Reading Comprehension Neutral Network with Attention and Post Attention Modeling.

Xu Zhao
Department of Computer Science
Stanford University
pglory@stanford.edu

Zhi Liu
Department of Computer Science
Stanford University
willzliu@stanford.edu

Abstract

Reading Comprehension has been a hot and challenging topic in NLP research during recent years. Since Stanford NLP group released the SQuAD dataset in 2016, there has been tremendous progress in this field and many new techniques have been emerging. In this paper, we implemented a Bi-directional attention flow based model, and explored several state-of-art techniques shown in recent papers. The model eventually achieved 65.47% F1 score and 49.18% EM score on the Dev set.

1 Introduction

AI technologies have gained tremendous breakthrough in many fields in recent years, such as self-driving cars and language translation. AI researchers have also been exploring tackling the reading comprehension problem. Reading comprehension, the ability to read text and then answer questions about it, is a basic task for human beings and performed daily. However, it is challenging for machines, as it requires machine to understand not only the natural language within the context and question, but also the linkage between them and more broadly the real world implication. In order to facilitate the research process, Stanford NLP group released the Stanford Question Answering Dataset (SQuAD) in 2016 (1). Since then, there has been amazing progresses made by various groups in this field. In Jan 2018, AI created by Chinese tech giant Alibaba and Microsoft have tied for first place on the SQuAD, beating the human score for Exact Match (providing exact answers to questions) for the first time.

Among the new techniques in the reading comprehension field, one key group of breaking-through techniques is attention mechanisms. Attention mechanisms generally allow the system to focus on a target area within a context paragraph that is more relevant to the question (question-aware-context). The recently developed bi-directional attention (2) also obtained context-aware-question and avoided early-summarizing the context paragraph into a fixed-size vector. The model enjoyed significant improvement over previous attention models.

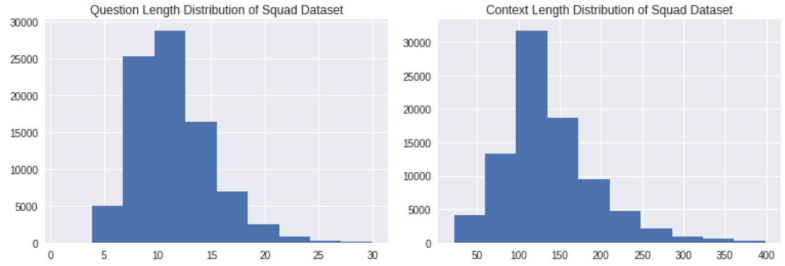
In this paper, we built a model based on the bi-directional attention, and also explored adding/analyzing various other techniques. The rest of the paper is organized as following: Section 2 analyzes the SQuAD dataset and discusses relevant parameters setting. Section 3 describes the model architecture. Section 4 presents the model's results and discussion. Section 5 concludes and discusses future work.

2 SQuAD Dataset Analysis

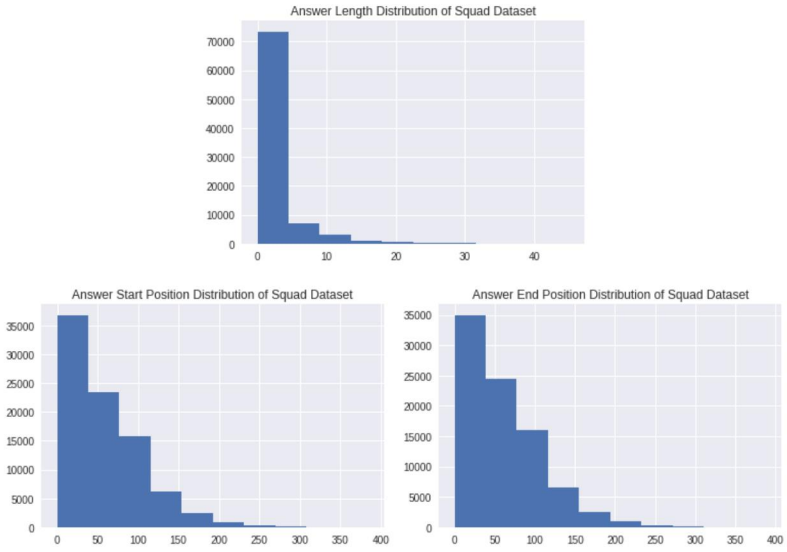
The SQuAD is a new reading comprehension dataset, consisting of questions posed by crowd workers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage. With 100,000+ question-answer pairs on 500+ articles, SQuAD is significantly larger than previous reading comprehension datasets.

We performed analysis on the statistics of the training data set of the project (86,084 question-answer pairs) to facilitate various parameters setting.

The histogram information shows that 100% of question length is below 30, 98.6% of context length is below 300, and 100% of context length is below 400. Accordingly, we set the maximum length of the question and context to be 30 and 400 respectively. We tried 300 for maximum length of context as well but there is no material impact on model performance.

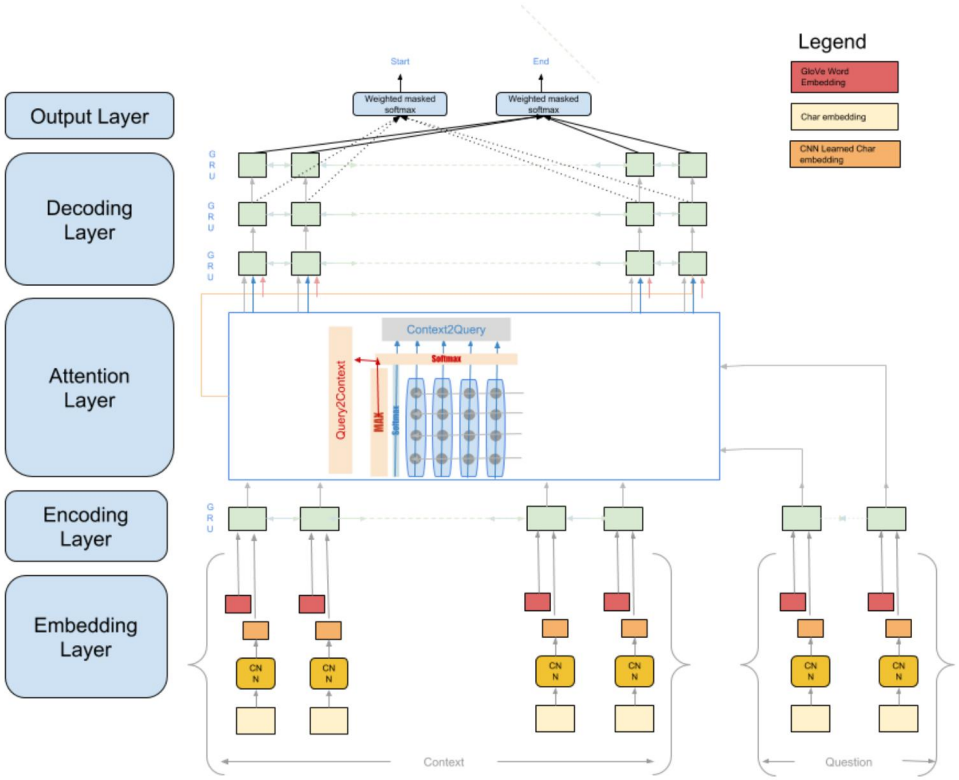


As for answer, we plotted histogram for the answer length, as well as starting position and ending position of the answer in context. 97.9% of answer the length is below 15. We set the maximum length of answer to be 15. Without limiting the answer length would deteriorate the model performance as there would be more false long answers although most answers are shorter than 15. The starting and ending position of the answer have similar histogram pattern. 99.9% of ending point of answer are within first 300 words in context. This confirms that setting maximum context length to be 300 would not improve the model performance much.



3 Model Architecture

Starting from the basic model framework of default project of CS224n, we enriched the model by adding multiple advanced new techniques, and tried to mutate the architecture using different subcomponents, derived from existing top performing squad models. The model is incrementally developed. Components that show meaningful improvements are kept. Additional training and fine-tuning are introduced for best performing models later. Among all architecture we have constructed, the best performance model contains relatively simple structure strongly resemble the BiDaF (2) architecture. The model contains Embedding layer with character embedding, an simple one level GRU encoding layer, a bi-directional attention layer, a three decoding layer and an output layer (Our best model turns out to omit the so called "High way" layer, and another key difference is we have utilized GRU as the default recurrent neural networks instead of the LTSM unit originally deployed by BiDaF).



3.1 Embedding Layer

Based on base model, we added character embedding. Character-level encoding offers additional information on the internal structure of words. For example, in case there are out of vocabulary word in the question and context, character embedding could provide word information in absence of GloVe vectors (3) . Character layer can be as easy as one hot, but that has similar shortcomings as one hot word vectors. In the BiDaF paper (2), and Google paper (3), character one hot vectors are fed into a CNN network to obtain embedding. We deployed similar CNN to embed word characters. Furthermore, we tested two approaches. One is to deploy a pre-trained embedding's from Google's one billing word projects. The existing embedding showed some aspect of reasonable relationship between characters. As tested by Morris using t-SNE (4), at certain dimension, not only are digits clustered closely together, they're basically arranged in order along a number line. In many cases, the uppercase and lowercase versions of a letter are very close.

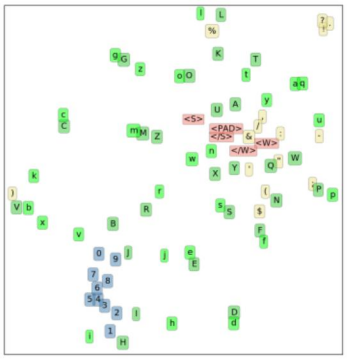


Fig. Pre-trained char embedding from one billion word project. (Extracted from published model check point). Note paddings are denoted as <PAD> --mapped to ASCI 4. <S>, </S> denotes start/end of sentence, <W>, </W> denote start/end of word. Unknown Unicode characters are mapped to ASCII 255. The figure is originally made by Morris. (3)

The second approach is to randomize the character embedding matrix and train it as other parameters. The two approaches generate similar results in the architecture we deployed.

The word embedding layer maps each word to a vector space using pre-trained GloVe vectors (3) as the baseline model provided by the project.

3.2 Encoding Layer

The encoding layer is to transform separate word information from embedding layer into a contextual mix encoding surrounding words to form a meaning representation of paragraphs. We tried six approaches on this layer: 1. Simple bi-directional GRU, weight-shared between question and context, obtaining $H \in \mathbb{R}^{2d \times N}$ for context and $= U \in \mathbb{R}^{2d \times M}$ by concatenating forward backward output (d is the hidden size). 2. Adding one "High way" network layer similar to the original BiDaF (2), before the encoding bidirectional GRU. 3. Adding two layers of "High way" network, before the encoding bidirectional GRU; 4. Adding one "High way" network layer after the encoding bi-directional GRU 5. Feeding question ending hidden state into the context encoding layer as the initial state (both forward and backward). 6. Concatenating the question and context matrices into one $\{\overrightarrow{q_1}, \overleftarrow{q_1}, \dots \dots \overrightarrow{q_N}, \overleftarrow{q_N}, \overrightarrow{C_1}, \overleftarrow{C_1}, \dots \dots \overrightarrow{C_M}, \overleftarrow{C_M}, \} = \text{biGRU}(\{y_1, \dots y_N, x_1, \dots . x_M\})$ and subsequently separating them before feeding into the attention layer. The best performing model turns out to be the simpler one, without any further processing before feeding into attention.

3.3 Attention Layer

We compared four attention architecture and their "mixings". Namely, a). The baseline model provided by the project (one way context to question). b). the Bi-attention flow structure in BiDAF. (2) c). Context level self-attention as proposed in R-Net (5) with basic attention. d).context level self-attention with bi-attention. It turns out the bi-attention layer performed best, and utilized a relatively low resources. Self-attention encountered large cross-context $M \times M \times 2d \times batch$ tenor, which forced us to reduce all other parameters including hidden size, batch size, and context size to avoid OOM error.

Briefly, the bi-attention layer first generated a similarity matrix using the following: (Note $\circ$ represent elementwise-multiplication. $S_{mn} = W^T[H_{:m}, U_{:j}; H_{:m} \circ U_{:n}]$ where $W^T \in \mathbb{R}^{6d}$, a trainable vector. A context to query attention is subsequently represented by attention vector$a_m = \text{softmax}(S_{m:})$. The attended query vector$\overline{U_{:n}} = \sum_n a_{mn} U_{:n}$; For query to question, we first calculated the weights on context word by $b = \text{softmax}(\max_{col}(S)) \in \mathbb{R}^N$. Where the attended context vector is$\bar{h} = \sum_n b_n H_{:n}$. $\bar{h}$ is then tiled N times. Finally, the contextual encoding and the attention vectors are combined together to get $[h; \bar{u}; h \circ \bar{u}; h \circ \bar{h}] \in \mathbb{R}^{8d}$ for each word. The final concatenated state is denoted as $\bar{\bar{H}}$.

3.4 Decoding Layer

Decoding layer is responsible for combining context-aware question and question-aware context to produce a new hidden state ready for finding the answer for question. We utilized two layers of bi-directional GRU, with output size same as previous layers default hidden size. The last output is denoted as D.

3.5 Output Layer

Output layer is responsible for finding the right positon for the answer. There are a few ways to do this, one is to predict the start and end position; alternatively, we can predict the span and a start (or an end). In the baseline model, the start and end positions are predicted independently. It is natural to assume the end position of the selected answer depends on start position. Therefore, we limited the accepted end position to be larger than start position. Based upon earlier examination of the statistics, we selected a max span of answer of 15. Besides, the start and end position is selected based on the maximum product of probability ${p_s}^{start} {p_e}^{end}$ among the acceptable spans. In addition, we also tried to add more layer of decoding for the end position, the output of this decoding layer is denoted as D'. Alternatively, the pointer Net structure encode start position prediction output when predicting end positions (6), so this is another structure we believed to be effective and tested. It turns out not robust in our test. The best performing model has the following prediction layer:

$$p^{start} = \text{softmax}\left({W_{start}}^T\left[\bar{\bar{H}}; D\right]\right)$$

$$D' = \text{biGRU}(\bar{\bar{H}}; D)$$

$$p^{end} = \text{softmax}\left(W_{end}{}^{T}\left[\bar{\bar{H}}; D'\right]\right)$$

$$(start, end) = \underset{s,e\,,e-s\leq 15, e>s}{\text{argmax}} \; p_s{}^{start} p_e{}^{end}$$

4 Results and Analyses

Consistent with squad challenge standard (1), F1 score and exact match (EM) score are used for evaluating the model. F1 score is the harmonic mean of precision and recall. We have achieved a F1 score of 65.47% on the dev set. Please note as current codelab have persistent issues, we haven't' submit on test board. The test F1 score will report separately. Notably, the second best model is with one more "High way" layer, while it performs similarly to the best model. This result is further discussed later in the initialization part.

4.1 Model Performance of Different Architectures

Based upon different trials with adding/removing of various components, it can be noticed that most important layers are the attention layer and modeling (after attention) layer. Interestingly, in our experiments, adding more preprocessing before feeding to the attention layer degraded performance. The complicated out layer is also important, as it avoids generating answers with end position ahead of start position.

Character embedding helped the performance to improve by a certain extent, around 2% improvement above base line and in the case of biattn. The exact match score increased a little bit, indicating certain word pattern recognition is at play. The effect is marginal though. Given the GloVe word vectors contains a total of 40k words with pre-trained knowledge, the out of vocabulary situation is rare. This is believed to be the reason why the improvement is limited.

Table1. Model Performance of Different Architectures

	Dev F1	Dev EM	Train F1	Train EM
Best Model (char+biattn+mod+out)	65.47%	49.18%	73.68%	59.60%
char+ 1 layer hiway +biattn + mod +out	65.46%	48.99%	73.25%	55.10%
Char + biattn	49.29%	36.68%	79.26%	68.80%
biattn(no dropout on attn.)	46.98%	34.64%	67.68%	56.40%
Self + basic attn (small parameters)	35.02%	25.97%	47.05%	36%
Char + baseline	41.70%	30.79%	58.15%	46%
self + biattn (small parameters)	35.38%	25.89%	41.58%	30.09%
baseline	40.34%	29.24%	63.17%	50.40%
char +xaivier 2 layer highway + biattn + mod	Early termination			
Char + biattn + pointer	Early termination			
Char + cqconcat + biattn + mod +out	Early Termination			
Char + hidden q2c + biattn +mod + out	Early termination			

** Explain of every sub component: "*hiway*": this is adopted from BiDaF paper "high way network". "*hiddenq2c*": The end of hidden state from the question encoding layer fed into context encoding layer as the initial state (both backward and forward). "*Char*": character embedding with 1 layer of CNN. "*Xaivier*": The gated matrix in highway network is usually initialized with identity matrix unless we use "xaivier" in the description above. "*Self*": refers to self-attention mechanism mentioned in the modeling section. Mod: refers to the modeling layer mentioned in the modeling section. "*Cqconcat*": context and questions are concatenated to feed RNN and separate later. "*Pointer*": refers to pointer mechanism adopted from (6). "*Early Termination*" refers to the fact it is pretty early in training that the loss reduction would not compete with the model it is built upon, such that it is not run into full rounds (begin to see overfitting signs) , but terminated early in training.

4.2 Model Performance under Different Hyper-parameters:

4.2.1 Dropout:

To fine tune the best model, we tried different dropout ratio. When ratio goes to only 0.01, the network shows apparent deterioration in performance, dropping form the high of ~65% to a peak of only 56%. When dropout s within range (0.15-/+ 0.05), the performance is similar.

4.2.2 Learning Rate:

As long as we adjust the Learning rate change within certain thresholds, (<0.01), we can see speed up in the initial loss reduction with a higher learning rate, but the end performance do not improve/degrade in a significantly way. Once learning rate is above the threshold, the model would fail to converge and loss would stuck.

4.2.3 Initialization:

We found different initialization impacts performance significantly. In the test of bi-attention, we used either constant initialization or Xavier initialization. The constant initialization version had a performance deficit of 14% with respect to the Xavier version. It is noted if we use the constant version, the model is actually performing worse than baseline.

In the test of highway network, we tested different initialization as well. Given the highway network is trying to compose a transformed version with original version of input, we tried to initialize the gated unit with either identity matrix (with negative bias) – so it will only function as direct pass-through at the beginning. Interestingly, this version performs better than Xavier version, this may be due to the fact a randomized transformation destroys original information in the context and question and makes the model failed to generalize knowledge and lose direction to learn.

4.2.4 Char embedding size:

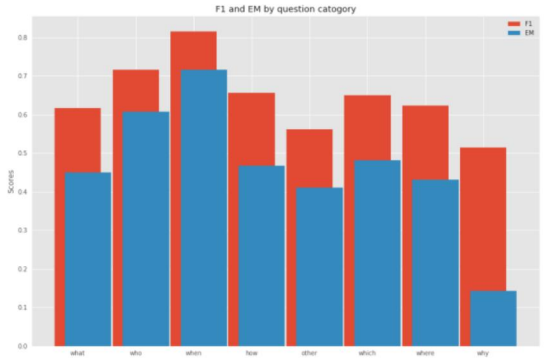
Given most English words are short (<15 characters), when we chose a large embedding size, there are many padding vectors exist, in our network we do not give "direct" instruction on how to deal with paddings and hope the model to learn itself. Such design with adding the char embedding size seems adding the total "difficulty" of the problem. Indeed, in our experiments, a larger embedding size (20) performed worse. Not only that, our pre-trained embedding size is 16, when we use a shorter but randomized embedding transformation matrix, the performance gets better.

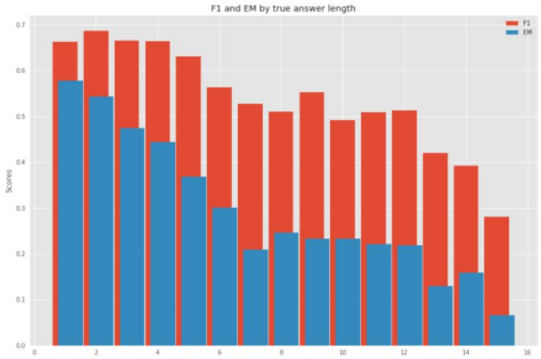
4.3 Error Analysis:

4.3.1 Model performance on question types

The model's performance on different types of questions are quite different. Understandably, for the questions that need more comprehension, like questions on "why", the performance is worst. On the other hand, "what", "when", "who" types usually only need to pick up direct clues from question and the model predicted them pretty well.

As noted in the below left figure, question on time ("when") are relatively easy for the model, with a F1 score on dev set above 80%. The next three are "who", "what" and "how" type of questions. Contrary to our intuition, the "how" type of questions are not that difficult to our model. On the other hand, the "why" questions are hardest, with an extreme low exact match score (11%). This may be due to the fact an answer to "why" questions tends to be long (even longer than 15) and less straightforward. On note is that there is an answer with 37 words and our model correctly picked the 15 words that is located in this 37 words answer.





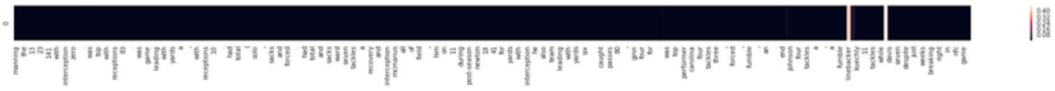
4.3.2 Model performance on true answer lengths:

From the right figure above, the model achieved higher performance with shorter true answers. Given we have limited our answer span, all true answers with a span longer than 15 will have an exact match score as zero, while we still archived a F1 of 27%. We do noticed the F1 score between questions with true answer span at 14,15,16,17 are not significantly different, indicating the trade-off limit we put on answer span did not significantly degrade long-answer span questions. The next step of improvement maybe should put more emphasis on reading comprehension rather than relaxing the answer span limit.

4.3.3 Detailed analysis on attention and start/end position picking mechanism.

Name picking (Right choice):

In a sample question "which player played even though he broke his arm two weeks prior to the super bowl?" Our model picked clue about **names**. When we look at the start position probability distribution, two names are highlighted and all others are not considered good candidate (shown below):

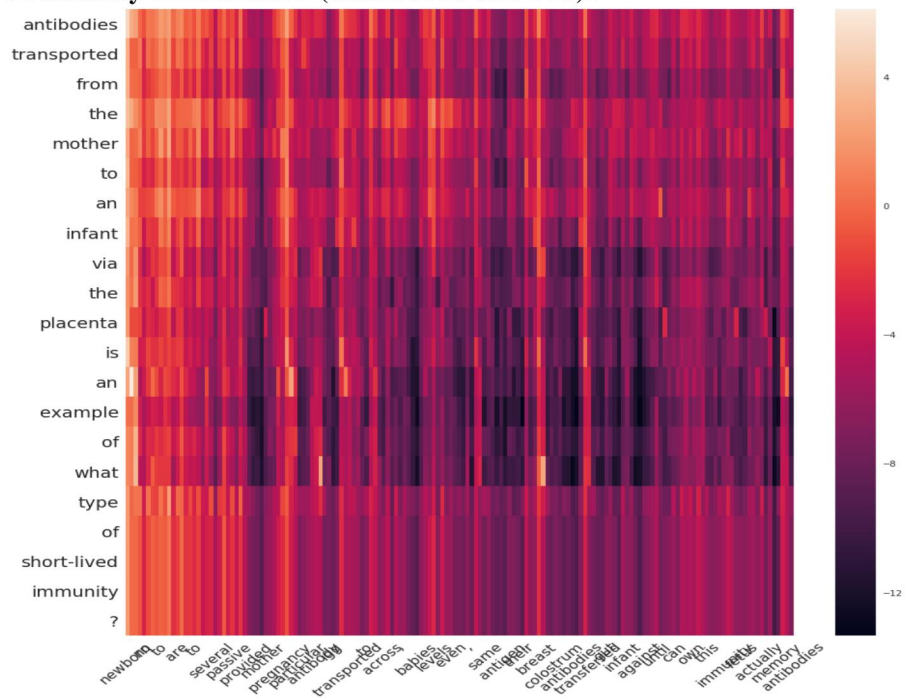


We also looked at the similarity matrix in bi-attention to check how the model highlights context words given a question word. For example, the word "**broke**" in question highlighted the context words (top 15 in terms of similarity matrix value) "**finished, game, the, breaking, yards, interception, caught, one, with, forced, two, a manning, just fumble**", which is believed to helped to anchor the name "Davis" instead of the other name "Luke Kuechly"

Lack of language transformation (Incorrect answer).

A question asked "what field of computer science analyzes the resource requirements of a specific algorithm isolated unto itself within a given problem." The answer lays in the sentence "a key distinction between **analysis of algorithms** and computational complexity theory is that **the former** is devoted to analyzing the amount of resources". The correct answer should be "analysis of algorithms" as that is what "the former" refers to. While our model picked up correct clue for answer, but failed transform "the former" to the original word it refers to. This is an interesting case as the model seems to get the basics of question and answer, but lack of language transformation capabilities. This reminds us that iterative reasoning method may help in the future generations of models, as the model may be able to learn "the former" type of answer need another round of transformation to its true underlying identity it refers to.

Proximity Preference (Incorrect answer).



QUESTION: antibodies transported from the mother to an infant via the placenta is an example of what type of short-lived immunity?

CONTEXT: newborn infants have no prior exposure to microbes and are particularly vulnerable to infection. Several layers of passive protection are provided by the mother. During pregnancy, a particular type of antibody, called **igg**, is transported from mother to baby directly across the placenta …… . This is **passive immunity** because the fetus does not actually make any memory cells or _antibodies—…… .

The true answer should be “passive immunity”, and our model picked “igg”. Note if we observe the heat map, the questions words highlighted the beginning part of the passage. (See chart above). The word “passive immunity” is introduced in the middle/end part of the passage. The model seems focused the search near the passage that resembles the questions -- indeed, if we only allow human to pick answer for the beginning of the paragraph, we may have to choose “igg” as well. This highlights the facts that the model tends to find answers correctly if they are closer to the question (or the context that resemble the question)

5 Future Work

In this paper, we report our result and implementation details of our reading comprehension model, which is mainly based upon the bi-directional attention flow model with variants derived from other state-of-art models. Experiments showed that attention and output modeling are giving performance gains while modeling before the attention layer get some hard-to-tune scenarios. It is worth further exploring before-attention multi-layer modeling, especially given state-of-the art results had implanted effective transformations. The current self-attention layer is quite resource-hungry, but we have devised a time-loop version of the layer which will cost significantly less memory resources and a full implementation would help to allow deeper abstracting context and produced better machine comprehension in general.

Bibliography

1. *SQuAD: 100,000+ Questions for Machine Comprehension of Text.* **Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, Percy Liang.** Vol. arXiv:1606.05250v3.

2. *Bidirectional Attention Flow for Machine Comprehension.* **Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, Hannaneh Hajishirzi.** 2017, ICLR, p. arXiv:1611.01603v5.

3. *One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling.* **Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, Tony Robinson.** Vol. arXiv:1312.3005v3.

4. *Dissecting Google's Billion Word Language Model Part 1: Character Embeddings.* **Colin, Morris.** Vols. https://colinmorris.github.io/blog/1b-words-char-embeddings.

5. *R-NET: Machine Reading Comprehension with self-matching networks.* **Natural Language Computing Group, Microsoft Research Asia.** Vols. https://www.microsoft.com/en-us/research/wp-content/uploads/2017/05/r-net.pdf.

6. *Machine Comprehension Using Match-LSTM and Answer Pointer.* **Shuohang Wang, Jing Jiang.** Vol. arXiv:1608.07905v2.