Machine Reading Comprehension

Akhila Josyula
College of Information and Computer Science
University of Massachusetts Amherst
ajosyula@cs.umass.edu

Sreeparna Mukherjee
College of Information and Computer Science
University of Massachusetts Amherst
sreeparnamukh@cs.umass.edu

Abstract

In this project, we built an end-to-end neural network architectures for the Question Answering task on the well-known Stanford Question Answering Dataset (SQuAD). Researchers have made significant progress using deep learning techniques, especially variations of recurrent neural networks and attention mechanism. We experiment with two such implementations, one has coattention encoder and dyanmic pointing decoder called Dynamic Co-attention networks (DCN) [13], and the other has a hierarchial multi-stage architecture for modeling the representations of context called Bidirectional Attention Flow (BiDAF) [7]. We were able to achieve a F1 and EM of 71.8 and 59 for BiDAF and 55 and 41 for DCN.

1. Introduction

Machine Comprehension(MC) is a special task of Question Answering(QA), where the machine is given a query about a given context and is required to predict the answer. This problem has gained popularity in the recent past and there is increasing reaseach in the fields language and neural networks. To solve this problem, usually an attention is adopted to focus on only a small portion of the context. This attention is used to model the interaction between the query and context. To motivate this line of research, Stanford NLP group released the SQuAD dataset [9], which consists of 100K question-answer pairs, along with a context paragraph for each pair. There is also a public leader board available. The state of the art is already very competitive, as there are many methods that are fast approaching and some have surpassed human level performance.

The rest of the paper is organized as following: This section further goes on to defines the problem, dataset and the relevance of our approach ; Section 2 briefly describes the related work in this field; In section 3 we describe the approaches taken for the baseline model, the DCN and the BiDAF architectures. Section 4 and 5 describe the experiments with data and models, and results. Section 6 outlines the conclusion and future direction of work.

1.1. Problem Statement

Let $c = \{c_1, c_2, \ldots, c_N\}$ be a sequence of context words of size N and $q = \{q_1, q_2, \ldots, q_M\}$ be a sequence of question words of size M. Our model learns the function $f : (q, c) \rightarrow (start, end)$, where $1 \leq start \leq end \leq N$ defines the span of words in the context that corresponds to the answer to the question. Consider the example below:

> **Context**: In meteorology, precipitation is any product of the condensation of atmospheric water vapour that falls under **gravity** . The main forms of precipitation include drizzle, rain, sleet, snow and hail..
> **Question**: What causes precipitation to fall?
> **Answer**: **gravity** [Answer span(17,17)]

2. Related Work

One of the earliest work in machine reading comprehension goes back to Hirschman et. al (1999) [3] where the authors acquired a corpus of 60 development and 60 test stories of 3^{rd} to 6^{th} grade material and used these to create a baseline model using pattern matching (bag-of-words) techniques and several linguistic processing techniques such as stemming, name identification, semantic class identification, and pro-

noun resolution. Following this there has been several other works in this domain and many other datasets have been published.

One of the recent dataset is SQuAD which was first introduced by Rajpurukar et. al (2016) [9]. It contained 100,000+ questions posed by crowd workers on a set of Wikipedia articles. The authors built a strong logistic regression model to extract feature of the candidate answers. Furthermore, the bi-gram frequencies and the root word matching in the question and the parts-of-speech tagging help the model to pick the correct sentences. Following the introduction of SQuAD, it generated lot of interest in the NLP community and several research work has been done using this dataset. A recent paper by Yu et. al (2017) [14] proposed a model called dynamic chunk reader (DCR) which encodes a document and an input question with recurrent neural network (RNN). Following this, the model applies a word-by-word attention mechanism to acquire a representation that is aware of the question. Furthermore, the model generates a chuck of representations and ranks them to find the top-rated chunk as the answer.

Another interesting work has been done by Wang et al [12] where the authors have introduced an end to end neural network model which uses Match-LSTM along with Ptr-Net. This work arises from the concept of Match-LSTM [11] which uses textual entailment such that if two sentences are given, then one is a premise and the other is a hypothesis. In order to predict if the given premise entails the hypothesis, the match-LSTM goes through the hypothesis sequentially. At each position of the hypothesis, attention mechanism is used to obtain a weighted vector representation of the premise. This weighted premise is then combined with a vector representation of the current token of the hypothesis and fed into an LSTM. This match-LSTM essentially aggregates the matching of the attention-weighted premise to each token of the hypothesis and uses the matching result to make a final prediction. Finally, the authors combined this with Pointer Network to use attention as a pointer to select a position from the input sequence and return an output token. This model achieves a score of an EM Score of 54.505 and F1 score of 67.748.

The current state-of-the-art has been achieved by the researchers at Google AI Learning. This model is called BERT (Bidirectional Encoder Representations from Transformers) [2] has outperformed the human performance by 2.0 by achieving an F1 of 93.2 on the SQuADv1.1 challenge. BERTs key technical innovation is applying the bidirectional training of Transformer, to language modelling. This is in contrast to previous efforts which looked at a text sequence either from left to right or combined left-to-right and right-to-left training. Transformer network is a novel neural network architecture relies entirely on an attention mechanism to draw global dependencies between input and output. This model architecture, also developed by researchers at Google as described in Attention is all you need [10], allows for parallelisation which overcomes the squentiality associated with RNN and CNNs.

3. Technical Approach

In this section, we describe the models that we are using for this problem. We start with a simple baseline model, following that we implement two different models and experiment with the performance

3.1. Baseline Model Description

The baseline model has three components: a RNN encoder layer, an attention layer, and an output layer, which applies a fully connected layer and then two separate softmax layers (one to get the start location, and one to get the end location of the answer span).

RNN Encoder Layer: For each SQuAD example (context; question; answer), the context is represented by a sequence of d-dimensional word embeddings $x_1, x_2, x_3, ...x_N \in \mathbb{R}^d$, and the question by a sequence of d-dimensional word embeddings $y_1, y_2, y_3, ...y_N \in \mathbb{R}^d$. These embeddings are pre-trained GloVe embeddings which are shared between answer and context and fed into a 1-layer bi-directional GRU. These provide a sequence of forward and backward hidden states, which we concatenate to get context hidden states c_i and question hidden states q_j.

$$\begin{aligned} c_i &= [\overrightarrow{c_i}; \overleftarrow{c_i}] \in \mathbb{R}^{2h} \quad \forall i \in \{1, 2, 3, ..., N\} \\ q_i &= [\overrightarrow{q_i}; \overleftarrow{q_i}] \in \mathbb{R}^{2h} \quad \forall i \in \{1, 2, 3, ..., N\} \end{aligned} \tag{1}$$

Attention Layer: we apply basic dot-product attention, with the context hidden states c_i attending to the question hidden states q_j . For each context hidden state c_i, the attention distribution $\alpha^i \in R^M$ is computed as follows:

$$e_i = [c_i^T q_1, ..., c_i^T q_M] \in \mathbb{R}^M$$
$$\alpha^i = softmax(e^i) \in \mathbb{R}^M \quad (2)$$

The attention distribution is then used to take a weighted sum of the question hidden states qj , producing the attention output a_i: $\sum_{j=1}^{M} \alpha_j^i q_j \in \mathbb{R}^{2h}$ The attention outputs are then concatenated to the context hidden states to obtain the blended representations b_i: $b_i = [c_i, a_i] \in \mathbb{R}^{4h} \quad \forall i \in \{1, 2, 3, ..., N\}$

Output Layer: Next, each of the blended representations are fed through a fully connected layer followed by a ReLU non-linearity with a weight and a bias vector. We assign a score (or logit) to each context location i by passing b_0 i through a down projecting linear layer and finally apply softmax over this.

Loss: Our loss function is the sum of the cross-entropy loss for the start and end locations. That is, if the start and end locations $i_{start} \in \{1, 2, 3, ..., N\}$ and $i_{end} \in \{1, 2, 3, ..., N\}$ respectively, then the loss is:

$$loss = -logp^{start}(i_{start}) - logp^{end}(i_{end}) \quad (3)$$

During training, this loss is averaged across the batch and minimized with the Adam optimizer.

Prediction: At test time, given a context and a question, we simply take the argmax over p^{start} and p^{end} to obtain the predicted span l^{start} and l^{end}.

3.2. Bidirectional Attention Flow Architecture

From our baseline, we incrementally developed the BiDAF model by appending a contextual layer that encodes the embedded sequences of context and question using bi-directional LSTM. We then get the representation matrix for the context and the question words. The attention layer captures the similarity these matrices and uses these to compute the attented question vectors and context vectors. The attended question vectors are then modeled using bi-directional LSTM and the output span is predicted using Logistic Regression with softmax. Cross-entropy loss is used during training.We achieve a F1 and EM of 71.8 and 59. The figure below shows the architecture and the layers of the model are described below.

The context c and the question q are inputs to the

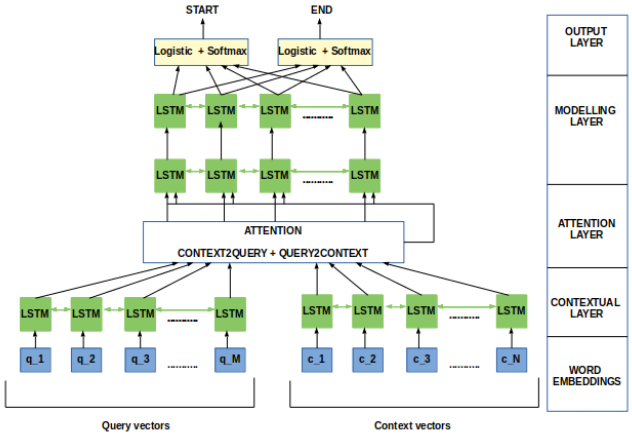


Figure 1: Bidirectional Attention Flow Architecture

model. Based on the size of the distribution of the questions and context, we chose N = 300 and M = 30. Samples larger or smaller than the threshold will be truncated or padded. [7]

- Word Embedding Layer
  The words are converted to vectors using GloVe word embeddings of size 100.

- Contextual Embedding Layer
  The word embeddings from the previous layer is fed to a 1-layer Bidirectional LSTM with hidden size h = 200. The same LSTM models are used for the questions and context vectors. This layer models the temporal interaction between words. This layer outputs foward and backward representation for each word of the input, which are concatenated to end up with a representation of size 2h for each word. The output of this layer is, $L_c \in \mathbb{R}^{N*2h}$ and $L_q \in \mathbb{R}^{M*2h}$

- Attention Layer
  We use bidirectional attention to combine the context and question. the similarity matrix $S \in$

$\mathbb{R}^{N*M}$

$$S_{i,j} = w_{sim}^T[(L_c)_i; (L_q)_j; (L_c)_i * (L_q)_j] \quad (4)$$

where w_{sim} is the similarity word vector and * is the element wise product.

- For each word i we have the context-to-question distribution and output in the form

$$\alpha^i = softmax(S_i) \in \mathbb{R}^M,$$
$$a_i = \sum_{j=1}^{M} \alpha_j^i (R_q)_j \in \mathbb{R}^{2h} \quad (5)$$

- We also have the question-to-context distribution and output in the form

$$m_i = \max_j S_{i,j},$$
$$\beta = Softmax(m) \in \mathbb{R}^N, \quad (6)$$
$$c' = \sum_{i=1}^{N} \beta_i c_i \in \mathbb{R}^2 h$$

- The output of the attention layer

$$(h_a)_i = [(L_c)_i; a_i; (L_c)_i * a_i; (L_c)_i * c'] \quad (7)$$

- Modeling Layer
  This layer uses the attention output to model the probability distribution of the context conditioned on the question. For each position i in the context the bidirectional LSTM outputs $\overleftarrow{b_i}$ corresponding to the forward pass and $\overrightarrow{b_i}$ corresponding to the backward pass. The output of the modeling layer at position i is the contatenation of $\overrightarrow{b_i}$ and $\overleftarrow{b_i}$.

- Output Layer
  We use two different weight vectors w_1 and $w_2 \in \mathbb{R}^2 h$ for predicting the start and end positions.

$$T_i^s = w_1^T[\overrightarrow{b_i}, \overleftarrow{b_i}], \quad T_i^e = w_2^T[\overrightarrow{b_i}, \overleftarrow{b_i}],$$
$$p^s = softmax(T^s), \quad p^e = softmax(T^e) \quad (8)$$

3.3. Dynamic Co-Attention Network (DCN)

In this section, we discuss the second architecture that we have implemented. Dynamic Co-Attention is an an end-to-end neural network for question answering. The model consists of a co-attentive encoder that captures the interactions between the question and the document, as well as a dynamic pointing decoder that alternates between estimating the start and end of the answer span. We achieve a F1 of 55% and EM 41%. The following section is arranged as follows: first we describe the architecture of the network, followed by a discussion about each of its component. In the experiments section, we will discuss about the implementation details and the results.

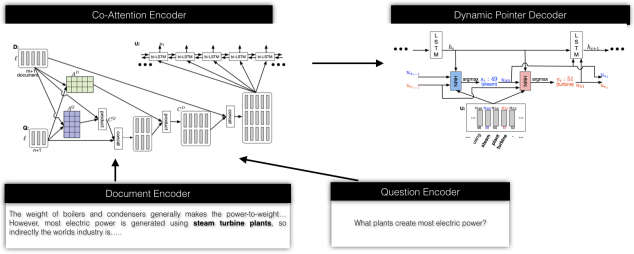


Figure 2: Dynamic Co-attention Network Architecture

3.3.1 Document and Question Encoder

Let $(x_1^Q, x_2^Q, ..., x_n^Q)$ denote the sequence of word vectors corresponding to words in the question and $(x_1^D, x_2^D, ..., x_m^D)$ denote the same for words in the document. Now, with the help of an LSTM [4] we encode the document as $d_t = LSTM_{end}(d_{t-1}, x_t^D)$ and the question encodings are computed in the similar manner. Thus, we have $q_t = LSTM_{end}(q_{t-1}, x_t^Q)$. Both of these calculations are done with the same LSTM hence the parameter sharing is achieved here. Following this, we get document matrix $D = [d_1, d_2, d_3, ..., d_m, d_\phi]$ and the question matrix $Q = [q_1, q_2, q_3, ..., q_n, q_\phi]$. Here d_ϕ and q_ϕ are a sentinel vectors [6] which allows the model not to attend to any particular word in the input. Finally, the question encoding is non-linearly projected using the tanh activation function so as to allow for variation between them. Thus, $Q = tanh(W^{(Q)}Q + b^{(Q)})$

3.3.2 Co-attention Encoder Layer

The next part of the architecture is the co-attention encoder layer. The main goal of the co-attention layer is to fuse both the attention contexts after simultaneously attending to both the document and the question.
The first step of co-attention is compute the affinity matrix. Affity Matrix contains all the scores corresponding to all pairs of document word and question words. It is denoted by $L = D^T Q$ where D is the document matrix and Q is the question matrix. Normalizing the affinity matrix row wise to produce attention weights A^Q and normalizing it column wise, we get A^D. Thus, mathematically, it can be expressed as $A^Q = \text{softmax}(L)$ and $A^D = \text{softmax}(L^T)$. Following this, we compute the attention context of the document under the attention span of each question. Thus, this can be expressed as $C^Q = DA^Q$. Similarly, we compute the attention context of the question under the attention span of each document. Thus, this can be expressed as $C^D = QA^D$. While calculating the question to document context, we also calculate, the summaries in the light of the previous attention context which can be expressed as $C^Q A^D$ which can be considered as a mapping of question encoding in to the space of document encoding. These two operations can be done in parallel. Finally, this C^D is refered to as the **co-attention context**. The final step in the encoder layer feeding temporal information with co-attention context. This is done via bi-directional LSTM and is defined as follows:

$$u_t = \text{Bi-lSTM}(u_{t-1}, u_{t+1}, [d_t, c_t^D]) \tag{9}$$

Thus, we can define $U = [u_1, u_2, ..., u_t]$ as a foundation for selecting which span may be the best possible answer.

3.3.3 Dynamic Pointing Decoder

As we have discussed before, due to the nature of the SQuAD dataset, one of the many possible ways of generating answer is by predicting the start and end points of the span. Now, since there may be several starting and ending points, the idea is to iterate over all possible spans to select an answer span by alternating between predicting the start point and predicting the end point. Thus, this helps the model to achieve the correct global maxima and recover from several initial local maxima. Here denote the hidden state of LSTM, start point and end point as h_i. s_i, e_i, then LSTM state update can be written as

$$h_i = \text{LSTM}_{dec}(h_{i-1}, [u_{s_{i-1}}, u_{e_{i-1}}] \tag{10}$$

In the above equation, $[u_{s_{i-1}}$ and $u_{e_{i-1}}$ denote the previous states of the start and end encoding of U. Following this, we find the current start and end position based on the previous states of these. Thus we get,

$$\begin{aligned} s_i &= argmax(\alpha_1, \alpha_2, ..., \alpha_m) \\ e_i &= argmax(\beta_1, \beta_2, ..., \beta_m) \end{aligned} \tag{11}$$

In the above equation, α and β denote the start and end score of the t^{th} word in the document. Here, α and β are calculated according to a Highway Max-out Network which has shown strong empirical performance with regard to deep architectures. The max-out network provides a simple and effective way to pool across multiple model variations.[13]

3.4. Implementation Details

In this section, we briefly describe the implementation of the data and model architectures.

3.4.1 Data Preprocessing

We train and evaluate our model on the SQuAD dataset. To pre-process the corpus, we use the tokenizer from Stanford CoreNLP. We use as GloVe word embeddings with 100 dimensions.

3.4.2 Dynamic Co-Attention Network Setup

We use a max sequence length of 600 during training and a hidden state size of 200 for all recurrent units, max-out layers, and linear layers. All LSTMs have randomly initialized parameters and an initial state of zero. Sentinel vectors are randomly initialized and optimized during training.

3.4.3 Bidirectional Attention Flow Model Setup

We use a max sequence length of 600 for the context and 30 for questions during training and a hidden state size of 200 for all recurrent units, max-out layers, and linear layers. All LSTMs have randomly initialized

parameters and an initial state of zero. The dropout for this model is set to 0.2, learning rate 5e-3, and batch size 24

4. Experiments

4.1. Dataset

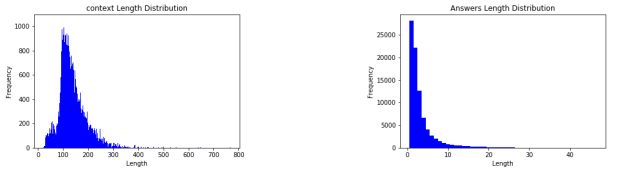
Stanford Question Answering Dataset (SQuAD) [9] is a reading comprehension dataset, consisting of questions posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage, or the question might be unanswerable.
More specifically we use the SQuAD2.0 version of the dataset. SQuAD2.0 combines the 100,000 questions in SQuAD1.1 with over 50,000 new, unanswerable questions written adversarial by crowdworkers to look similar to answerable ones. To do well on SQuAD2.0, systems must not only answer questions when possible, but also determine when no answer is supported by the paragraph and abstain from answering. The dataset is randomly partitioned into a test set(80%), a development set(10%) and a test set(10%). The test set is not released to the public. In Figure 1 we explore the dataset to gain insights about context, question, answer and word distribution. Based on the distributions,we set the context length to 400, answer length to 15, question length to 30 and maximum character length of word length to 15. Smaller maximum lengths helps us to speed up the training process.

4.2. Evaluation of models

Evaluation on the SQuAD dataset consists of two metrics. The exact match score (EM) calculates the exact string match between the predicted answer and a ground truth answer. The F1 score calculates the overlap between words in the predicted answer and a ground truth answer. Because a document-question pair may have several ground truth answers, the EM and F1 for a document-question pair is taken to be the maximum value across all ground truth answers.

4.2.1 Baseline Model

The baseline model has been trained and the plots of the EM, F1 score for development set has been recorded for 15k iterations as 40 and 29 as shown in figure 4. After 15k iterations, we notice that the development loss starts increases, at this point we realise that the model has started to over-fit.



(a) Context Length Distribution

(b) Answer Length Distribution

(c) Question Length Distribution

(d) Word Length Distribution

Figure 3: Histograms of Context, Answer, Question and Word Lengths in Training Set

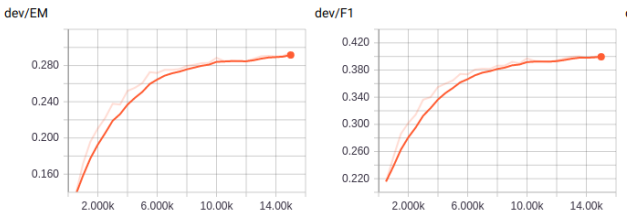


Figure 4: F1 and EM for the baseline model

4.2.2 Experiments for DCN and BiDAF

We achieved F1 and EM scores of 71.8 and 59 over the development set for the BiDAF model, as can be seen from figure 5. From the results, we can see that the performance is comparable to state of the art machine comprehension methods. The performance gap with the reference implementation of BiDAF can be explained by lack of character level embeddings

We achieved F1 and EM scores of 55 and 40.1 over the development set for the DCN model, as can be seen from figure 6. On the decoder side, we have experimented with various pool sizes for the HMN layers. After several analysis, we found the best performance when pool size is equal to 16. The following table

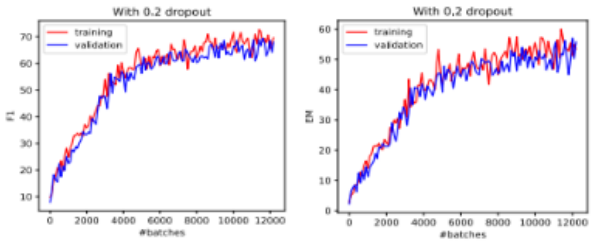


Figure 5: F1 and EM for the BiDAF model

Model	Dev F1 Score	Dev EM Score
Dynamic Co-Attention Network		
Pool size 16 HMN	55	41
Pool Size 8 HMN	53	39
Pool Size 4 HMN	51	35

Figure 6: F1 and EM for the DCN model

MODEL	F1 score	EM score
Baseline- Logistic Regression (first squad paper)	51.0	40.0
Baseline – RNN with Attention (our implementation)	39.5	28.2
Dynamic Coattention network (our implementation)	55.0	41.0
Bidirectional Attention flow (our implemention)	71.8	59.0
Human Performance	91.2	82.3

Figure 7: Results on Development Set

illustrates the F1 and EM Scores for each of these experiments.

5. Results

Figure 7 lists the performance of our implementations of BiDAF and DCN with our baseline model. We have also compared with the baseline performance from the first squad paper using Logistic Regression. We have also mentioned the human performance for reference.

6. Conclusion and Future Work

Our implementation has clarified that in the current setting of experimentation, the BiDAF model performs well, despite it's rather simple bi-directional attention mechanism. The DCN model could not surpass the BiDAF model, but have performed significantly better than both the baseline models.

We believe we were able to achieve compentent results given the constraints in time and resources, but there is still a large scope of improvement toward the state-of-art performance.

- We can try an ensemble of models for the same architecture and among different architectures. This decreases the miss rate, and helps achieve higher accuracies.
- We can also try to use a larger vocabulary size, like the Common Crawl dataset, it contains 2.2mil words. This could decrease the OOV <unk >at test time.
- We could also try a larger embedding size, to learn a richer representation for each word.
- We could also learn a richer representations for word embeddings like Embedding from Language modeling [8] or like learn embeddings from self attention. [2]
- The BiDAF could have learnt a richer model if character level modeling using CNN was implemented like its reference implementation. [7]
- Experiment with iterative reasoning based techniques which use iterative attention to arrive at the answer. [5]
- We could also use adversial training using attention supervision to answer the query. [1]

References

[1] M. Cho, R. K. Amplayo, S. Hwang, and J. Park. Adversarial tableqa: Attention supervision for question answering on tables. *CoRR*, abs/1810.08113, 2018.

[2] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[3] L. Hirschman, M. Light, E. Breck, and J. D. Burger. Deep read: A reading comprehension system. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, ACL '99, pages 325–332, Stroudsburg, PA, USA, 1999. Association for Computational Linguistics.

[4] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, Nov. 1997.

[5] D. A. Hudson and C. D. Manning. Compositional attention networks for machine reasoning. *CoRR*, abs/1803.03067, 2018.

[6] S. Merity, C. Xiong, J. Bradbury, and R. Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[7] A. F. H. H. Minjoon Seo, Aniruddha Kembhavi. Bidirectional attention flow for machine comprehension. *ICLR*, abs/1606.05250, 2017.

[8] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. *CoRR*, abs/1802.05365, 2018.

[9] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[11] S. Wang and J. Jiang. Learning natural language inference with LSTM. *CoRR*, abs/1512.08849, 2015.

[12] S. Wang and J. Jiang. Machine comprehension using match-lstm and answer pointer. *CoRR*, abs/1608.07905, 2016.

[13] C. Xiong, V. Zhong, and R. Socher. Dynamic coattention networks for question answering. *CoRR*, abs/1611.01604, 2016.

[14] Y. Yu, W. Zhang, K. S. Hasan, M. Yu, B. Xiang, and B. Zhou. End-to-end reading comprehension with dynamic answer chunk ranking. *CoRR*, abs/1610.09996, 2016.