# Hierarchical Attention Fusion Network for Question Answering

**Yang Chen**
LinkedIn
Mountain View, CA 94043
yangc27@stanford.edu

**Marius Seritan**
LinkedIn
Mountain View, CA 94043
mseritan@stanford.edu

## Abstract

In this project we explored the Hierarchical Attention Network for question answering. In particular, we experimented with the following modifications to the Bidirectional Attention Flow (BiDAF) model: (1) adding character embedding to address the out-of-vocabulary words, (2) applying fusion network to combine the context and question hidden states from the encoder layer and the corresponding attention vectors instead of directly concatenating them, and (3) incorporating passage and question self-attention to further refine the representations. We compared the F1 and EM scores of the Hierarchical Attention Network with BiDAF. Results show that passage self-attention led to the major performance boost, while the other components also helped improve the performance on the dev set. Overall, we achieved F1 score of 64.6 on the dev set, and 61.6 on the test set. Comparing with the baseline, we improved the F1 by 3.2 on the dev set. Error analysis of the result shows that the most promising next step would be developing an accurate no-answer prediction strategy.

## 1 Introduction

In this project, we focus on reading comprehensive style question answering (QA). Particularly, we seek to optimize the model for the Stanford Question Answering Dataset (SQuAD) 2.0 (1), in which the answers are manually collected through crowdsourcing, and are constrained to spans within the reference passage. Comparing with SQuAD 1.0, the new version contains a good amount of questions with no answers.

A lot of efforts have been made since the release of the dataset. One major direction is around introducing effective attention layer. For example, Wang and Jiang (2) introduced match-LSTM to build question-aware passage representation and predicted answer boundaries in the passage with pointer networks (3). Seo et al. proposed the bi-directional attention flow networks to model question-passage pairs (4). Xiong et al. proposed fusing the co-attended representations of the question and passage, and iterates over potential answer spans with a dynamic pointing decoder to allow recovering from local maxima (5).

Attention is an effective way to capture the relation between the question and the passage. It produces more meaningful representations for both questions and passages than the encoded word embedding. Also, a network architecture with attention at different levels offers consistency with human reading pattern: when looking for answers from a passage, we usually try to connect the passage with the question, and understand more deeply about the intention of the question related to the passage theme. After that, we roughly locate the most relevant part of the passage and identify the best answers from there.

Preprint. Work in progress.

Inspired by previous works (4; 6; 7), we decided to focus on experimenting different attention mechanisms and compare with the bi-directional Attention Flow (BiDAF) model as the baseline. Our model, modified on top of BiDAF (4), consists of four joint layers: 1) encoder layer where recurrent neural network was used to build representations for question and passage separately. Though our reference paper (7) proposed incorporating pre-trained contextual embedding (PCE), we did not include this part because we wanted to experiment the effectiveness of the attention alone and participate in the non-PCE leader board. 2) Gated co-attention combined with the original representations. 3) Self-attention layer for passage and question separately to further refine the representations. 4) Match question and passage and output the probability for start and end point for the answer span. The major differences comparing with BiDAF include 1) fusion function at the co-attention layer, 2) self-attention layer, 3) character embedding at the embedding layer, and 4) bilinear match at the match layer. Experiment result shows that self-attention for the passage offered the largest performance boost.

## 2 Related Work

Previous work for QA on SQuAD can be roughly divided into three categories: one major direction is around applying pre-trained contextual embeddings, such as Embeddings from Language Models (ELMo) (8) and Bidirectional Encoder Representations from Transformers (BERT) (9). BERT has brought significant improvement to not only QA, but also many other natural language processing (NLP) tasks. We believe that this trend reflects that more meaningful representation is the key to success for NLP generally.

A second category of previous work on QA is about exploring different attention approaches. For example, Chen et al. developed a simple but effective model with bilinear match function and a few manual features (10). Wang and Jiang (2) build question-aware passage representation with match-LSTM. Seo at al. proposed the bi-directional attention flow networks to model question-passage relation (4); this model was considered as the baseline model in our experiment. Microsoft Research Asia introduced R-Net (11), which contain a self-matching attention mechanism to refine the representation by matching the passage against itself.

The third category focuses on building more accurate strategy for predicting "no-answer" probability, and is specifically for SQuAD 2.0. Sun et al. proposed U-Net, which depends on a universal node that encodes fused information from the question and the passage to predict whether the question is answerable (12). Hu et al. introduced auxiliary losses for a neural reader to better handle no-answer detection and built a answer verifier on top of that (13).

In this project, we focus on improving the attention layer, we incorporate both gated co-attention and self-attention. We also explore which component contributes most to the improvement.

## 3 Approach

### 3.1 Baseline model

Our baseline model is the BiDAF model (4) with the following five layers from the bottom to the top:

1. Embedding layer, which contains an fixed Glove embedding look up table and a two-layer highway network
2. Encoder layer, which consists of a bidirectional LSTM
3. Bidirectional attention flow layer
4. Modeling layer, which is a bidirectional LSTM to refine the question-aware context representation
5. Output layer, which applies softmax on the transformed the concatenated output of the attention layer and the modeling layer

Specifically in the attention layer, we first calculate a similarity matrix $S$:

$$S_{ij} = w_{sim}^T[c_i; q_j; c_i \circ q_j], \tag{1}$$

Table 1: Ratio of Out Of Vocabulary words

| Set | OOV Words | Total Words | Ratio |
|---|---|---|---|
| Train | 136,228 | 51,840,172 | 0.26% |
| Dev | 22,230 | 235,8170 | 0.94% |

where $c_i$ is the context hidden states from the encoder layer and $q_j$ is the question hidden states. We then get Context-to-question (C2Q) and Question-to-context (Q2C) attention from $S$: the C2Q output $a_i$ is calculated as the weighted sum of question hidden states

$$\bar{S}_{i,:} = softmax(S_{i,:}) \tag{2}$$

$$a_i = \sum_j \bar{S}_{i,j} q_j \tag{3}$$

The Q2C output $b_i$ is calculated as the weighted sum of context hidden states:

$$\bar{\bar{S}}_{:,j} = softmax(S_{:,j}) \tag{4}$$

$$S' = \bar{S}\bar{\bar{S}}^T \tag{5}$$

$$b_i = \sum_j S'_{i,j} c_j \tag{6}$$

### 3.2 Character embedding

In order to judge the usefulness of integrating character encoding, we evaluated the number of unknown words in the training and dev set.

Based on the analysis result shown in Table 1, we decided to combine character embedding (14) with word embedding to better handle out-of-vocabulary words. The character embedding layer contains embedding lookup, a convolutional network, and highway network with dropout. In the convolutional network, we use 1-dimensional convolutions, and learn a weight matrix $W$ and a bias vector $b$. Assume $x$ is the reshaped character embedding lookup result, the output of the convolutional network is calculated as

$$x_{conv} = Conv1D(x) \tag{7}$$

$$x_{conv_out} = MaxPool(ReLU(x_{conv}) \tag{8}$$

We experimented with adding character level embedding in BiDAF and in our model with hierarchical attention layers.

### 3.3 Hierarchical attention fusion network

The model we built consists of four layers (Figure 1): (1) word and character embedding encoded by BiLSTM; (2) bi-directional co-attention, which was combined with the output of the first layer with a fusion function; (3) passage and question self-attention, which was combined with the co-attention with a funsion function; (4) bilinear match between passage and question and output span probability.

When implementing the network, we adopted from BiDAF the embedding and encoder layer, the similarity matrix calculation, and question-to-passage (Q2P) and passage-to-question (P2Q) co-attention.

### 3.4 Co-attention

In the co-attention layer, we replace the concatenation between attention and original representation with a gated fusion function. Specifically, assume that the original context/passage representation and question representation is $P$ and $Q$, respectively. The aligned representation for context/passage is $\tilde{P} = \{b_j\}, j = 1, ..., N$ and for question is $\tilde{Q} = \{a_i\}, i = 1, ..., N$. We first fuse the original and aligned representation as follows:

$$P' = g(P, \tilde{Q}) \cdot m(P, \tilde{Q}) + (1 - g(P, \tilde{Q})) \cdot P \tag{9}$$

where $g$ is a gate function and $m$ is defined as

$$m(P, \tilde{Q}) = tanh(W_f[P; \tilde{Q}; P \circ \tilde{Q}; P - \tilde{Q}] + b_r) \tag{10}$$

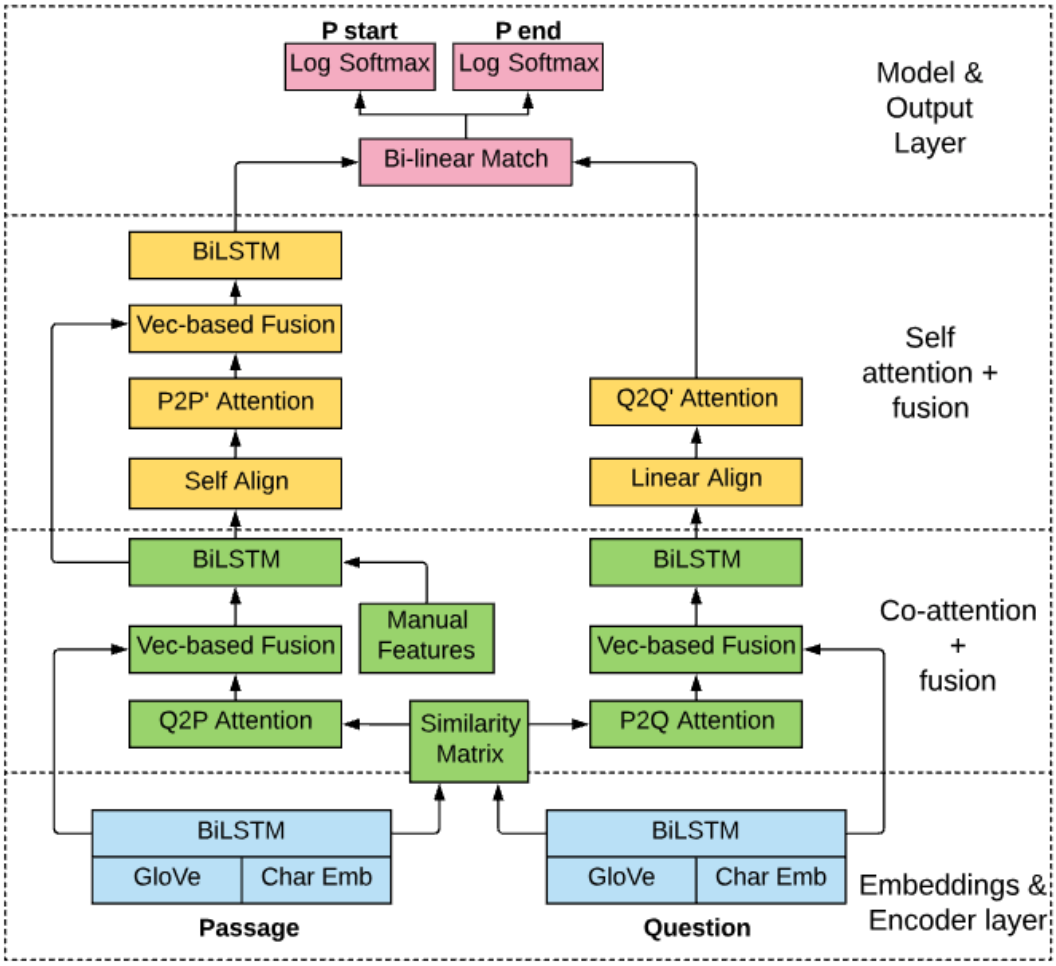


Figure 1: Architecture for hierarchical attention fusion network

### 3.5 Self-attention

In the self-attention part, which further refine the obtained information from the context-question-attention layer. In this layer, we explore adding one single manual feature, the frequency of each word, to the question-aware passage representation. After combining the manual feature with $p'$, we further encode the refined question-aware passage representation:

$$D = BiLSTM([P'; feat_{man}]) \tag{11}$$

and then calculate bilinear self-attention:

$$L = softmax(D \cdot W_1 D^T) \tag{12}$$

Therefore the self-aligned representation is

$$\tilde{D} = L \cdot D \tag{13}$$

After that, we apply the fusion function again to get

$$D' = g(D, \tilde{D}) \cdot m(D, \tilde{D}) + (1 - g(D, \tilde{D})) \cdot D \tag{14}$$

Finally, a BiLSTM was used to get the final passage repesentation:

$$D" = BiLSTM(D') \tag{15}$$

We also explored question self-attention with the following steps: first BiLSTM is applied on the fused question representation:

$$Q" = BiLSTM(Q') \tag{16}$$

Table 2: BiDAF performance with different dropout probability

| Dropout probability | EM | F1 |
|---|---|---|
| 0.0 | 58.03 | 61.4 |
| 0.2 | 57.44 | 60.74 |
| 0.5 | 52.13 | 55.53 |

Then we aggregate the result:

$$\gamma = softmax(w_q^T \cdot Q") \tag{17}$$

Finally, we calculate the self-aligned question representation:

$$q = \sum_j \gamma_j \cdot Q"_{:j}, j \in [1, ..., m] \tag{18}$$

### 3.6 Matching and output layer

We use bilinear matching between question and passage:

$$P_{start} = softmax(q \cdot W_s^T \cdot D") \tag{19}$$

$$P_{end} = softmax(q \cdot W_e^T \cdot D") \tag{20}$$

## 4 Experiments

### 4.1 Data and evaluation method

We trained and tuned models on the provided SQuAD 2.0 dataset, and followed the default setting on splitting train, dev and test set.

We use the EM and F1 metrics as defined by SQuAD to evaluate models. We compared our results with the baseline model and on the class leaderboards.

### 4.2 Experimental details and results

#### 4.2.1 Dropout hyper parameter tuning

For our first experiment we ran the baseline model BiDAF and tuned the dropout probability to understand the status of the model. Table 2 shows the performance of BiDAF with different dropout probability. The result shows that with probability 0, the BiDAF model achieved the best F1 and EM score. It indicates that the model is not overfit, but possibly underfit. This motivates us to add more layers and parameters to better model the problem. In the following experiment, we considered BiDAF with dropout probability 0 as the baseline to compare with.

#### 4.2.2 Character embedding

Our second experiment is adding character embedding to the BiDAF model. We tuned CNN kernel and its output layer size as hyper-parameters for the character embedding layer. The results are as shown in Table 3. With kernel size 5 and output size 8, We got an F1 score of 62.43. The result demonstrate that fine tuned character embedding indeed contribute to improve the performance by better handling unknown words. This again shows that better representations is the key in NLP problems.

### 4.3 Hierarchical attention fusion network

Our third experiment is building the hierarchical attention fusion network described in Section 3.4 - 3.6. In the following parts, we call this model SLQA (Semantic Learning for Query Answering) for short (7).

Table 3: BiDAF + character embedding performance

| Kernel | Output | EM | F1 |
|---|---|---|---|
| 3 | 8 | 58.63 | 61.96 |
| 5 | 16 | 58.88 | 62.41 |
| 5 | 8 | 59.15 | 62.43 |

Table 4: Performance comparison between SLQA and BiDAF

| SLQA Layers | Dev | | Test | |
|---|---|---|---|---|
| | EM | F1 | EM | F1 |
| Embedding (Word + Char) + Output | 47.01 | 48.65 | - | - |
| ... + Paragraph Co-attention + Fusion | 51.59 | 54.09 | - | - |
| ... + Paragraph Self Attention+ RNN | 60.78 | 63.62 | 57.95 | **61.59** |
| ... + Question Self Align + RNN | 61.45 | **64.61** | 57.80 | 61.35 |
| ... + Manual feature | 58.60 | 61.76 | - | - |
| Baseline | EM | F1 | EM | F1 |
| BiDAF (Dropout prob = 0) | 58.03 | 61.4 | - | - |
| BiDAF + Char embedding | 59.15 | 62.43 | - | - |

We have taken a layered implementation strategy: we started with the embedding and output layers of BiDAF, and incrementally added the middle layer on top of them. Metric comparison in Table 4 and the Tensorboard curves in Figure 2 show that the initial performance was low, but the scores started to improve as we added more layers. As compared with the BiDAF, the SLQA model have raised the F1 score on the dev set by $3.07\%$ to $64.61$. The passage self-attention layer brought the largest performance boost.

# 5 Analysis

## 5.1 Analyzing performance

While our changes in attention improved the performance, the reference paper (7) claimed a higher F1 score (74.43) on SQuAD 2.0 than ours in the experiments. We can see a couple of reasons for this result:

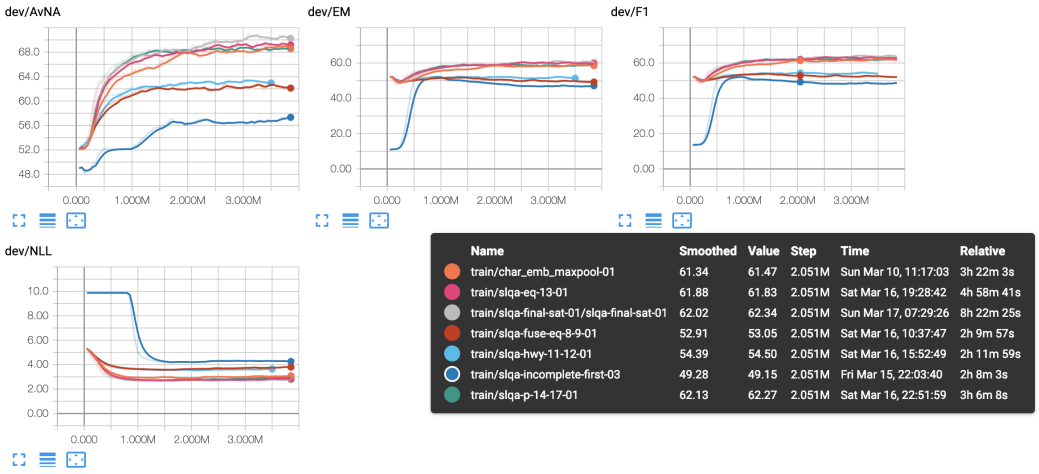


Figure 2: Tensorflow board

Table 5: Summary of error cases

| Situation | Count |
|---|---|
| Exact match | 2919 |
| We don't answer when we should | 2624 |
| Not exact match | 209 |
| We answer when we shouldn't | 199 |

Table 6: Counts for error examples with different keywords

| Question | Avg EM | Avg F1 | Total |
|---|---|---|---|
| what | 49.7 | 50.0 | 3496 |
| when | 44.8 | 44.9 | 429 |
| which | 37.3 | 38.7 | 212 |
| who | 51.9 | 52.1 | 640 |
| who-when | 40.0 | 40.0 | 10 |
| where | 48.2 | 48.2 | 251 |
| how | 52.3 | 52.3 | 555 |
| – | 49.4 | 49.7 | 174 |

1. We are using different training, dev, and test set comparing with the model submitted to the public leader board.
2. The paper started with ELMo pre-trained contextual embeddings. We did not want to explore this category because we wanted to stay in the non-PCE leader board and we had a fined tuned baseline to compare with.
3. Due to the limited time, we only explored one manual feature, which is the frequency of each word. It actually caused decreasing F1 score. This indicates that manual features could have high impact on the final result, but frequency is not an effective feature to choose. Better feature could include entity tagging, part-of-speech tagging, etc.
4. We used a different similarity matrix that co-attention depends on. This could also contribute to the performance disparity.

### 5.2 Analyzing errors

We categorize the error cases based on the failing reasons and count the number of examples that fall into each category. Result is shown in Table 4, in which "Exact match" means the correctly answered examples, and the other three rows represent different failing examples. The result shows that the "We don't answer when we should" category contributes to 87% of the errors. This motivates us to build a more accurate no-answer predictor in the next step.

We also did classify the error examples based on the single keyword they contain (Table 5) and the keyword combinations (Table 6). Interestingly, we found that the model works best for the "how" questions and the worst for the "which" questions. The reason for the bias could be mixed and needs further investigation.

## 6 Conclusion

We experimented fine tuning the baseline model, adding character embedding to the baseline model, and building a hierarchical attention fusion network. The new model with both co-attention and self-attention achieved better F1 score comparing with the baseline. Analysis of different layers shows that passage self-attention led to the largest performance boost. Other components such as character embedding and fusion functions also contribute. Based on the error analysis, our future work will mainly focus on building an accurate no-answer predicting strategy.

Table 7: Counts for error examples with different keyword combinations

| Question | Avg EM | Avg F1 | Total |
|---|---|---|---|
| what-when | 46.3 | 47.1 | 95 |
| what-which | 32.1 | 32.1 | 28 |
| what-who | 45.5 | 45.5 | 11 |
| when-how | 40.0 | 40.0 | 10 |
| what-where | 55.6 | 55.6 | 9 |
| what-how | 66.7 | 66.7 | 9 |
| when-which | 20.0 | 20.0 | 5 |
| who-where | 80.0 | 80.0 | 5 |

## References


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

[2] Shuohang Wang and Jing Jiang. 2016. Learning natural language inference with LSTM. In NAACL HLT 2016, The 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, San Diego California, USA, June 12-17, 2016.

[3] Vinyals, Oriol and Fortunato, Meire and Jaitly, Navdeep. Pointer networks. Advances in Neural Information Processing Systems, 2692–2700, 2015.

[4] Seo, M., Kembhavi, A., Farhadi, A., Hajishirzi, H. 2016. Bidirectional attention flow for machine comprehension. arXiv preprint arXiv:1611.01603.

[5] Caiming Xiong, Victor Zhong, Richard Socher. 2016. Dynamic coattention networks for question answering. arXiv preprint arXiv:1611.01604.

[6] Wang, W., Yang, N., Wei, F., Chang, B. Zhou, M., 2017. 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, pp. 189-198).

[7] Wang, W., Yan, M., Wu, C. 2018. Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. arXiv preprint arXiv:1811.11934.

[8] Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K. and Zettlemoyer, L., 2018. Deep contextualized word representations. arXiv preprint arXiv:1802.05365.

[9] Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

[10] Chen, D., Fisch, A., Weston, J. and Bordes, A., 2017. Reading wikipedia to answer open-domain questions. arXiv preprint arXiv:1704.00051.

[11] Microsoft Research Asia. 2017. R-NET: Machine Reading Comprehension with Self-matching Networks. https://www.microsoft.com/en-us/research/wp-content/uploads/2017/05/r-net.pdf

[12] Sun, F., Li, L., Qiu, X. and Liu, Y., 2018. U-Net: Machine Reading Comprehension with Unanswerable Questions. arXiv preprint arXiv:1810.06638.

[13] Hu, M., Peng, Y., Huang, Z., Yang, N. and Zhou, M., 2018. Read+ verify: Machine reading comprehension with unanswerable questions. arXiv preprint arXiv:1808.05759.

[14] Kim, Y., Jernite, Y., Sontag, D., Rush, A. M. 2016. Character-aware neural language models. In Thirtieth AAAI Conference on Artificial Intelligence.