Self-Attention is All You Need

Stanford CS224N Default Project
March 2022

Joanne Zhou
Department of Statistics
Stanford University
`joannezhou@stanford.edu`

Brian Hill
Department of Statistics
Stanford University
`bwhill@stanford.edu`

Abstract

The internet contains the answer to most questions, the challenge mostly lies in finding it. In this report we build a model that approaches a narrower scope of the larger question-answering problem, namely pointing within a provided passage to the answer to a question using the SQuAD 2.0 dataset. We employ various self-attention mechanisms, including those from BiDAF, R-Net, and residual self-attention in order to reveal which types of self-attention work best for question-answering, and analyze why some types improve performance and others do not. Further, we employ a CNN to generate character-derived word embeddings and employ ensembling to improve prediction quality across various model types. With these techniques, we are able to improve on the BiDAF baseline significantly without overfitting to the training and dev splits.

1 Key Information to include

- Mentor: Vincent Li
- External Collaborators: None
- Sharing project: Not Applicable

2 Introduction

A greater ease of finding information when searching text (e.g. the internet) would have the potential to improve accessibility for the disabled, improve verbal question and answering (e.g. Siri), and constitute the beginning of using a language model as a knowledge base. One component of this answer retrieval network is to evaluate where in a passage of text the answer is contained, and we tackle this question with our model. The SQuAD Dataset [1] was designed specifically to train and evaluate question-answering models. We use the ability of our model to match the passage answers (F1 and EM scores) in order to compare various models.

We started with the Bi-Directional Attention Flow (BiDAF) [2] model and we experimented with adding various types of self-attention to the network. We first considered gated dot and additive self-attention taking inspiration from the R-Net model [3]. However, we observed poor results, and in conversation with our project mentor Vincent Li were pointed towards trying residual self attention following *Simple and Effective Multi-Paragraph Reading Comprehension* by Clark and Gardner [4]. This model drastically improved results. In addition, we found character-derived word embeddings and R-Net's answer pointer network to be helpful in improving performance.

3 Related Work

There have been many approaches to question-answering largely falling into two groups. First, there were RNN-based approaches that leveraged LSTMS to enhance the information flow across longer sentences and passages. These models use self-attention to pick out which words are most relevant from the question and passage. However, these have mostly fallen out of favor as transformers, which discard RNNs, and use position encoding and many self-attention blocks as well as residual connections and layer normalization, popularized by the paper *Attention is All You Need* [5]. We focus here on three RNN-based approaches that inspired our work, BiDAF (which served as our baseline model), R-Net and Residual Self Attention.

3.1 BiDAF

The BiDAF model [2] starts by encoding the word embeddings and runs through a bi-directional attention layer that both localizes relevant parts of the question as well as creates a question-responsive passage representation. This bi-directional attention flow was the main contribution to BiDAF over prior models, and produced promising results. The attention results are then run though a LSTM modeling layer and outputted into probabilities over the passage for start and end positions separately.

3.2 R-Net

The key contributions of this paper lie in R-Net's attention mechanism, which largely improved the model performance. First, instead of using a traditional attention-based recurrent networks, the authors introduced a gated attention mechanism. When computing the question-aware representation of the passage words, this the gated attention can mask out the unimportant passage words and add weight to the important ones.

Second, besides matching the question and the passage, the authors also proposed a self-matching attention mechanism that attends each passage word to the rest of the passage. The self-matching enables each answer candidate to consider information and clue given in the rest of the passage. This attention layer is also gated, and is applied on the question-aware passage representation, which further emphasized the words in the entire passage that is relevant to the question.

Last but not least, this model employs an answer pointer output layer, based off of Wang and Jiang [6]. This answer pointer layer uses question pooling from the pre-processing layer as well as the self-attention outputs, and puts them through an attention mechanism to generate start scores, which are used in conjunction with the question pooling again to generate the end scores. We go into more detail on answer pointer in the description of Model I.

3.3 Residual Self-Attention

In the study done by Clark et. al. [4], an adaption of the BiDAF model is built for the purpose of multi-paragraph reading comprehension. Their model structure incorporates both bi-directional attention and self-attention by applying the self-attention layer residually, so its output is additionally summed with the input. Their model structure achieves promising results on the SQuAD 1.1 dataset (Dev EM = 71.60, F1 = 80.78) but was not tested on SQuAD 2.0.

4 Approach

We detail below the key approaches taken by our model.

4.1 Character-Derived Word Embeddings

In addition to the pretrained GloVE word embeddings, we wanted to pick up on meaning in sub-words to handle unseen or composite words. Based on work by Kim [7], we implement a CNN that scans over the randomly initialized character embeddings for each word using hidden size number of filters to create a character-derived word embedding that is concatenated with the GloVE word embedding before passing into the encoding layer. We employ a window size of 4 to capture most prefixes (a hyperparameter search could be easily performed with other sizes in future work). In our experiments the character embedding had a large improvement on model performance, as it was effectively able to pick up on sub-word components.

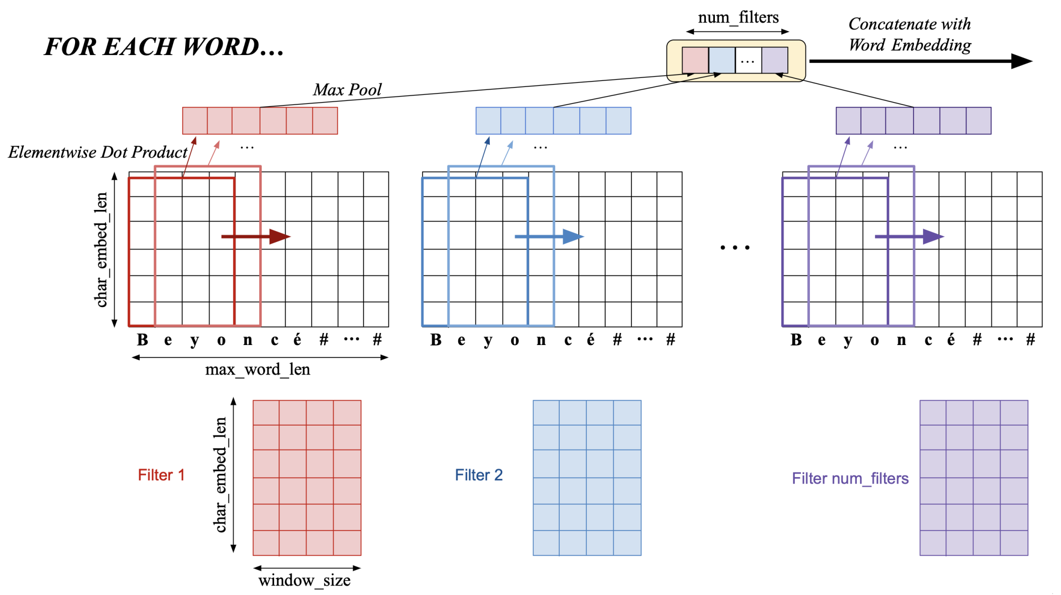

Figure 1: Self-made schematic depicting CNN applied to randomly initialized character embeddings.

4.2 Model I: BiDAF + Pointer Networks

We build our first model based on the baseline BiDAF model, then we add character embedding and change the output layer to be the pointer networks structure implemented by the R-Net paper [3].

The model structure is illustrated in Figure 2. First, the word level Glove embeddings and the character-derived word embeddings are concatenated and passed through a Bi-directional GRU layer (encoding layer). The context and the question share the same encoding layer weights. Then, we implement the bidirectional attention flow layer using the same method as the baseline BiDAF model. The output from this layer incorporates both context-to-question attention and question-to-context attention. A two-layer Bi-directional GRU is used as the modeling layer to transform the attention output.

In the pointer network, an attention mechanism is used to compute the start and end position probability distributions. Given context encoding v_j after the modeling layer, the start position probability a^{start} can be obtained by:

$$s_j^{start} = v^T tanh(W_v v_j + W_r r^Q) \qquad a_i^{start} = exp(s_i^{start}) / \sum_{j=1}^{c\_len} exp(s_j^{start})$$

where r^Q, the initial state of the pointer networks, is an attention-pooling vector of the question encodings u_j.

$$s_j = v^T tanh(W_u u_j + W_V V^Q) \qquad a_i = exp(s_i) / \sum_{j=1}^{q\_len} exp(s_j)$$

$$r^Q = \sum_{i=1}^{q\_len} a_i u_i$$

Then, we set up the second part of the pointer network to get the end position probability. To obtain the second initial state, we attention-pool the context vectors and pass it through a GRU cell with hidden state r^Q.

$$c^{start} = \sum_{i=1}^{c\_len} a_i^{start} v_j \qquad h^{start} = GRUcell(c^{start}, r^Q)$$

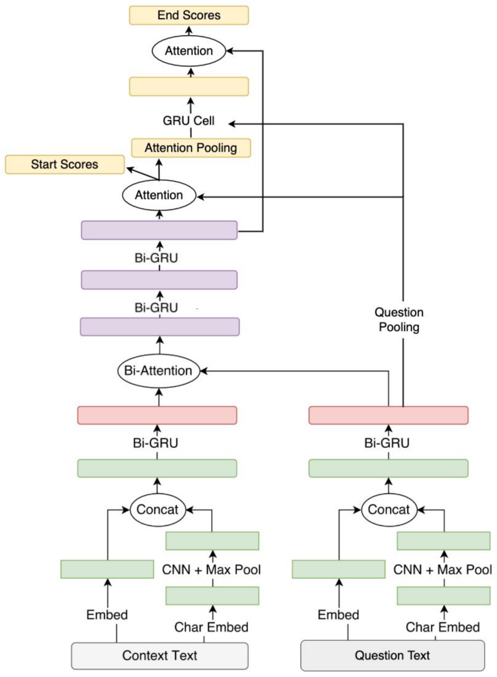

Figure 2: Model I architecture.

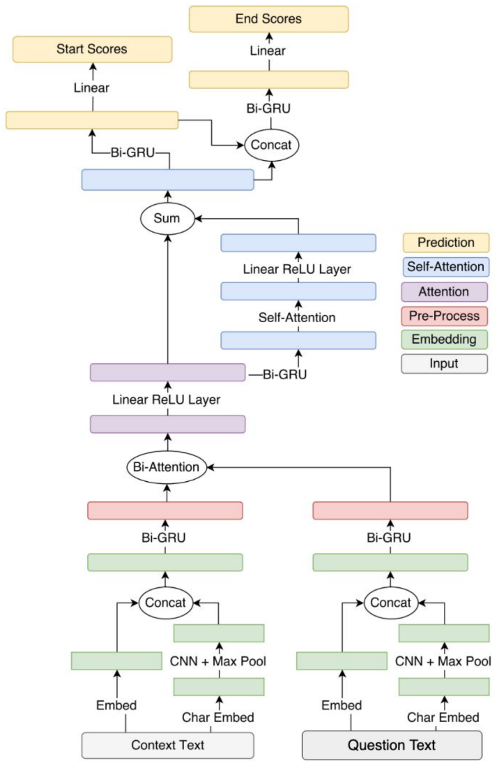

Figure 3: Model II architecture.

Lastly, we can obtain end position probability a^{end} using the same attention mechanism as before.

$$s_j^{end} = v^T tanh(W_v v_j + W_h h^{start}) \qquad a_i^{end} = exp(s_i^{end}) / \sum_{j=1}^{c\_len} exp(s_j^{end})$$

4.3 Model II: BiDAF + Residual Self-Attention

Inspired the model structure proposed by Clark el. al. [4], we implement a passage-to-passage self-attention layer. The model architecture can be found in Figure 3. Different from the self-attention in R-Net, which transforms the output from the context-to-question attention layer before the modeling layer, this implementation adds the self-attention output to the BiDAF attention output residually. We attempted implementing both forms of self-attention and found that the residual self-attention yields the best results.

Model II has the same embedding, encoding, and BiDAF attention layers as Model I. Instead of passing through an RNN layer, the attention output goes through a linear transformation and a ReLU function. Self-attention output a_i is computed as follows:

$$S_{ij} = w_{sim}^T [c_i; c_j; c_i \odot c_j] \qquad \bar{S}_{i,:} = softmax(S_{i,:})$$

$$a_i = \sum_{j=1}^{N} \bar{S}_{ij} c_j$$

where c is passage representation after a Bi-GRU layer. Diagonal of the similarity matrix is set to $-inf$ before the softmax step. The self-attention output is passed through another linear ReLU layer and added residually to the first linear ReLU layer output.

In the output layer, a Bi-GRU is applied to the summed attention outputs, and a linear layer is applied to obtain the start position probability. The hidden state of the Bi-GRU is concatenated with the summed attention, and another Bi-GRU and linear layer are applied to compute the end position probability.

4.4 Ensembling

Because model II yields better performance, we trained model I with two different random initialization, and trained model II with three different initialization. When predicting on a test or dev split, we average the output probabilities across the combination of models. Different combinations of the models are tested.

5 Experiments

In this section we cover the details of our model and the results from various iterations of it.

5.1 Data

For this task we employ the SQuAD 2.0 dataset, as described in [1]. In contrast to the SQuAD 1.0 dataset, this dataset also provides unanswerable questions. Since the course does not have access to the actual test set targets, we break down the provided training data into a training, dev, and "test" set. As students, we employ the training and dev sets in training our model and use the "test" set for checking our model on truly unseen data. The dataset is best described via an example with Beyoncé.

Passage: Beyoncé Giselle Knowles-Carter (born September 4, 1981) is an American singer, songwriter, record producer and actress. Born and raised in Houston, Texas, she performed in various singing and dancing competitions as a child, and rose to fame in the late 1990s as lead singer of R&B girl-group Destiny's Child. Managed by her father, Mathew Knowles, the group became one of the world's best-selling girl groups of all time. Their hiatus saw the release of Beyoncé's debut album, Dangerously in Love (2003), which established her as a solo artist worldwide, earned five Grammy Awards and featured the Billboard Hot 100 number-one singles "Crazy in Love" and "Baby Boy".
Question: In what city and state did Beyoncé grow up?
Answer: Houston, TX
Question: What is Beyoncé's most recent album?
Answer: N/A (Passage cannot answer question)

5.2 Evaluation method

We employ the EM (Exact Match) and F1 metrics, as is standard for SQuAD. EM is reported as the percent of questions who's answer exactly matched at least one of the three human responses. F1 applies a more generous "partial credit", counting answers that partially overlap with at least one of the three human responses as well.

5.3 Experimental details

5.3.1 Hyperparameters:

We apply dropout with a probability of 0.2 to all the embedding, GRU, and attention layers. The hidden vector length is set to 100. The models are trained with 35 epochs and batch size of 64. The optimizer used is Adadelta with an initial learning rate of 0.5. The objective loss function is the sum of the negative log-likelihood (cross-entropy) loss.

5.3.2 LSTM vs. GRU:

Both performed similarly in terms of performance, but GRU was slightly faster so we used it in all our RNN layers.

5.3.3 Self-Attention:

The model performance was highly sensitive to the type of self-attention used. We implemented R-Net's [3] gated self-attention in both additive and dot product forms, but observed minimal change from the baseline model results, leading us to believe the first bi-directional attention was sufficient. With the advice of our mentor Vincent Li, we then implemented residual self-attention modeled in [4], with great success. Our main takeaways from this experimentation are that:

- An additional layer of self-attention after the BiDAF attention may not confer additional value, as the context-question alignment has already been performed.
- A larger hyperparameter search may be required to find optimal hyperparameters specific to the added self-attention layer.

5.4 Results

In order to decide how to ensemble our model, we compared the results of different combinations of Model I and Model II. (The earlier gated self-attention models we built performed poorly, as did a model combining both answer pointer and residual self-attention.)

Dev Set Results

Model	**EM**	**F1**
Baseline BiDAF	58.965	62.281
(I) Character Embeddings + Answer Pointer Network	61.586	65.143
(II) Character Embeddings + Residual Self Attention	62.729	66.201
Ensemble: 2 (I) & 3 (II)	**65.233**	68.409
Ensemble: 0 (I) & 3 (II)	65.115	**68.514**
Ensemble: 1 (I) & 3 (II)	65.199	68.489

We decided to ensemble Model I and Model II to improve performance, and chose the model with the best combined EM performance since our training script chose the best model by F1 performance, and we wanted to consider both metrics. We further present the test board results.

Test Set Results

Model	**Ensemble: 2 (I) & 3 (II)**
EM	65.934
F1	68.909

We spent most of the project adding intricate self-attention layers following parts of the R-Net paper and experienced slight drops in performance, so when we implemented character embedding, residual self-attention, and answer pointer, we were very excited to see the model improving. We were particularly impressed by the residual self-attention's leap in performance despite the remarkably similar architecture. This highlights the importance of the residuals in repeated attention schemes, to retain the ability to carry information from one layer to the following layer.

It is also impressive to note that many of the QA-Net transformer implementations perform only about 5 points better than our model with only two attention-like methods. We hypothesize that this is because there are diminishing marginal returns to additional attention blocks, or that perfecting the architectural nuances of one attention layer can yield results comparable to many transformer blocks.

6 Analysis

The type of self-attention used matters greatly. Our results show that naïvely adding complexity to the model does not result in improved performance. Instead, the complexity must alleviate a bottleneck in information flow through the model, or provide a chance to encode additional information separately from the other layers/blocks. Our main achievement was the identification of successful self-attention layers with the help of our project mentors. We found great improvement by adding a self-attention block, but it had to be a self-attention block that was capable of picking up on information separate from the prior bidirectional attention flow. R-Net's self-attention failed to account for more info, and it was only upon adding residual connections or performing an attention-like mechanism in the answer pointer that we were able to observe a high degree of improvement – that our self-attention layers added complexity *in the right place* to improve performance. Even combining residual self-attention and answer pointer did not yield a performance improvement over one or the other, indicating that they fill similar voids. Further, we are proud to have successfully implemented character-derived word embedding using CNNs and a way to assess optimal CNN hyperparameter tuning.

We further noticed interesting patterns in some of our model's predictions, which suggest a direction for future work. For example, one pattern we noticed was a failure to handle negations of words properly, signaling the additional challenge created by augmenting SQuAD 1.0 with unanswerable questions.

Question: When did violence end in war?
Passage: The war was fought primarily along the frontiers between New France and the British colonies, from Virginia in the South to Nova Scotia in the North. It began with a dispute over control of the confluence of the Allegheny and Monongahela rivers, called the Forks of the Ohio, and the site of the French Fort Duquesne and present-day Pittsburgh, Pennsylvania. The dispute erupted into violence in the Battle of Jumonville Glen in May 1754, during which Virginia militiamen under the command of 22-year-old George Washington ambushed a French patrol.
Answer: N/A
Prediction: May 1754

Here, we see that our model has confused opposites, lacking the understanding that "erupting into violence" is very different from violence ending. Our model simply sees that there is violence and a date and figures that is when violence ends. This is a pattern observed in many models, even BERT struggles to avoid the word-association trap when specific sentences are added to trick the model.

Future work investigating this phenomenon would be poised to have a major impact on the field. It is possible that solving this problem would also solve others. For example, we would expect our model to get confused by a question like "When did the war begin ending?", potentially reporting the war's beginning date as opposed to N/A or the wars ending.

7 Conclusion

We were impressed by our model's improvement over the baseline, especially the residual self-attention mechanism as we discussed in the analysis section. We were also proud to learn more about the preprocessing datastructures when implementing character-derived word embeddings and looking into POS tagging. With more time, there are a few additional techniques we would want to look into. While we use GloVE word embeddings and a character-derived word embedding CNN, additional feature augmentation such as part of speech tagging and coreference could lead to improvement if they capture additional useful information. Further, we would want to explore sub-word embeddings as a comparison to character-derived word embeddings more time we would have done. Additionally, the next logical model architecture to implement would be transformers such as QA-Net which the literature suggests offer improvement.

All in all, we are grateful to have implemented a successful model and explored character embeddings, answer pointer network, ensembing, and three variants of self-attention. If you know which self-attention to employ, it may be all you need.

References

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

[2] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hajishirzi Hananneh. Bi-directional attention flow for machine comprehension. In *ICLR 2017 Confrence Paper*, 2017.

[3] Natural Language Computing Group. R-net: Machine reading comprehension with self-matching networks. In *Association for Computational Linguistics (ACL)*, 2018.

[4] Christopher Clark and Matt Gardner. Simple and effective multi-paragraph reading comprehension. *CoRR*, abs/1710.10723, 2017.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[6] Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer, 2016.

[7] Yoon Kim, Yacine Jernite, David A. Sontag, and Alexander M. Rush. Character-aware neural language models. *CoRR*, abs/1508.06615, 2015.