Machine Reading Comprehension on SQuAD 2.0

Yancheng Li, Shichang Zhang
Department of Statistics
Stanford University
{lycheng, shichang}@stanford.edu

Yiyang Li
Department of Electrical Engineering
Stanford University
yiyang7@stanford.edu

Abstract

In this paper, we present an end-to-end neural network model for question answering on the SQuAD 2.0. Our model features ideas from the Hierarchical Attention Fusion Networks to combine representations from multi-level granularity. It achieves 57.9 F1 and 54.4 EM score on test set on the Non-PCE leaderboard; 66.6 F1 and 63.7 EM on development set.

1 Introduction

Machine reading comprehension (MRC) is a challenging task, where the goal is to have systems read a text passage and then answer any question about the passage. This task is an useful benchmark to demonstrate natural language understanding, and also has many applications, e.g. conversational agents and customer service support. Stanford NLP group presented the Stanford Question Answering Dataset (SQuAD) 2.0 [1], which combines exsiting SQuAD 1.1 with over 50,000 unanswerable questions. To do well on the new dataset, systems must not only answer questions when possible, but also determine when there's no answer for a question.

In this project, we built an end-to-end deep learning model for question answering on the SQuAD 2.0. Our model features ideas from the Hierarchical Attention Fusion Networks to combine representations from multi-level granularity [2]. The rest of this paper is organized as follows: We discussed related work inSse tion 2. cection 3 introduces out model architecture and approach. Section 4 describes the SQuAD dataset, experiment details, and results. Section 5 gives both quantitative and qualitative analysis of the model and we concluded the paper in Section 6.

2 Related Work

Traditional question answering systems relies heavily on linguistic annotation, structured world knowledge, semantic parsing and etc. [3]. Recently, machine reading comprehension has largely beneted from the availability of large-scale benchmark datasets and it is possible to train large end-to-end neural network models. Moreover, [4] use additional features other than word vectors, such as Part-of-Speech tag (POS), the named-entity recognizer (NER) tags and etc., to improve model's performance. Another key in MRC task is how to incorporate the question context into the paragraph, where attention mechanism is most widely used [5] [6] [7]. [8] introduced the Bi-Directional Attention Flow (BIDAF) network to represent the context at different levels of granularity and uses bidirectional attention flow mechanism to obtain a query-aware context representation without early summarization[8].

CS224n Winter 2019, Stanford University, CA. (LateX template borrowed from NIPS 2018.)

3 Approach

We implement the end-to-end hierarchical attention fusion MRC model (shown in Figure 1) from scratch using Pytorch deep learning framework. Our is a multi-stage network is composed of the following 6 components:

Lexicon Embedding Layer transforms a sequence of discrete word tokens into continuous word embedding vectors for semantics representation. We adopt the pre-trained GloVe embeddings.

Contextual Encoding Layer employs Bi-LSTM to refine the coarse word-embeddings and obtain contextual representations of both query and passage.

Co-Attention Layer captures the relationship between query and passage by using the hierarchical fusion kernel that combines representation from different ganularities.

Self-Attention Layer employs a bilinear self-attention function to address the long-distance dependency within different contexts [2].

Matching and Output Layer inherits the idea of dropout and bilinear matching function to locate the span of final answer in the corresponding passage for each query.

AvNA Classifier applies two fully-connected layers followed by nonlinear activation function for Answer vs. No Answer binary classification.

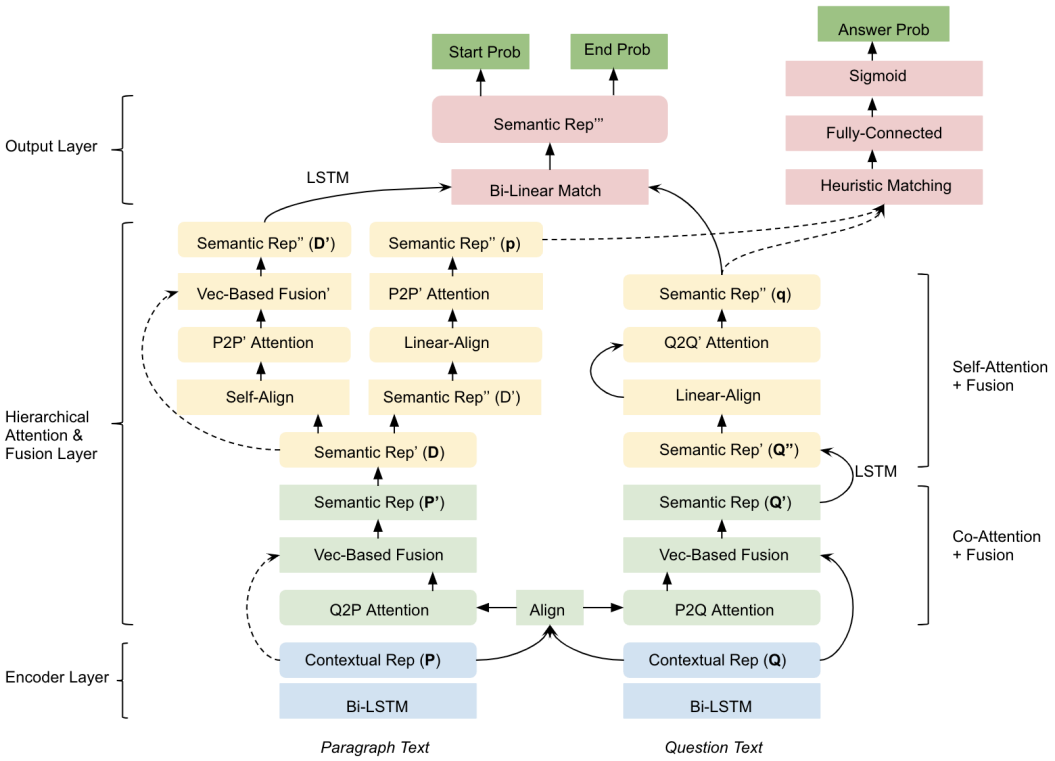

Figure 1: Hierarchical Attention Fusion Network

3.1 Lexicon Embedding Layer

We use the pre-trained word embedding vectors GloVe [9] with embeded size $d_w = 300$ for lexicon embeddings to extract information from passages $P = \{p_0, ..., p_{n-1}\}$ and queries $Q = \{q_0, ..., q_{m-1}\}$. As a result, we obtain the word-level embeddings $\{e_t^P\}_{t=1}^n$ and $\{e_t^Q\}_{t=1}^m$, where each token within the dataset in represented as a 300-dimensional vector. We also add a 300-

dimension character-level embeddings $\{c_t^P\}_{t=1}^n$ and $\{c_t^Q\}_{t=1}^m$by applying char-CNN and highway module (reference CS224n assignment 5 for detail).

3.2 Contextual Encoding Layer

To better extract the contextual information from surrounding words, we first apply a 1-layer bidirectional LSTM (bi-LSTM) to the concatenated word and character embeddings.

$$u_t^P = [\text{BiLSTM}_P(e_t^P, c_t^P), e_t^P] \tag{1}$$

$$u_t^Q = [\text{BiLSTM}_Q(e_t^Q, c_t^P), e_t^Q] \tag{2}$$

This contextual output from bi-LSTM is then concatenate with the original word-embeddings. This serves as the final output of the contextual encoding layer of our model, with both encoded contextual information and the raw lexicon level representation.

3.3 Co-Attention Layer

In this attention layer, we construct the working memory that fuses the information from passages and queries. We employ the bi-directional attention flow mechanism to link the representations from both sources. This layer aims to better understand to query and align it with passages in order to generate the question-aware context representation. Our co-attention layer inherits the hierarchical attention structure as in [2]. We applied the fusion function that combines representations from different granularities to better the model's semantic learning process.

Specifically, we first compute the similarity matrix S, and then obtain the context-to-question attention and question-to-context attention:

$$S_{ij} = \text{Att}(u_t^Q, u_t^P) = \text{ReLU}(W_s^T u_t^Q)^T \text{ReLU}(W_s^T u_t^P) \tag{3}$$

where W_s is a trainable weight matrix.

Context-to-Question (C2Q) attention: For each every single word in the passage, C2Q attention generates the weighted sum of query-word representations according to the relevancy between the query word and the context word. In detail, the attention weights for each query word with respect to the context word is given by:

$$\alpha_j = \text{softmax}(S_{:j}) \tag{4}$$

Hence, the aligned context representation is obtained by:

$$\tilde{Q}_{:t} = \sum_j \alpha_{tj} Q_{:j}, \forall j \in [1, ..., m] \tag{5}$$

Question-to-Context (Q2C) attention: Similar to the C2Q attention described above, Q2C attention provides the attention weights of each context word with respect to a given query word. This allows us to integrate information from the most similar span of words in the passage for each question. The computation is shown below:

$$\beta_i = \text{softmax}(S_{i:}) \tag{6}$$

Thus, the full attended question representation is given by:

$$\tilde{P}_{k:} = \sum_i \beta_{ik} P_{i:}, \forall i \in [1, ..., n] \tag{7}$$

Hierarchical fusion kernel: To further improve the effectiveness of our model's learning process, we inherit the idea of hierarchical connection that fuses representations for passage and query, separately. Referring to [2], we apply the matrix-based fusion function, $\phi(\cdot, \cdot)$. The first step is to use the matching trick for combining different representations as suggested in [10] and [11]. After that we project the concatenated tensor back to the dimension of the original representation and plut it into the $\tanh(.)$ function.

$$P_f = \phi(P, \tilde{Q}) = \text{tanh}(W_f^p [P; \tilde{Q}; P \circ \tilde{Q}; P - \tilde{Q}] + b_f^p) \tag{8}$$

$$Q_f = \phi(Q, \tilde{P}) = \text{tanh}(W_f^q [Q; \tilde{P}; Q \circ \tilde{P}; Q - \tilde{P}] + b_f^q) \tag{9}$$

where $\circ$ denotes the element-wise product, [...] denotes vector/matrix concatenation, and W_f, b_f are trainable parameters.

Inspired from human reading pattern, we employed the gating mechanism to ensure that we focus mainly on the most relevant part of the passage when given a certain query and, conversely, addressing to the right part of the query when going over a passage. Therefore, the final output of this co-attention layer is the gated combination of fused representation and the original contextual representation:

$$P' = g(P, \tilde{Q})P_f + (1 - g(P, \tilde{Q}))P \tag{10}$$

$$Q' = g(Q, \tilde{P})Q_f + (1 - g(Q, \tilde{P}))Q \tag{11}$$

where $g(\cdot, \cdot)$ is the vector-based fusion kernel:

$$g(P, \tilde{Q}) = \sigma(w_g^T \cdot [P; \tilde{Q}; P \circ \tilde{Q}; P - \tilde{Q}] + b_g) \tag{12}$$

3.4 Self-Attention Layer

The previous co-attention layer captures the dependency information between queries and questions. In this layer, we further refine the result from the co-attention layer, but we consider queries and passages separately. This is meant to address the long distance dependence problem and allow the contextual information to flow within passages and queries. To achieve this goal, we first pass the passage representation through a bidirectional LSTM, and then we adopt the idea of bilinear self-alignment attention from [2].

$$D = \text{BiLSTM}(P') \tag{13}$$

$$L = \text{softmax}(DW_1D^T) \tag{14}$$

$$\tilde{D} = LD \tag{15}$$

Then we perform one more step fusion to combine this self-aligned passage representation with the question-aware passage representation from co-attention layer.

$$D' = \phi(D, \tilde{D}) \tag{16}$$

After another bidirectional LSTM, we are ready to use this passage representation for matching and output:

$$D'' = BiLSTM(D') \tag{17}$$

Queries, on the other hand, is not very long and contains less information. According to [4], we employ a linear rather than bilinear self-alignment to save parameters. Before doing that we apply a bidirectional LSTM first as for passage representations. Then the final representation of queries is the following:

$$Q^{''} = \text{BiLSTM}(Q') \tag{18}$$

$$\gamma = \text{softmax}(w_q^T Q'') \tag{19}$$

$$\mathbf{q} = \sum_j \gamma_j Q''_{:j}, \forall j \in [1, ..., m] \tag{20}$$

The final representation D'' and $\mathbf{q}$ will be passed to the matching layer for generating the output span of our answer. Besides, we construct a another representation $\mathbf{p}$ similarly to $\mathbf{q}$ for the classification task. More details about the classification layer can be find in section 2.6.

$$\tau = \text{softmax}(w_p^T D') \tag{21}$$

$$\mathbf{p} = \sum_j \tau_j D'_{:j}, \forall j \in [1, ..., m] \tag{22}$$

3.5 Matching and Output Layer

Due to the fact that gold-key for queries in SQuaD 2.0 with exact answer are all continuous subspans of the context, our model can generate the answer by predicting the start and end position of the answer. Our span detector adopts the stochastic prediction dropout feature, as in [12], which significantly improves the model efficiency and prediction robustness. The notion of stochastic is inherited into the multi-step reasoning neural network at different stages to ensure the model is learning the precise span distribution at every individual step.

Specifically, we have our final self-aligned representation D'' and $\mathbf{q}$ for passages and queries respectively. At each timestamp $t \in 1, 2, \ldots, T$, we compute the hidden state input for next timestamp h_t, where $h_0 = \mathbf{q}$ is query self-aligned representation:

$$h_t = \text{GRU-Cell}(h_{t-1}, q'_t) \tag{23}$$

where q'_t is computed from contextual representation D'' and previous hidden state h_{t-1}. We then apply the bilinear mechanism at each step t to predict the start and end index at each timestamp:

$$P^t_{start} = \text{softmax}(q'_t W_s^T D'') \tag{24}$$

$$r^t_{end} = [h_t; \sum_j P^{t,j}_{start} D''_j] \tag{25}$$

$$P^t_{end} = \text{softmax}(r_t W_e^T D'') \tag{26}$$

During the training process, our matching layer performs stochastic dropout on the final predictive distributions. Specifically, we apply stochastic dropout on top of these T probability distributions, then the leftover ones are averaged to obtain the final log-probability distribution over the whole context, for start and end position respectively.

The idea of stochastic prediction dropout is to randomly ignore the output from several steps and average the rest of predictions to generate the final output. This forces the model to learn an accurate prediction at every individual step. The intuition behind is different from the idea of applying dropout to neurons of hidden layers. Intermediate level of neuron dropout ignores a randomly chosen set of neurons during forward passes, which serves to de-correlate features and prevent overfitting. Whereas the stochastic prediction dropout is mainly designed for stopping the model from making predictions mainly based on a particular step. For validation or test purpose, the predicted answer is generated according to the average of the predictions across all steps, without applying stochastic prediction dropout. Disabling stochastic dropout prevents us from getting different output for each decoding.

3.6 AvNA Classifier

The idea of adding an extra classifier to enhance the AvNA accuracy is adopted from [12]. In our particular case, the classifier makes use of the $\mathbf{q}$ and $\mathbf{p}$ from the self attention layer. The first step is to concatenate $\mathbf{q}$ and $\mathbf{p}$ using the heuristic matching trick as we were doing in the fusion function. Then we pass the result through a three-layer fully connected neural network with ReLU activation to do binary classification. Our classifier returns a probability indicating how likely this question is answerable.

$$X = \text{FullyConnected}([Q; \tilde{P}; Q \circ \tilde{P}; Q - \tilde{P}]) \tag{27}$$

$$P_{score} = \text{sigmoid}(X) \tag{28}$$

One more classifier using convolutional neural network is also explored. See section 3.4 for the comparison.

4 Experiments

4.1 Data

We trained, validated, and compared our model on the SQuAD 2.0 dataset, consisting of 150,000+question-answer pairs on 500+ Wikipedia articles where the answer to each question is a span takenfrom the article. In addition, the dataset also includes over 50,000 unanswerable questions that are generatedby changing the rule-based procedure for editing original SQuAD questions [1]. This new version of SQuAD dataset forces the model to comprehend the both questions and passages more thoroughly.

4.2 Evaluation method

The performance of all the models in this project are evaluated using EM score and F1 score. We also use a AvNA binary classification accuracy to measure how good our model is for separating answerable questions for not-answerable questions

4.3 Experimental details

Our model is designed to jointly optimize the answer span accuracy and answerable v.s not-answerable classification accuracy. We use cross-entropy loss for the answer span detection and binary cross-entropy(BCE) loss for classification. When the question is answerable with the gold start position i and end position j, our the objective function is the following.

$$L_{span} = -\log p_{start}(i) - \log p_{end}(j) \tag{29}$$

$$L_{classify} = -\mathbf{1}_{\{answerable\}} \log P_{score} - (1 - \mathbf{1}_{\{answerable\}}) \log P_{score} \tag{30}$$

$$L_{joint} = L_{span} + L_{classify} \tag{31}$$

After a broad search of different combinations of hyperparameters on a sub-dataset, we decided to train the network using Adam optimizer, with a hidden size of 128 and mini-batch size of 32 for 20 epochs. With the learning rate equal to 0.001. A 15-epoch training process takes roughly 20 hours on a single Nvidia Tesla M60 GPU. To prevent overfitting, we use a 0.4 dropout for all LSTM layers.

4.4 Comparison of Classifiers

Besides the fully connected neural network classifier mentioned above. We also implement another classifier based on convolutional neural network(CNN). This CNN classifier uses the representation produced by the co-attention layer, i.e. P' and Q' above. We first apply a 2-dim convolution to the concatenation of these two representations. Then we use max-pooling followed by logistic regression to get a prediction score. Again, this score is the probability of how likely a certain question is answerable.

$$C = Conv2D([P; Q]) \tag{32}$$

$$M = MaxPool2D(C) \tag{33}$$

$$P_{score} = \text{logistic}(M) \tag{34}$$

We compare the performance of these two classifiers after 20 epochs training on a subset with 10,000 training examples and evaluate the AvNA accuracy on 1000 dev examples. In fact two results are fairly similar.

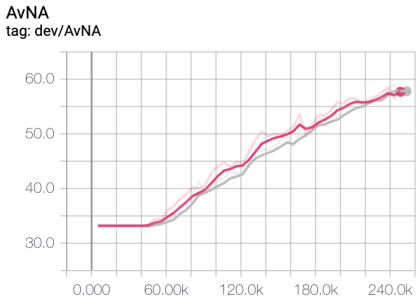

Figure 2: Comparing the Performance of Two Classifiers (Pink: Fully Connect; Grey: CNN)

4.5 Results

We submitted our results to non-PCE leaderboard. The result comparison with other models is showing in the following table. According to Table 1, we can see that our model achieves competitive F1 and EM socres on dev set but with lower score on test set. Though we expected for a slight decrease, this huge decreases are unexpected. Further analysis indicates that our model is too sensitive to the distribution differences between datasets. The gap in performance is caused by difference in distribution of question type and length of answer spans.

	Dev Set	Test Set
Models	EM/F1	EM/F1
BiDAF baseline	55.9/59.2	-/-
BNA	59.8/62.6	59.2/62.1
DocQA	61.9/64.8	59.3/62.3
SAN	69.3/72.2	68.7/71.4
SLQA+ (with ELMo)	-/-	71.4/74.4
Human	-/-	82.3/91.2
Our Model	**63.7/66.6**	**54.4/57.9**

Table 1: SQuAD 2.0 Models Performance Comparison

5 Analysis

5.1 Analysis by Question Type

A closer look at our model is done by monitoring the evaluation metric by question type. According to Figure 3, we observe that our model perform extremely well when answering "whe" and "which" types of questions. However, the model performs much worse when predicting answers for "why" and "other" types of questions.

This result is somewhat expected, because "why" type of question may involve more logical reasoning across context and the model is not so good at retrieving these complicated reasoning underneath; while "when" and "which" type of questions typically have shorter dependency between object and target answer. As for "other" type of question, due to the large variation in query statement but limited training samples, it is tolerable for the model not performing well on capturing the relationship between the query and context. In addition, we found that the type of questions with larger average span length typically have lower F1/EM score than those with shorter spans. For instance, "why" type of questions' answer span is three times as long as the span for "when" type and twice as long as the span for "which" type, hence leading to less desired performance.

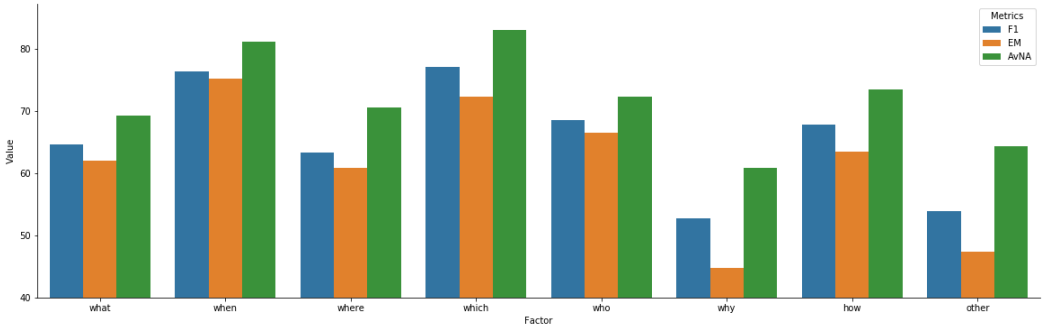

Figure 3: Model Performance Breakdown by Question Type

5.2 Error Analysis

We now look closely to the questions and compare the actual answer with our predicted answer to understand the weakness of our model. Two examples are discussed below.

5.2.1 Example 1: Failure of Attention Mechinism

Passage: A function problem is a computational problem where a single output (of a total function) is expected for every input, but the output is more complex than that of a decision problem, that is, it isn't just yes or no. Notable examples include the traveling salesman problem and the integer factorization problem.
Question: A function problem is an example of what?
Gold Answer: a computational problem

Predicted Answer: traveling salesman problem and the integer factorization problem

The span our model predicts is the last part of the passage. This error is could because of the attention mechanism we use. For this question, the last sentence contains "Notable example", "traveling salesman problem", and "factorization problem". When we apply co-attention between words in passages and questions, this part is likely to be assigned a higher similarity score. One possible way to deal with this kind of issue could be using different attentions.

5.2.2 Example 2: Tedious Logical Reasoning

Passage: Internet2 is a not-for-profit United States computer networking consortium led by members from the research and education communities, industry, and government. The Internet2 community, in partnership with Qwest, built the first Internet2 Network, called Abilene, in 1998 and was a prime investor in the National LambdaRail (NLR) project. In 2006, Internet2 announced a partnership with Level 3 Communications to launch a brand new nationwide network, boosting its capacity from 10 Gbit/s to 100 Gbit/s. In October, 2007, Internet2 officially retired Abilene and now refers to its new, higher capacity network as the Internet2 Network.
Question: Abilene was a prime investor in what project?
Gold Answer: NO ANSWER
Predicted Answer: National LambdaRail

Having passages with no answers in it is a new challenge brought up by SQuAD 2.0 comparing to SQuAD 1.1. In the example above, this not-answerable question is especially tricky. The key word "Abilene" in the question was matched with the "Abilene" in the second sentence in passage. Also, there is a phrase "was a prime investor in" quite close following "Abilene". All this information is captured by our model, which makes the model pretty confident to conclude what comes after is the correct answer. However, the verb "was" here actually refers to "the Internet2 community" in the beginning of the sentence. Possible solutions for this kind of tricky issues could be adding more linguistic features to help our model capture the dependence in sentences.

6 Conclusion

In this paper, we present an end-to-end neural network model for question answering on the SQuAD 2.0. Our model features ideas from the Hierarchical Attention Fusion Networks to combine representations from multi-level granularity and stochastic span detection module. It achieves 66.6 F1 and 63.7 EM score on the development dataset on the Non-PCE leaderboard and a relatively lower score on test set. From both quantitative and qualitative analysis of the model, we found that question type and span length could effect F1/EM scores and the structure of our model could result in certain types of errors listed above. The above analysis indicates that our hierarchical attention fusion mechanism is able to capture the most related information between queries and context and synthesize the information to predict the right answer span.

We realize that the major limitation of our model is too sensitive to the distribution between datasets. Our performance on test set is worse than its performance on the dev set, possibly due to the difference in distribution of question type. Our model is significantly good at answering a certain type of question, while struggling with the type of question involving complicated reasoning. As all our parameter tuning and checkpoint selection are done on dev set, our model may not result as good score on test as on dev. Tons of experiments indicates that answer module is not effective enough in solving this uneven distribution problem. To bridge the performance gap, we plan to try following approaches, including exploring and applying multi-head attention on top of our model to improve model efficiency when having multi-reasoning steps, augmenting dev set to balance the number of sample of each type of questions of various lengths, adding pre-trained embeddings like ELMo/BERT.

References

[1] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*, 2018.

[2] Wei Wang, Ming Yan, and Chen Wu. Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. *arXiv preprint arXiv:1811.11934*, 2018.

[3] Edward Grefenstette Lasse Espeholt Will Kay Mustafa Suleyman Karl Moritz, Tomas Kocisky and Phil Blunsom. Teaching machines to read and comprehend. 2015.

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

[5] Jason Weston Danqi Chen, Adam Fisch and Antoine. Reading wikipedia to answer opendomain questions. 2017.

[6] Weiguang Mao Rui Liu, Wei Wei and Maria Chikina. Machine comprehension using match-lstm and answer pointer. 2016.

[7] Dipanjan Das Ankur Parikh, Oscar Tackstrom and Jakob Uszkoreit. A decomposable attention model for natural language inference. 2016.

[8] Ali Farhadi Minjoon Seo, Aniruddha Kembhavi and Hannaneh Hajishirzi. Bidirectional attention ow for machine comprehension. 2016.

[9] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[10] Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. Natural language inference by tree-based convolution and heuristic matching. *arXiv preprint arXiv:1512.08422*, 2015.

[11] Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, and Diana Inkpen. Natural language inference with external knowledge. 2018.

[12] Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. Stochastic answer networks for machine reading comprehension. *arXiv preprint arXiv:1712.03556*, 2017.