An Approach to Machine Reading Comprehension on SQuAD

Jiafu Wu
Electrical Engineering
Stanford University
Stanford, CA 94305, USA
jiafuwu@stanford.edu

Alan Flores-Lopez
Computer Science
Stanford University
Stanford, CA 94305, USA
alanf94@stanford.edu

Abstract

In this paper we present our approach to tackling the problem posed by the Stanford Question Answering Dataset, SQuAD. SQuAD is an effective dataset to train and test the task of machine reading comprehension – it offers a large number of real questions and real contexts from which to select answers. Our architecture is based on techniques that have performed well on the SQuAD dataset, including an embedding layer with a character-level CNN, BiDAF and self-attention at the attention layer, and pointer-network-based span selection in the output layer. Our single model achieved a dev F1 score of 74.18% and a dev EM score of 62.69%.

1 Introduction

SQuAD, or the Stanford Question Answering Dataset, (Rajpurkar et al., 2016) provides an effective testbed for reading comprehension style question answering, in which the answer to a question is a span selected from a certain context. In this paper we present the end-to-end approach we constructed from modules that build off from work mainly by Seo et al. 2016, Wang & Jiang 2016, Chen et al. 2017, and Microsoft's 2017 R-NET model [1].

Additionally, we argue that while hand-crafted features like those used by Chen et al. significantly improve a baseline model and are computationally cheap, they are redundant when utilizing more powerful attention mechanisms. We also present an alternative method of smart span selection to that given in Chen et al., 2017, and we analyze the performance of our model on various question types in the SQuAD dataset, and the extent to which our best model improves our baseline with respect to different question types.

2 Related work

The SQuAD dataset has seen rapid progress since its release in 2016. Since the beginning of 2018, several ensemble models have surpassed human-level exact-match performance (EM 82.304) [2]. Yet as of March 2018, no single model submission has reached human level performance. The best single model was submitted by Google Brain and CMU under the name QANet, with F1 and EM scores of 87.773 and 80.929 respectively.

An important technique to increase the performance is to use attention, and almost all of the top-performing models use some sort of attention. For example, Bidirectional Attention Flow (BiDAF) uses question-to-context and context-to-question attention to capture the two-way interaction between the question and the context (Seo et al., 2016), R-NET uses self-attention in the context to

[1] Microsoft R-NET paper.
[2] SQuAD leaderboard.

dynamically collect evidence from the whole passage and encode the evidence relevant to the current passage word into the passage representation[1] or attention at the output layer (Wang & Jiang, 2016).

Another important method is to add more input features for each input word. In BiDAF, it utilizes pre-trained Glove word embedding as well as character-level CNN embedding(Seo et al., 2016). On the other hand, DrQA included several input features such as exact match, token features which include the Part-of-Speech tag, the Named Entity type and the Normalized Term Frequency, as well as aligned question embedding which used the word embedding to attend to the word embeddings for the question and use the resulting attention output vector as an additional feature (Chen et al., 2017).

Less standard techniques have been attempted with various degrees of success, like predicting the probability of a span directly instead of predicting answer start and end positions (Yu et al., 2016).

3 Baseline

We began work from a baseline with three components: 1) a **RNN encoder layer** to encode the question and the context, 2) an **attention layer** that combines the question and context representation and provides basic context-to-question attention, and 3) an **output layer** that down-projects the hidden representations and runs two independent softmax layers to get the start and end location of the answer span, (s, e).

4 Our Approach

Much like the approach taken in BiDAF, we can describe our model in terms of five layers: 1) a **representation layer** for word embeddings and character-level encodings, 2) a **contextual embedding layer** that coalesces the encodings in the previous layer, 3) an **attention flow layer** in which the context and question representations are combined with bidirectional attention and self attention, 4) a **modeling layer** that takes the mixed context-question representations from the previous layers into their final form, and finally 5) an **output layer** that ingests that final form to predict the start and end positions of an answer. Additionally, we use a new method of smart span selection at test time that gives preference to the answer boundary that the model is most confident about.

In the following descriptions, $C \in \mathbb{R}$ is the length of the context, $Q \in \mathbb{R}$ is the length of the question, and $h \in \mathbb{R}$ is the size of the hidden representation, or the output of the contextual embedding layer.

4.1 Representation Layer

We use 100-dimensional pre-trained Glove embeddings (Pennington et al., 2014). Additionally, we also include character-level CNN (Kim, 2014). For our character-level CNN, we first embed each individual character into a vector representation of length 20; for simplicity, we choose to only embed digits, lower-case characters while converting all the upper-case ones to lower-case, and some punctuations. Then we embed each character vector using a one-dimensional CNN with a window width of 5 and an output dimension of 100. We then perform max pooling on all the character embeddings for a word and obtain the final vector representation by concatenating the Glove word embedding and the character-level CNN embedding. The Glove word embedding is an effective model to capture the semantics of words in a large vocabulary, while our own character-level CNN helps deal with out-of-vocabulary words.

4.2 Contextual Embedding Layer

Similar to BiDAF, we feed the representations of context and question vectors into a contextual embedding layer for RNN encoding. A slight modification is that we utilize GRU instead of LSTM because they have similar performance while GRU is more computationally efficient. Using this 1-layer bidirectional GRU, we are able to further refine our word embedding by providing the context of the words. Since the embedding is still focusing on the word level, we share the weights of the GRUs for both the context and question embedding. Then we concatenate the forward and backward

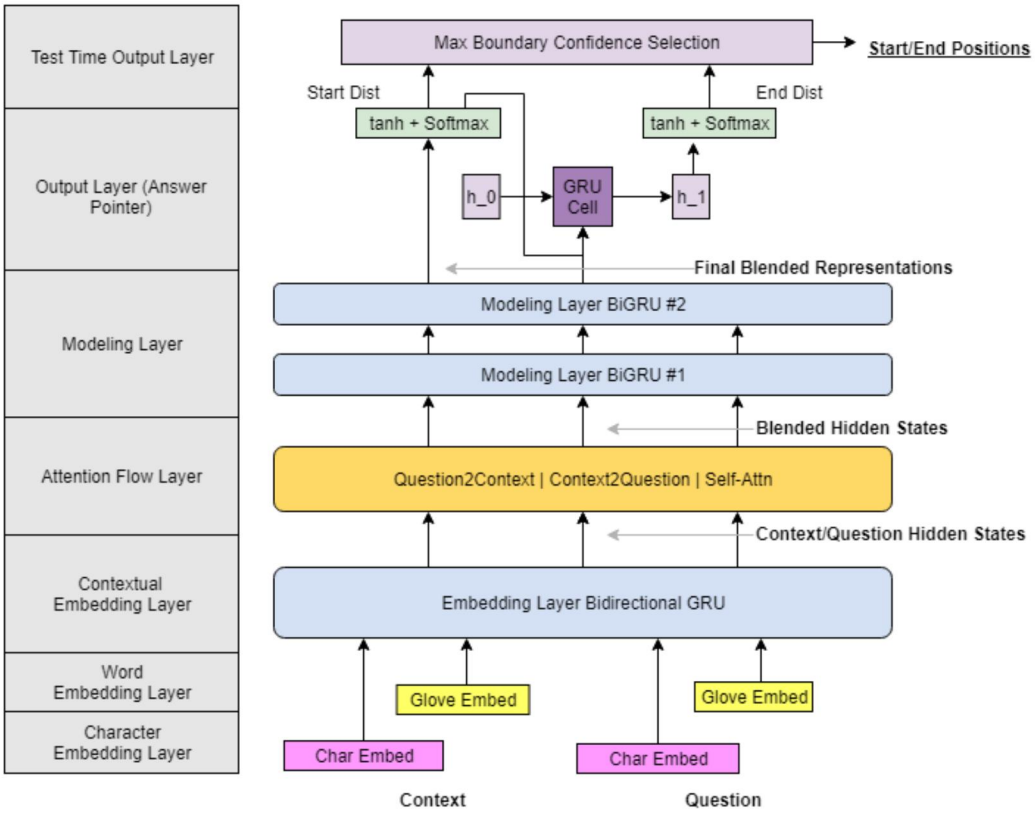

Figure 1: Model architecture diagram

outputs of the context and word embedding to get the context hidden state $c_i = [\overrightarrow{c_i}, \overleftarrow{c_i}] \in \mathbb{R}^{2h}$ and question hidden state $q_i = [\overrightarrow{q_i}, \overleftarrow{q_i}] \in \mathbb{R}^{2h}$.

4.3 Attention Flow Layer

The next layer of our model includes several types of attention: Context-to-Question(C2Q), Question-to-Context(Q2C), and self-attention. The first two attentions are taken from BiDAF, while the last one was used in R-NET.

In order to compute the C2Q and Q2C attentions, we first need to get the similarity matrix $S \in \mathbb{R}^{C \times Q}$ using the context hidden state $c_i \in \mathbb{R}^{2h}$ and question hidden state $q_i \in \mathbb{R}^{2h}$.

$$S_{ij} = w_{sim_1}^T c_i + w_{sim_2}^T q_j + w_{sim_3}^T c_i \circ q_i \in \mathbb{R} \tag{1}$$

We make a modification here. In its original form, the attention mechanism takes the concatenation of the vector representations $[c_i; q_j; c_i \circ q_i]$. But it is more memory-efficient to perform the matrix multiplication and addition since the concatenation requires more memory.

4.3.1 C2Q Attention

For C2Q attention, we take the row-wise softmax of S to obtain the attention distribution of context hidden states attending on each question hidden state, and we use that for a weighted average sum of the question hidden states, which is the C2Q Attention output a_i.

$$\alpha^i = \text{softmax}(S_{i,:}) \in \mathbb{R}^Q, \forall i \in \{1, \cdots, C\} \tag{2}$$

$$a_i = \sum_{j=1}^{Q} \alpha_j^i q_j \in \mathbb{R}^{2h}, \forall i \in \{1, \cdots, C\} \tag{3}$$

4.3.2 Q2C Attention

For the Q2C Attention, we take the maximum of the corresponding row of the similarity matrix, which is the maximum corresponding attention for the question attending on that context. After obtaining all the weight vector for the context, we perform a softmax over the resulting vector and use the softmax result to obtain a weighted sum of the context hidden states c_i.

$$m_i = \max_j S_{ij} \in \mathbb{R}, \forall i \in \{1, \cdots, C\} \tag{4}$$

$$\beta = \text{softmax}(m) \in \mathbb{R}^C \tag{5}$$

$$c' = \sum_{i=1}^{C} \beta_i c_i \in \mathbb{R}^{2h}, \tag{6}$$

4.3.3 Self-Attention

Third, for the Self-Attention, we perform matrix multiplication on the context representations with themselves, along with weight matrices. Then we perform softmax on the attention distribution for each context location and obtain a weighted average sum.

$$e_j^i = v^T \text{tanh}(W_1 v_j) + v^T \text{tanh}(W_2 v_i) \in \mathbb{R} \tag{7}$$

$$\alpha^i = \text{softmax}(e^i) \in \mathbb{R}^C \tag{8}$$

$$a' = \sum_{j=1}^{C} \alpha_j^i v_j \in \mathbb{R}^{2h}, \tag{9}$$

A modification is also made here. The original form is $e_j^i = v^T \text{tanh}(W_1 v_j + W_2 v_i)$ but the operation $W_1 v_j + W_2 v_i$ requires memory allocation of a tensor with shape $(\textit{batch size}, C, C, 2h)$. With the modification, it only requires $(\textit{batch size}, C, 2h)$.

At the end, we concatenate the context hidden state c_i, the C2Q attention output a_i, the Q2C attention output c' and Self-Attention output a' to get

$$b_i = [c_i; a_i; c_i \circ a_i; c_i \circ c'; a'] \in \mathbb{R}^{10h}, \forall i \in \{1, \cdots, C\} \tag{10}$$

4.4 Modeling Layer

After performing various types of attentions, we feed the attention distributions to the modeling layer, which consists of two layers of bidirectional GRU. Given the attention distributions on the context, this layer intends to capture the interactions among the context words. The output size of this modeling layer is h for each direction.

4.5 Output Layer

Our output layer is an adaptation of work by Wang & Jiang, 2016. Let $H \in \mathbb{R}^{h \times C}$ be the matrix of final blended representations such that column i of H is b'_i. We run a GRU network for two timesteps to get probability distributions over the context for the start location $\beta_s \in \mathbb{R}^C$ and the end location $\beta_e \in \mathbb{R}^C$. We condition the end distribution on the start distribution by giving β_s as part of the input to the second step of the output RNN. The probability distribution for $k \in \{s, e\}$ and $s = 1, e = 2$ is given as follows,

$$F_k = \text{tanh}(VH + \text{tile}(W^o h_{k-1} + b^o)), \tag{11}$$

$$\beta_k = \text{softmax}(v^T F_k + \text{tile}(c)), \tag{12}$$

where $V \in \mathbb{R}^{h \times h}$, $W^o \in \mathbb{R}^{h \times h}$, $b^o \in \mathbb{R}^h$, $c \in \mathbb{R}$, and $v \in \mathbb{R}^h$ are all parameters to train, and 'tile' simply tiles its input to match the dimensions required for the immediate surrounding computation. The vector h_k is the hidden state of the k-th step of the output RNN:

$$h_k = GRU(H\beta_k^T, h_{k-1}) \tag{13}$$

The first hidden state h_0 is also a trainable parameter.

4.6 Max Boundary Confidence Selection

We use a smart span selection mechanism we call Max Boundary Confidence Selection at test time. The method relies on fixing the answer boundary the model is most confident about and choosing the other based on the first. Given the probability distributions for the start and end locations β_s and β_e, and a desired maximum answer length A, we choose the answer span (s, e) as follows.

1. Fix the boundary condition with the highest confidence, $\max(\beta_k)$. Assume for ease of explanation that $\max(\beta_s) \geq \max(\beta_e)$, so $s = \arg\max \beta_s$.
2. Select $e = s + \arg\max \beta_e[s : s + A]$

The method is symmetric if $\max(\beta_s) < \max(\beta_e)$.

5 Experiments

We trained our model on the official SQuAD v1.1 training dataset. Based on token-count analysis, we set the question length to 25, the context length to 300, and the answer length to 15. The great majority of examples in the dev set are smaller than this, and we save computation with smaller dimensions. We use a dropout percentage of 20%, a hidden size of 250, and a batch size of 90. We use TensorFlow's Adam optimizer because it performed better than AdaGrad. Performance is measured by a word-level F1 score and an exact match score computed against ground-truth answers. These are the scores by which all models submitted to the SQuAD leaderboard are evaluated.

5.1 Results

The model we describe in this paper achieved F1 and EM scores of 74.18% and 62.69% on the dev set after training for 6k iterations.

Q: What is the other NHL team aside from the Anaheim Ducks to reside in Southern California?
C: Professional sports teams in Southern California include teams from the NFL (Los Angeles Rams, San Diego Chargers); NBA (Los Angeles Lakers, Los Angeles Clippers); MLB (Los Angeles Dodgers, Los Angeles Angels of Anaheim, San Diego Padres); NHL (**Los Angeles Kings**, Anaheim Ducks); and MLS (LA Galaxy).

Figure 2: Sample question, context, and model-generated answer in bold

5.2 Ablations

In order to examine the contribution of each component, we performed ablations on the dev set. The result is shown on Table 1. Word and character embeddings play an essential role in the model, but word embedding contributes more by providing the important semantics of the inputted words while character embedding acts as a complement to that to handle out-of-vocabulary words.

We also ablate our three different types of attention. To ablate C2Q Attention, we replace the attended question vectors with the average of the outputs of the question's contextual embedding layer. Similarly, to ablate Q2C Attention and Self-Attention, we replace the attended context vectors with the outputs of the context's contextual embedding layer. Out of the three types of attentions, C2Q Attention contributes more on the improvement of the F1. In order to ablate the model layer, we retrained our model without them and directly feed the output of the attention to the output layer. Since the model layer acts as an encoding mechanism to capture the interaction between words, its removal leads to a huge loss in F1; in fact, it plays the largest contribution on the F1 score. At the output layer, we can ablate smart span select by removing it, and we can ablate answer pointer by replacing it with the original fully connected layers. We can also observe that both provide positive contribution on the final F1 scores.

Table 1: Ablations

Model	Impact on F1
Word Embedding	(-) 12.56
Character Embedding	(-) 1.1
C2Q Attention	(-) 11.67
Q2C Attention	(-) 6.12
Self-Attention	(-) 4.9
Model Layer with GRUs	(-) 20.26
Smart Span Select	(-) 0.87
Answer Pointer	(-) 1.25

5.3 Analysis

5.3.1 Performance of Max Boundary Confidence Selection

Adding Max Boundary Confidence Selection at test time with $A = 15$ improved or model's F1 and EM score by about 1% from 73.19% to 74.18% and 61.90% to 62.69% respectively. We also tried the smart span selection presented in Chen et al., 2016, where (s, e) is found by maximizing the quantity $p(s) \times p(e)$ across all spans. Besides being more computationally expensive, this method increased our scores less to 73.75% F1 and 62.16% EM.

5.3.2 Additional Features

As part of our experimentation, we added two hand-selected features discussed in Chen at al., 2016:

1. Exact Match: is the token at context location c_i also in the question?
2. Aligned question embedding: similar to exact match, but uses an attention mechanism to add 'soft alignments' between similar but non-identical words.

When added to our baseline model, these features improved F1 and EM score by about 20% at almost no computational cost. Yet adding these features to our large model resulted in an F1 decrease of about 1%, and we detected overfitting earlier in the training processes. We suspect that exact features like these are redundant with a more complex model because a more complex model has enough representational power to deduce and exploit the features, and the model suffers from redundancy. Indeed, attention diagrams from our large model show that both exact match and aligned question embedding are already 'noticed' by the model. In Figure 3, attention is high at the places where a context token appears in the question exactly (bob, soda) and approximately (buy, buys, bought). Nonetheless, working with hand-crafted features may be a good way to quickly experiment with and motivate novel attention mechanisms.

5.3.3 Performance by Question Type

We break down the performance of our top model on a variety of question types in Table 2 [3]. We also present the change in score from the baseline to the model described in this report. We consider the following interrogatives: who, whose, whom, which, what, how, why, when, where. Some results are intuitive. The longer the ground-truth answer, the lower the performance, and 'why' questions are more difficult to get right than 'when' questions. However, some results are worth analyzing further. For example, questions that have two or more interrogatives or no interrogatives at all (e.g. 'the atomic number of the periodic table for oxygen?') have considerably worse performance than

[3]The first interrogative is 'Early', 'Mid', or 'Late' if it occurs at the first 25%, the middle 50%, or the last 25% of the question, respectively.

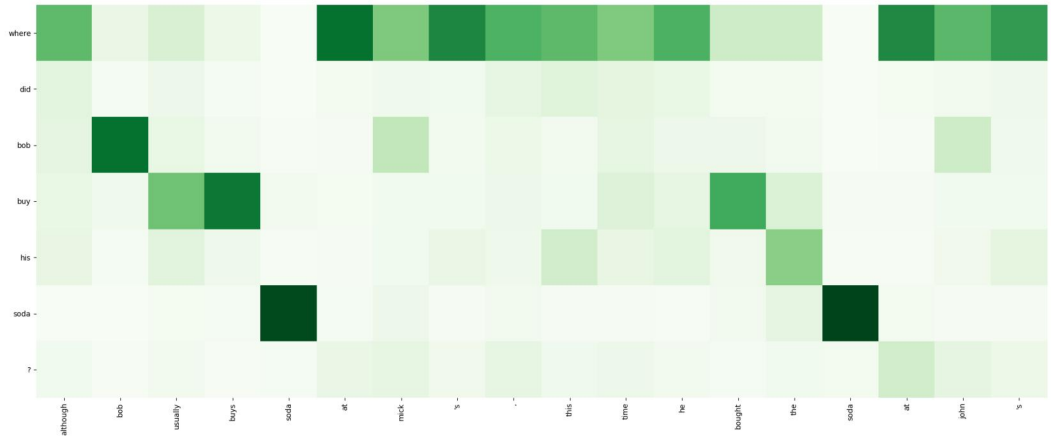

Figure 3: Attention diagram for an example question

questions with a single interrogative. Interestingly, extra short answers and 'how' questions, which make up a substantial part of the dev set, saw a markedly smaller increase in score from the baseline.

Questions in which the interrogative appears in the last 25% of the question achieve a substantially higher performance than a regular interrogative placement at the beginning of the question. It is also not obvious why 'whose', 'who', and 'how' questions are easier to answer. We suspect that analyzing particularly low-performing and high-performing question types more deeply can yield insights for new and better end-to-end methods, e.g., by learning what information the model has not learned to capture and building an attention mechanism inspired by that deficiency.

6 Conclusion

We presented our end-to-end neural network for machine reading comprehension on the SQuAD dataset. Our method builds off work done by high-performing models. We attempted a different method of smart span selection at test time, which we call Max Boundary Confidence Selection. In our dev dataset, this method worked better than the method that optimizes for the maximum values of $p(s) \times p(e)$. We analyzed how our model improves the baseline with respect to different question types, and we argued that hand-crafted features are ultimately redundant given more complex attention models.

Future work could involve carrying out experiments with features involving question types. Our analysis gives evidence that question types play an important role in what the model must understand to provide a good answer, since our model learns how to answer different question types at different rates. A future piece of work could provide a baseline model with hand-crafted features related to question types (what interrogative is in the question, where it occurs, the number of interrogatives in the question, etc.), see how the model improves each question type, tune the features, and then use the feature-tuning process to inform novel model components, i.e. a new attention mechanism.

Table 2: Performance by Question Type on the Dev Set

Question Type	EM %	F1 %	Δ EM	Δ F1	# Examples
Extra Short Ans (Single token)	68.29	74.81	(+) 23.39	(+) 25.87	4147
Short Ans (2-4 tokens)	63.44	76.00	(+) 31.56	(+) 33.64	5101
Medium Length Ans (5-9 tokens)	45.85	67.38	(+) 31.71	(+) 36.52	1132
Long Ans (10-19 tokens)	20.85	52.47	(+) 12.30	(+) 26.67	187
Extra Long Ans (20+ tokens)	0.00	26.19	(+) 0.00	(+) 19.09	3
No Interrogatives	44.14	61.55	(+) 20.72	(+) 31.15	111
One Interrogative	62.96	74.48	(+) 27.93	(+) 30.64	9933
Two or More Interrogatives	61.41	71.17	(+) 31.56	(+) 33.64	526
First Interrogative Early	62.89	74.38	(+) 27.93	(+) 30.65	9424
First Interrogative Mid	60.14	71.21	(+) 30.14	(+) 32.39	700
First Interrogative Late	68.36	78.93	(+) 28.96	(+) 31.62	335
First Word 'How'	63.58	75.82	(+) 21.47	(+) 25.13	1090
First Word 'What'	58.24	71.08	(+) 29.64	(+) 32.55	4753
First Word 'When'	80.17	86.05	(+) 24.86	(+) 25.96	696
First Word 'Where'	58.89	72.51	(+) 23.79	(+) 28.55	433
First Word 'Which'	62.56	73.13	(+) 28.41	(+) 29.07	454
First Word 'Who'	71.35	78.92	(+) 31.01	(+) 32.79	1061
First Word 'Whom'	-	-	-	-	0
First Word 'Whose'	79.41	86.96	(+) 26.47	(+) 30.72	34
First Word 'Why'	34.43	63.44	(+) 22.52	(+) 34.99	151
Other First Word	64.91	75.27	(+) 28.61	(+) 30.93	1898
Total	74.18	62.69	(+) 28.03	(+) 30.80	10570

References

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

[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 forquestion answering. *arXiv preprint arXiv:1611.01604*, 2016.

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

[5] 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.

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

[7] Jeffrey Pennington, Richard Socher, and Christopher D Manning. GloVe: Global vectors for word drepresentation. In *proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2014.

[8] Danqi Chen, Jason Bolton, and Christopher D. Manning. A thorough examination of the CNN/Daily Mail reading comprehension task. In *Proceedings of the Conference on Association for Computational Linguistics* , 2016.

[9] Yoon Kim. Convolutional neural networks for sentence classification. InEMNLP, 2014.