networks. Each word is fed into the network to receive a value and in the end the values are normalized with softmax operation to get the probability distributions.

2.3 Ensemble

As we got two distinct models working, we originally decided to try an ensemble answering system, where it would combine answers predicted by the two models to a final prediction.

For a single model, it assigns two probability distributions to the context paragraph, one for where the starting word might be, and one for the ending word. To produce an answer, we pick the starting word with the highest probability assigned to it, then the ending word *following the starting word* with the highest probability.

For our first ensemble, we predict an answer using each model separately, and choose the one that assigned the higher probability to the starting word. Intuitively, we are picking the answer we are more confident about.

For our second ensemble, we took an average of the probabilities of the starting word from each model and similarly for the ending word, then predicted an answer using these combined probabilities. The intuition behind this is that it would take into account the results from both models to produce a result that both models agree with. For example, if a wrong answer has one model very confident about and the second not at all, it might lose in this case to the right answer that both models are pretty confident about together.

3 Experiments

In this section, we will discuss our experiments in detail. We start with the dataset we used for this assignment.

3.1 Dataset

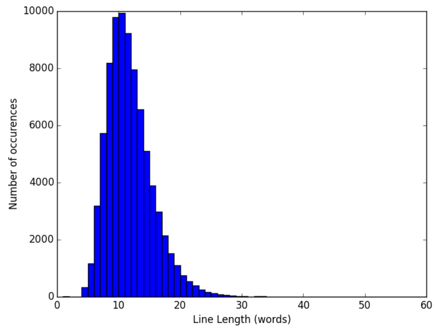
We use Stanford Question Answering Dataset (SQuAD) with 10^5+ questions. It has three sections:

- Training set (80%)
- Validation set (10%)
- Test set (10%)

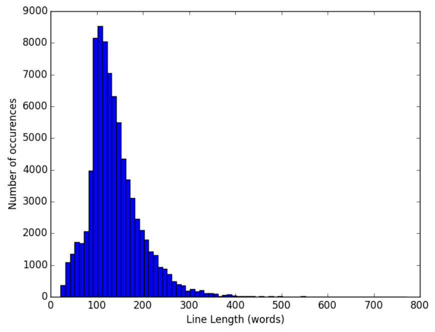
Test set is not available to the general public, so in our research we used the dev set as our test set. We split 5% of the training set as our validation set, and used it for hyperparameter tuning. SQuAD contains of questions that require different kinds of reasoning.

Figure 3 shows the length counts for the training questions and contexts. For readability, the context counts are in groups of 10.

This lead us to choosing the context and question size parameters in Table 1. Context size is the size we forced each context paragraph to be, either padding or truncating and using a mask as needed. For DCN we chose 400, as very few paragraphs were over that length. For M-P CM, we chose 250 as that is also closer to the tail end, and allowed the model to run faster. Similarly, we padded or



(a) Question



(b) Context

Figure 3: Length of Training Questions and Contexts

truncated every question to match the question size parameter. For DCN we chose 60, as that was the maximum length of the training questions. For M-P CM, we chose 30, as very few questions were over that length.

Model	Method 1 (DCN)	Method 2 (M-P CM)
Context Size	400	250
Question Size	60	30
Learning Rate	0.001	0.0005
Embedding Size	100	100
Batch Size	32	30
Dropout	0.2	0.2
Regularization	10^{-6}	None
Gradient clip	Yes, 300	None

Table 1: Parameters

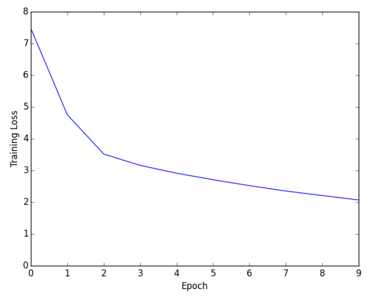
3.2 Results

Once we fixed the bugs in the DCN model, it outperformed the prior ensemble methods. You can see a quantitative evaluation of the performance of our best model with various other models in Table 2. Note that we can observe some slight overfitting in Figure 4 after epoch 4 even though learning rate was decayed manually. Finally, training loss is the average over batches across the training in a single average, while the rest of the values are computed at the end of each epoch.

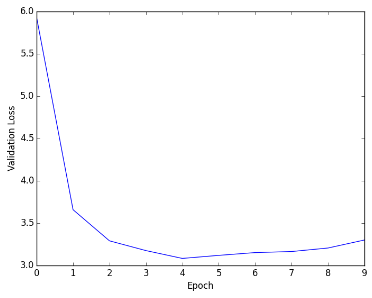
Method	TEST F1	TEST EM	DEV F1	DEV EM
Our Model (expanded vocab)	**63.975**	**50.33**	**70.342**	**57.89**
Our Model (old vocab)	**61.107**	**46.617**	**70.268**	**57.9**
Old Method 1 (DCN)			43.561	29.157
Method 2 (M-P CM)			43.789	29.301
Ensemble (Old Max)			46.786	31.183
Ensemble (Old Average)			47.916	32.479
Ensemble (Average)			70.144	57.455
r-net (best)	84.006	76.922		
Human	91.221	82.304		

Table 2: Final Model Results on [3]

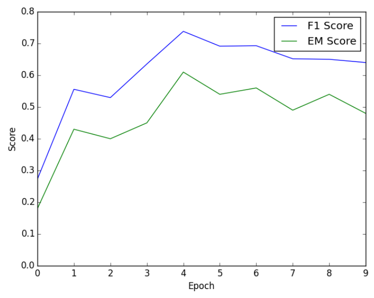
We also want to share some specific (P, Q, A) tuples to qualitatively evaluate the performance of our systems. Due to space limitations, we will only consider the ensemble model and try to talk about both models.



(a) Average loss on training set in each epoch



(b) Average loss on validation set after each epoch



(c) Sampled F1 and EM after each epoch

Figure 4: Results from training, DCN.

Question: What is the Super Bowl program called that gives local companies business opportunities for the Super Bowl ?

Paragraph: For the first time , the Super Bowl 50 Host Committee and the NFL have openly sought disabled veteran and lesbian , gay , bisexual and transgender-owned businesses in Business Connect , the Super Bowl program that provides local companies with contracting opportunities in and around the Super Bowl . The host committee has already raised over $ 40 million through sponsors including Apple , Google , Yahoo ! , Intel , Gap , Chevron , and Dignity Health .

Model 1 answer: 'the Super Bowl 50 Host Committee and the NFL have openly sought disabled veteran and lesbian , gay , bisexual and transgender-owned businesses in Business Connect'

Model 2 answer: 'Business Connect , the Super'

Final chosen answer: 'Business Connect'

Interpretation: This example is very interesting because it shows possible benefits of applying ensemble with the mean technique. Both initial answers are wrong, but we get the correct answer after combining them.

Question: What day of the week was Media Day held on for Super Bowl 50 ?

Paragraph: The game 's media day , which was typically held on the Tuesday afternoon prior to the game , was moved to the Monday evening and re-branded as Super Bowl Opening Night . The event was held on February 1 , 2016 at SAP Center in San Jose . Alongside the traditional media availabilities , the event featured an opening ceremony with player introductions on a replica of the Golden Gate Bridge .

Model 1 answer: 'Monday evening'

Model 2 answer: 'February 1'

Final chosen answer: 'February 1'

Interpretation: This is a possible downfall of the average method. Even though we weigh both methods according to their accuracy, the second model is so confident about its answer that it ends up dominating and leading to a false answer.

Question: Who did the sign language of the National Anthem at Super Bowl 50 ?

Paragraph: Six-time Grammy winner and Academy Award nominee Lady Gaga performed the national anthem , while Academy Award winner Marlee Matlin provided American Sign Language (ASL) translation .

Model 1 answer: 'Marlee Matlin'

Model 2 answer: 'Lady'

Final chosen answer: 'Marlee Matlin'

Interpretation: This final example shows that thanks to the weighting, the method is somewhat robust to false answers from the underperforming model. Also the last two examples suggest that model 1 (DCN) is in general more successful, so ensembling doesn't bring in too much.

4 Conclusion and Future Work

Our experiments demonstrate that our DCN model was successfully implemented. Results are fairly close to the results in 1, and match our theoretical expectations considering that the dynamic pointing decoder was not implemented. One interesting point to note is the initial difference between our test and dev set performances. Before training, **we created our vocabulary with words from training, validation and dev sets**. As we don't have access to the test set, we couldn't add the words there. **As a result, there is a large number of OOV (<unk>) words in test time**, and this causes an increased number of errors. Fixing this problem partly by using all of the 400k words resulted in increased performance naturally, but there is still a performance gap between our dev performance and test performance. We should also note that 1 uses the Common Crawl dataset (2.2m words).

We believe that our results and overall exploration of different methods (given the time constraints and difficulty) exceeds the expected standards in this assignment. Nevertheless, there is still room for improvement, and it is interesting to discuss some future directions for us.

- Implement a more complex decoder for DCN: It would be interesting to try out more complex decoder architectures such as the dynamic pointing decoder to observe how they improve performance. We also built our own decoder model where a_e is chosen conditionally on a_e, but didn't have the time to train it. We should note though that we did explore different techniques to enforce $a_s \leq a_e$ in our decoder. We found that choosing a_s first and then choosing a_e as the word with highest probability improved F1 by slightly more than 1%. Choosing the one of a_s and a_e with higher probability, and then choosing the other one under the constraint imposed by the first also resulted in a similar performance increase.
- Use different GloVe embeddings: Using Common Crawl would greatly reduce the number of OOV words as mentioned, but given that the current GloVe vectors that we use are trained on Wikipedia dataset, we may lose some accuracy in the representation. Also increasing embedding size could result in improved results. We should note that we did try both of these out, but that was before our model started working properly so we chose not to report our results.
- Debug M-PCM implementation and ensemble: Given that our initial plan was to create an ensemble of two different models with the hope that they would learn different things, the final future direction would be to fix our M-PCM implementation and use an ensemble. We believe that this could easily lead to a >75% F1 score.

5 Contributions

- Aykan implemented the DCN based-model and helped with debugging M-PCM.
- Huseyin implemented the M-PCM model and found the critical bug in DCN.
- Kevin worked on the baseline implementation and helped making ensemble work.

References

[1] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ Questions for Machine Comprehension of Text. *ArXiv e-prints*, October 2016.

[2] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. *ArXiv e-prints*, February 2017.

[3] Stanford NLP Group. The Stanford Question Answering Dataset, 2017. [Online; accessed 21-March-2017].

[4] Zhiguo Wang, Haitao Mi, Wael Hamza, and Radu Florian. Multi-perspective context matching for machine comprehension. *ArXiv e-prints*, December 2016.

[5] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.