Improving Bi-Directional Attention Flow for Machine Comprehension

Bosen Ding
Stanford University
`bosend@stanford.edu`

Yue Wang
Stanford University
`wyue0125@stanford.edu`

Abstract

Machine comprehension (MC) requires a complex model to capture the interaction between a context paragraph and a query to answer the question correctly. Traditional fixed-size vector attention mechanism couples attention between adjacent steps and usually attends in just one direction. As a hierarchical multi-process framework, Bi-Directional Attention Flow (BiDAF) system [9] provides a novel way to obtain query-aware context representation without early summarization. In this project, using the starter code as our baseline, we extend the BiDAF with character-level embedding, add a complex residual self-attention layer to our model and replace all LSTM with GRU. Our single model obtains EM 64.24 F1 67.71 on SQuAD 2.0 dev set. With ensemble method, we achieve EM 67.18 and F1 70.07 on dev board and EM 66.09 F1 68.96 on non-PCE test board. By analyzing questions by type, we show that our model improves most significantly on who type questions and on unanswerable questions. We also find that randomly initialized character-level embedding performs comparably well with trained character-level embedding while the training with randomly initialized embedding is much faster.

1 Introduction

One of the key challenges in Question answering (QA) is to achieve a meaningful interaction between the context and the query. Before the paper Bi-Directional Attention Flow For Machine Comprehension by Seo et al. published at ICLR 2017, attention mechanism has mostly been restricted as uni-directional. Previous works mainly employ a dynamic attention mechanism[1] introduced by Bahdanau, Cho, and Bengio, in which the decoder vector attends to all encoder vectors to obtain a focused weighted vector. In the area of machine translation, the query vector attends to each context hidden state to attain the weighted context vector. As noted by the authors of BiDAF, the traditional attention mechanism has its limitations of uni-directional query-to-context attending. The next attention depends on previous Recurrent Neural Network(RNN) output, implicitly depending on the previous attention result. Therefore, the traditional method might potentially reinforce the wrong attending direction. Instead of summarizing the context as a fixed-size vector, BiDAF computes attention for every time step. The attended vector for each step is free to flow through to modeling layer along with the previous layer output. The author claims that this new framework overcomes the early summarization information loss and imposes a memory-less attention mechanism. By decoupling the attention layer and modeling layer, BiDAF forces the separation of the task "learning the attention between query and context" and the task "learning the interaction within the query-aware context representation". It also overcomes the dependency limitation because the attention input at each time step is not dependent on the attention of previous time steps.

2 Related work

Stanford Question Answering Dataset (SQuAD) [8] is a large reading comprehension dataset consisting of 100,000+ questions. Since the release of this dataset, many complex neural models are invented to tackle the task of machine comprehension, and some of them have achieved human-level accuracy. However, because all questions in SQuAD1.1 are answerable, models can perform on SQuAD1.1 well by simply learning context and type-matching heuristics[13] and the good performance does not ensure the robustness of the system to distracting contexts[5]. SQuAD2.0[7] introduced last year has addressed this problem by adding 50,000+ unanswerable questions to the SQuAD1.1 dataset. The SQuAD 2.0 dataset challenges neural network models to not only answer questions correctly when possible but also determine if an answer exists at all[7].
Bi-Directional Attention Flow For Machine Comprehension by Seo et al. provides a multi-step hierarchical process that presents context at both word and character levels and uses a bi-directional attending mechanism to generate query-aware context embedding. We mainly base our model on this paper and improves its attention mechanism to exploit the potential of its bi-directional context embedding further. Transformer [11], an attention-only framework is the very first model that get rids of the sequential recurrent structure. Its extensive use of residual self-attention block leads us to the idea of self-attention. Similar to BiDAF++ [3], we add a layer of residual self-attention, which helps better capturing the embedding for each word in its context. We later show that this significantly improves the performance on the unanswerable question set.

3 Approach

As a hierarchical multi-stage framework, our BiDAF variant consists of six functional layers. Our major difference with the baseline model is the addition of character-level embedding, the replacement of LSTM with GRU, and the addition of a complex residual self-attention layer.

1. **Embedding Layer** consists of parallel character embedding layer and word embedding layer. The character embedding layer is responsible for mapping each word to a high-dimensional space on the character level using learned Convolutional Neural Networks (CNN). The word embedding layer maps each word to another high-dimensional space with pre-trained Glove word vectors. The original BiDAF model directly concatenates the Glove word embedding with CNN output and pass them to encoding layer as input. We started with this approach first and then added the projection layer between word-embedding and the encoding layer. We have experimented with both randomly initialized character embedding that comes with the starter code as well as trained character-level embedding. In the presence of the projection layer, we project the Glove embedding to d-dimensional(hidden-size) vectors and concatenate with the d-dimensional output of CNN. The two embeddings are concatenated and passed into a two-layer Highway Network[10], whose output is the same size as the concatenated embedding input. The experiment result and our findings will be presented in section 4.4.
2. **Encoding Layer** consists of a single-layer bi-directional Gated Recurrent Unit(GRU) [2] to exploit the temporal connections between words. We concatenate the forward and backward output of the GRU to obtain $\mathbf{H} \in \mathbb{R}^{2d \times T}$ from context embedding and $\mathbf{U} \in \mathbb{R}^{2d \times J}$ from query embedding from the previous layer.
3. **Bi-Directional Attention Flow Layer** is responsible for the attention between the context and query vectors.

   Similarity matrix $\mathbf{S} \in \mathbb{R}^{T \times J}$ is first computed where $\mathbf{S}_{ij}$ represents the similarity between i-th context word and j-th query word. Both context-to-query (C2Q) attention and query-to-context (Q2C) attention will be computed using this similarity matrix.

$$\alpha(\mathbf{h}_i, \mathbf{u}_j) = \mathbf{w}_{\mathbf{S}}^T[\mathbf{h}_i; \mathbf{u}_j; \mathbf{h}_i \circ \mathbf{u}_j] \tag{1}$$
$$\mathbf{S}_{ij} = \alpha(\mathbf{h}_i, \mathbf{u}_j) \tag{2}$$

   where $\mathbf{h}_i$ is the output of the embedding layer for i-th context word and $\mathbf{u}_j$ is the embedding layer output for j-th query word. and $\mathbf{w}_{\mathbf{S}} \in \mathbb{R}^{6d}$ is a learnt weight vector.

   Context-to-query Attention is used to highlight query words that are most relevant to each context word. It takes the row-wise softmax of similarity matrix $\mathbf{S}$ to yield the weighted-sum of the query vectors for each context word. For i-th context word,

$$\alpha_i = softmax(\mathbf{S}_{i,:}) \tag{3}$$

$$\tilde{\mathbf{u}}_i = \sum_j \alpha_{ij}\mathbf{u}_j \tag{4}$$

Query-to-context Attention attends to the most similar context word to each query word and is significant for the performance of the framework because all meaningful answers are just spans of context words.

$$\mathbf{r}_i = \max_j \mathbb{S}_{ij} \tag{5}$$

$$\beta = softmax(R) \tag{6}$$

$$\tilde{\mathbf{h}} = \sum_i \beta_i \mathbf{h}_i \tag{7}$$

Then for each context word, its context encoding and its attention vectors are combined to yield G, whose i-th column is $[\mathbf{h}_i; \tilde{\mathbf{u}}_i; \mathbf{h}_i \circ \tilde{\mathbf{u}}_i; \mathbf{h}_i \circ \tilde{\mathbf{h}}]$.

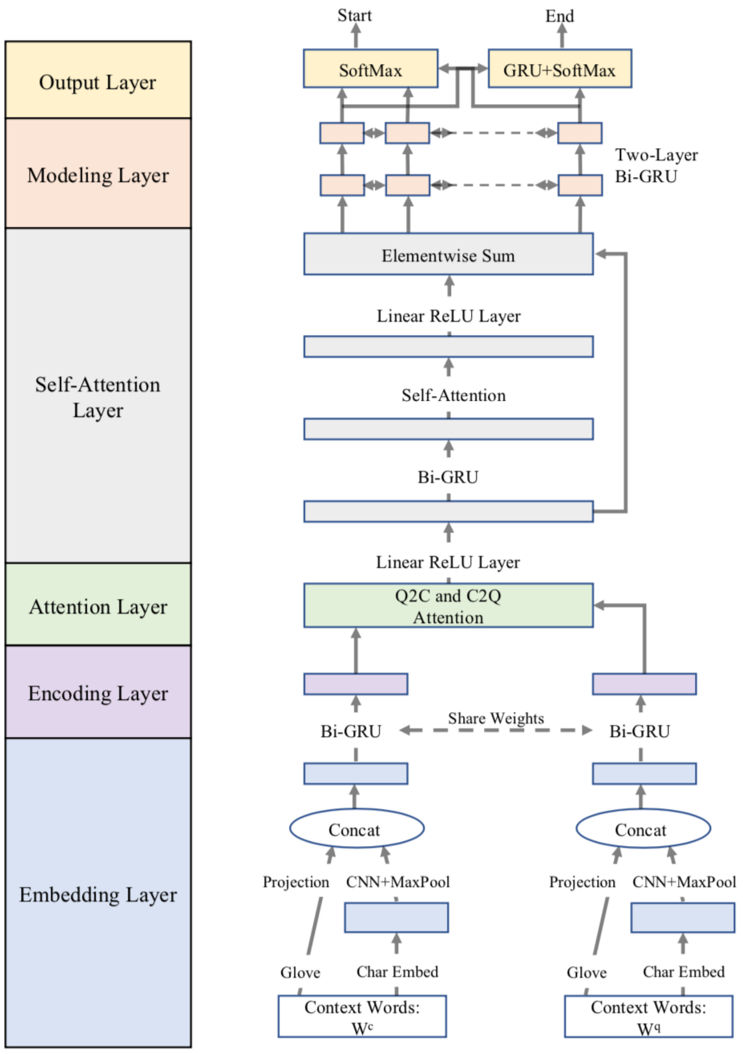


Figure 1: Residual self-attention BIDAF model diagram

4. **Self-Attention Layer** employees a modified residual self-attention structure. The input G is first projected to $X \in \mathbb{R}^{2d \times T}$ by passing through a linear layer with ReLU activation. We feed X through a one-layer bi-directional GRU and compute the self-attention of GRU output for each position of the context and pass them through another linear layer with ReLU activation to get $Y \in \mathbb{R}^{2d \times T}$. Finally, we add X and Y elementwise as the output of this residual self-attention layer.
5. **Modeling Layer** consists of two-layer bi-directional GRU, which takes the sequence of query-aware context representation and outputs concatenated forward-and-backward outputs of the GRU. The output is annotated as $\mathbf{M} \in \mathbb{R}^{2d \times T}$ where i-th column represents the representation of i-th context word based on the whole contextual information as well as the query information.
6. **Output Layer** computes the start $\mathbf{p}_1$ and the end $\mathbf{p}_1$ positions of the contextual span that the model deems as the answer to the query. Another bi-directional GRU is used to take $\mathbf{M}$ as input and outputs $\mathbf{M^2}$. Note that we differ from the original approach by only using the model layer output.

$$\mathbf{p}_1 = softmax(\mathbf{w}_{p_1}[\mathbf{M}]) \tag{8}$$

$$\mathbf{p}_2 = softmax(\mathbf{w}_{p_2}[\mathbf{M^2}]) \tag{9}$$

4 Experiments

4.1 Dataset

In this project, we use SQuAD 2.0 to both train and evaluate our models. SQuAD 2.0 extends SQuAD 1.0 with over 50,000 adversarial unanswerable questions. In our project, we will use a modified SQuAD 2.0 given by our course staff, which contains 129,941 training examples, 6,072 dev examples, and 5,921 test examples.

4.2 Evaluation method

Since we use the provided SQuAD dataset, our main evaluation metrics are Exact Match(EM) and F1 scores. We use the provided word-embedding-only BiDAF model as our baseline model and compares our extended model to this baseline model.

4.3 Experimental details

The main model architecture is shown in Figure 1. Each character-level embedding is of size 32. We use hidden state size 100 and 100 1D filters for the CNN character embedding, each with a kernel size of 5. We use AdaDelta optimizer[14] with a batch size of 64 and initial learning rate 0.5. During training, we use a dropout rate of 0.2 on embedding, all GRU layers as well as the linear layer before softmax output. Each model is trained for 30 epochs and the best model saved is evaluated every 50,000 steps by F1 score. We run our models on a local machine with 1080Ti GPU as well as Google Cloud VM instance with V100 GPUs. It takes us 5 hours to train the baseline model. For our extended models, it takes us 8-10 hours on 1080Ti or V100 to train one model. We have trained three baseline models and 20+ BiDAF variant models of different structures with various settings.

4.4 Results

Due to the limited attempts allowed on test dataset, all results without explicitly pointing out test dataset are obtained from dev dataset. With ensemble method, our final model achieves EM 67.18 and F1 70.07 on dev board and EM 66.09 F1 68.96 on non-PCE test board.
The original paper has no projection layer between pre-trained Glove embedding and encoding layer. When we follow this approach, we get EM 56.47 and F1 60.14 shown as trained-char-embed-BiDAF-w/o-proj in Table 1. When we switch to randomly-initialized-char-emebd-BiDAF-w/o-proj, we get roughly the same score. Then, when we added the projection layer, which projects Glove embedding of dimension 300 to 100, matching the CNN as well as hidden state size, we get 2 points improvement on both EM and F1. When we use the randomly initialized char-embedding with projection layer, we get 2 points improvement as well. In both cases, the model with randomly

Table 1: Result on SQuAD 2.0 dev set

Model	EM	F1
baseline	57.87	61.23
trained-char-embed-BiDAF-w/o-proj	56.47	60.14
rand-char-embed-BiDAF-w/-proj	56.56	60.17
trained-char-embed-BiDAF-w/-proj	59.87	63.17
rand-char-embed-BiDAF-w/-proj	60.28	63.32
residual-self-attention-char-embed-BiDAF	**64.24**	**67.27**
baseline ensemble (3 models)	60.71	63.76
residual-self-attention-char-embed-BiDAF-ensemble(3)	66.14	69.00
residual-self-attention-char-embed-BiDAF-ensemble(7)	**67.18**	**70.07**

Table 2: Result on SQuAD 2.0 dev set by answerability

Model	EM	F1	A-EM	NA-EM	A-F1	NA-F1
baseline	57.87	61.23	59.66	55.82	66.90	55.82
char-embed-BiDAF	59.87	63.17	61.62	58.58	**68.68**	58.58
our final model	**64.24**	**67.27**	**61.38**	**66.87**	67.71	**66.87**

initialized embedding performs as well as the model with trained embedding. The training time of the model with randomly initialized embedding is just half of that of the model with trained embedding. The randomly initialized vector for character embedding is of size 64 while the trained embedding is of size 8. One possible explanation is that randomly initialized high-dimensional vector has very low cosine similarity. The CNN layer is thus able to learn to react to each randomly initialized vector accordingly. Since in Linguistics, each character has almost no correlation in word meaning, a randomly initialized high-dimensional character-level embedding (64 in this case) could compensate the training-gain of low-dimensional embedding (8 in our case), and thus might be good enough for machine comprehension task.
The ensemble of 3 baseline models brings 2.5 points improvement. We also experiment with different ensemble method. The ensemble result shown below is acquired by max-pooling the probability for each word position across models, which improves the common average method by approximately 0.3-0.6 points. We find that the larger the size of the ensemble, the more improvement the max operation seems to have over the mean operation. We suspect that it is because max operation decides by the most certain choice from the ensemble of models while mean operation reverts to the average choice.

5 Analysis

5.1 Quantitative error analysis

We perform error analysis by answerability of questions, which is the major difference of SQuAD2.0 with SQuAD1.1. There are 2910 answerable questions and 3041 unanswerable questions in our evaluated dev set. As shown in Table 2, we can see that the baseline model performs poorly on the unanswerable questions. The introduction of char-level embedding improves both question types approximately equally. The addition of residual self-attention layer improves the unanswerable performance significantly, while only drags down slightly the performance of answerable questions, leading the substantial improvement in its performance across entire dev set, demonstrating the usefulness of our residual self-attention layer. One possible explanation for the slightly worse performance on answerable questions might be that the baseline model might be biased towards predicting an answer on border cases while our final model is more confident for a no-answer choice.
We have also classified each question in dev dataset by its question type and compare the performance of our model with that of the baseline model by question type. Figure 2 shows that our model improves decisively upon the baseline model across the question spectrum. The largest percentage improvement comes from who type.

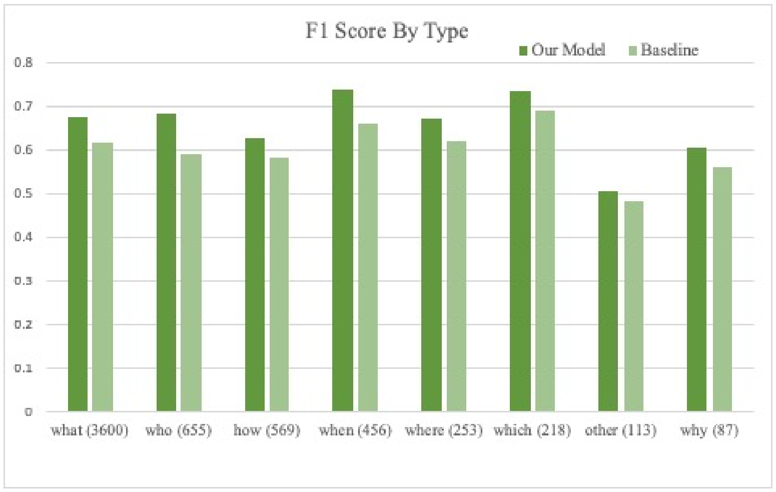
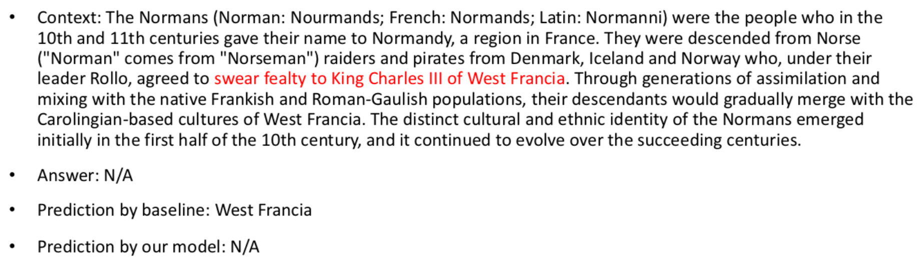


Figure 2: F1 score by question type

5.2 Qualitative error analysis



- Context: The Normans (Norman: Nourmands; French: Normands; Latin: Normanni) were the people who in the 10th and 11th centuries gave their name to Normandy, a region in France. They were descended from Norse ("Norman" comes from "Norseman") raiders and pirates from Denmark, Iceland and Norway who, under their leader Rollo, agreed to swear fealty to King Charles III of West Francia. Through generations of assimilation and mixing with the native Frankish and Roman-Gaulish populations, their descendants would gradually merge with the Carolingian-based cultures of West Francia. The distinct cultural and ethnic identity of the Normans emerged initially in the first half of the 10th century, and it continued to evolve over the succeeding centuries.
- Answer: N/A
- Prediction by baseline: West Francia
- Prediction by our model: N/A

Figure 3: Example 1 from dev set

In Figure 3, we show an adversarial example which the baseline model fails but our model succeeds. In order to answer this question correctly, the model has to not only understand the context but also reason about the irrelevance of the question to the context. By reading through some adversarial examples, we find one common pattern of the adversarial examples is that the question asks who did a thing while nobody in the context actually did it. There are 351 unanswerable questions of who type. The baseline only answers 172 questions correctly while our final model answers 219 questions correctly. This accounts for almost all the improvement in the who type question.

6 Conclusion

In this project, we implemented a BiDAF variant neural network that improves more than 6 points on a single model performance of SQuAD2.0 dataset. We first added character-level embedding to the baseline model, and then extended the model with a residual self-attention layer. We have experimented with various attention structures and many parameter settings. Our ablation studies show that char-level embedding improves the BiDAF by 2 points while the addition of residual self-attention layer improves by 4 points. We performed error analysis by question type especially the who type question to reason about the source of our improvements. We showed that our residual self-attention structure improves decisively the performance on unanswerable questions.
We have experimented with multi-head attention structure from transformer model[11] in residual self-attention layer, but we only get EM 63.67 and F1 66.58 on dev set, so we did not adopt this structure in our final model. Due to the superiority of the transformer model and good performance of BERT-

based models on the SQuAD2.0 dataset, we believe that this structure has space for improvement if more time and computing resource is given.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *CoRR* abs/1409.0473 (2014). arXiv: 1409.0473. URL: http://arxiv.org/abs/1409.0473.

[2] Kyunghyun Cho et al. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *CoRR* abs/1406.1078 (2014). arXiv: 1406.1078. URL: http://arxiv.org/abs/1406.1078.

[3] Christopher Clark and Matt Gardner. "Simple and Effective Multi-Paragraph Reading Comprehension". In: *CoRR* abs/1710.10723 (2017). arXiv: 1710.10723. URL: http://arxiv.org/abs/1710.10723.

[4] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[5] Robin Jia and Percy Liang. "Adversarial Examples for Evaluating Reading Comprehension Systems". In: *CoRR* abs/1707.07328 (2017). arXiv: 1707.07328. URL: http://arxiv.org/abs/1707.07328.

[6] 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)*. 2014, pp. 1532–1543.

[7] Pranav Rajpurkar, Robin Jia, and Percy Liang. "Know What You Don't Know: Unanswerable Questions for SQuAD". In: *CoRR* abs/1806.03822 (2018). arXiv: 1806.03822. URL: http://arxiv.org/abs/1806.03822.

[8] Pranav Rajpurkar et al. "SQuAD: 100, 000+ Questions for Machine Comprehension of Text". In: *CoRR* abs/1606.05250 (2016). arXiv: 1606.05250. URL: http://arxiv.org/abs/1606.05250.

[9] Min Joon Seo et al. "Bidirectional Attention Flow for Machine Comprehension". In: *CoRR* abs/1611.01603 (2016). arXiv: 1611.01603. URL: http://arxiv.org/abs/1611.01603.

[10] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. "Highway Networks". In: *CoRR* abs/1505.00387 (2015). arXiv: 1505.00387. URL: http://arxiv.org/abs/1505.00387.

[11] Ashish Vaswani et al. "Attention Is All You Need". In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[12] Wenhui Wang et al. "Gated self-matching networks for reading comprehension and question answering". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vol. 1. 2017, pp. 189–198.

[13] Dirk Weissenborn, Georg Wiese, and Laura Seiffe. "FastQA: A Simple and Efficient Neural Architecture for Question Answering". In: *CoRR* abs/1703.04816 (2017). arXiv: 1703.04816. URL: http://arxiv.org/abs/1703.04816.

[14] Matthew D. Zeiler. "ADADELTA: An Adaptive Learning Rate Method". In: *CoRR* abs/1212.5701 (2012). arXiv: 1212.5701. URL: http://arxiv.org/abs/1212.5701.