Answer Pointer Inspired BiDAF And QANet For Machine Comprehension

Stanford CS224N Default IID Project
Mentor: Prerna Khullar

Weiyun Jiang
Department of Electrical Engineering
Stanford University
wyjiang@stanford.edu

Fang Qin
Department of Electrical Engineering
Stanford University
fangq@stanford.edu

Bingxian Chen
Department of Bioengineering
Stanford University
bchen72@stanford.edu

Abstract

Machine comprehension has been one the hottest topics in natural language processing (NLP) over the past few decades. With the help of emerging high-performance GPUs, deep learning for machine comprehension has progressed tremendously. RNN-based methods, such as Match-LSTM [1] and Bidirectional Attention Flow (BiDAF) model, and transformer-like methods, such as QANet [2], keep pushing the performance boundary of machine comprehension on the SQuAD datasets [3, 4]. Our team proposes to improve the performance of the baseline BiDAF and the QANet models on SQuAD 2.0. We replace the original output layer of BiDAF and QANet with Answer Pointer inspired output layers and add character level embedding and ReLU MLP fusion function to the baseline BiDAF model. We achieve significantly better performance using ensemble learning with majority voting on modified BiDAF, QANet1, and QANet3 models. Specifically, the ensemble learning achieves a F1 score of **66.219** and an EM score of **62.840** on the test datasets and a F1 score of **68.024** and an EM score of **64.561** on the validation datasets.

1 Introduction

Machine comprehension has always been a hot topic for natural language processing (NLP) research. Researchers try to find the best way for computers to understand a given context and output the correct answers to corresponding questions. In the past, researchers relied on traditional NLP algorithms, such as sentimental analysis, syntactic parsing and semantic parsing, to solve machine comprehension problems. In addition, researchers tended to utilize classical benchmark datasets [5, 6], which have the correct answer among several potential answers. With the help of emerging high-performance GPUs, deep learning methods, such as Birectional Attention Flow (BiDAF) [7], QANet [2], and Match-LSTM [1], have made tremendous progress in the area of machine comprehension. The boundary of machine comprehension is further pushed due to the emergence of SQuAD 2.0 [4], which is crowd-sourced like SQuAD 1.0 [3]. Thus, these emerging datasets provide more realism to the machine comprehension tasks. SQuAD 2.0 is even better than SQuAD 1.0 since the former includes adversarial no-answer questions.

In this paper, we propose to replace the regular output layers in both BiDAF and QANet with our specially designed Answer Pointer output layer. And we call these modified models Answer

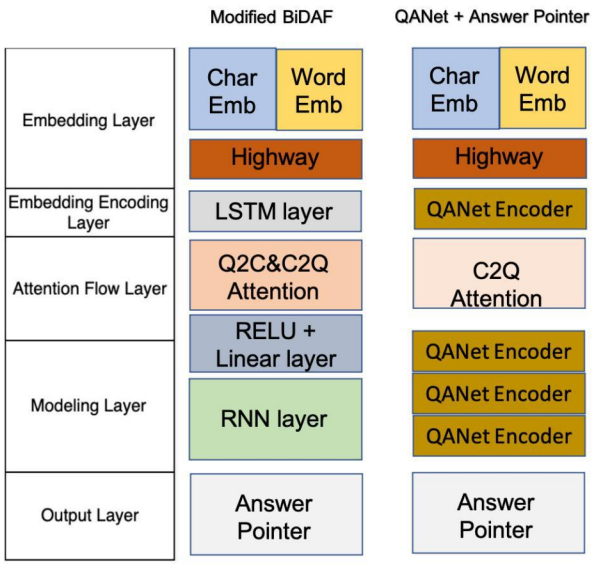

Figure 1: Our Answer Pointer inspired BiDAF and QANet

Pointer inspired BiDAF and QANet. In addition, we further improve the baseline BiDAF model with character level embedding and ReLU MLP fusion functions.

1.1 Paper Contributions and Organizations

Our contributions can be summarized as follows:

- We propose Answer Pointer inspired BiDAF and QANet 1, where the output layers of these two networks are replaced with our special designed Answer Pointer layer.
- We achieve significant improvement using ensemble learning compared with the baseline BiDAF model. We combine predictions from Answer Pointer inspired QANet, regular QANet and enhanced BiDAF through majority voting. The ensemble learning achieves a F1 score of **66.219** and an EM score of **62.840** on the test datasets and an F1 score of **68.024** and an EM score of **64.561** on the validation datasets.
- We also find that Answer Pointer inspired output layers have significant improvement when the previous layers of the model have relatively less parameters. And the Answer Pointer layer itself needs to be small when the model is extremely complex.

The rest of this paper is organized as follows: Section 2 briefly talks about the related work in attention-based methods and transformer-like methods. Section 3 discuss the neural network architecture of Answer Pointer inspired BiDAF and Answer Pointer inspired QANet. Section 4 talks about the experimental setup and results. Section 5 analyzes the performance of Answer Pointer output layer in different models and the qualitative performance of various models on particular question types, such as "what", "why", "who" and so on. Finally, Section 6 concludes the paper.

2 Related Work

Machine comprehension and Question answering is one of the most popular topics in natural language processing. Two of the promising methods are RNN-based neural network and transformer-like neural network.

For RNN-based neural network, a lot of approaches "attend" to a certain short part when processing a large amount of information or use uni-directional attention. Later works try to solve the problem with previous methods. The Bi-Directional Attention Flow (BIDAF) network[7] proposed a multi-stage hierarchical process and use bi-directional attention. Machines are expected to understand the text by

answering a question given a passage. [1] proposed to combine match-LSTM [8] with the Answer Pointer layer, which selects the start and end tokens coordinately. Google[9] has introduced a model entirely rely on self-attention to compute input and output.

For transformer-like neural network, they don't reply on RNNs but multi-head attention instead. The QAnet model[2] combined local interactions captured by convolution models and global interactions captured by self-attention models. Transformer model has limitation of fixed-length context in the setting of language modeling. Another approach, Transformer-XL models[10] solves this problem by proposing a segment-level recurrence mechanism and a positional encoding scheme.

3 Approach

Our best model is a majority voting based ensemble learning that combines three different models: one Answer-Pointer inspired QANet model [2], one regular QANet model and one enhenced BiDAF [7] model with character embedding and ReLU MLP fusion function. We adapt the QAnet model from this git repository: https://github.com/andy840314/QANet-pytorch-.git. We implement the ensemble learning, character embedding, ReLU MLP fusion function and various Answer Pointer inspired output layer on our own.

3.1 Answer Pointer Inspired BiDAF

The baseline model is based on the Bidirectional Attention Flow (BiDAF) model for machine comprehension as described by Seo et al except that it does not include the character level embedding and ReLU MLP function function [7].

1. ***Embedding Layer***

   We follow the BiDAF paper [7] and write our own code to add the additional character-level embedding layer. Each word in the context and question are mapped to a high dimensional vector space. Unlike the original work of taking this vector as the 1D input to a CNN network, we treat this vector as an input to a 2D CNN network. The output of the CNN network is then applied through a ReLU function followed by max-pooling over the entire width. Then, a dropout rate of 0.1 is used for the character embedding vector while a dropout rate of 0.05 is used for the word embedding vector. Finally, both character embedding vector and word embedding vector are concatenated and passed to the Highway network.

2. ***Attention Layer***

   We follow the BiDAF paper [7] and add a ReLU MLP funtion function after the attention layer output, $\mathbf{g_i} \in \mathbb{R}^{8H} \forall i \in \{1, \ldots, N\}$, where H is the hidden size and N is the context length. This new attention output, $\hat{\mathbf{g}}_\mathbf{i}$ can be formally formulated as follows:

   $$\hat{\mathbf{g}}_\mathbf{i} = ReLU(\mathbf{W_a}(\mathbf{g_i}) + \mathbf{b_a}) \tag{1}$$

3. ***Answer Pointer Output Layer***

   We follow the paper [1] and write our own code to replace the BiDAF output layer in the baseline model with the Answer Pointer output layer. The purpose of the Answer Pointer layer is to condition the probability of selecting corresponding word as the end token, p_{end} by the probability of selecting corresponding word as the start token, p_{start}. The Answer Pointer layer is a two-time step process. During the first time step, the layer performs a linear layer followed by a softmax layer on the input $\mathbf{H^r} \in \mathbb{R}^{2H \times N}$, where H is the size of hidden layer and N is the length of the passage, to generate p_{start}. Then, p_{start} is fed into a LSTM layer and the outputted hidden state of p_{start} is again fed back into the linear layer followed by a softmax layer to predict p_{end}. The above process can be formulated as follows:

   $$\mathbf{F_k} = tanh(\mathbf{V H^r} + (\mathbf{W^a h^a_{k-1}} + \mathbf{b^a}) \otimes \mathbf{e}_N) \tag{2}$$

   $$p_k = softmax(\mathbf{v^T F_{start}} + c \otimes \mathbf{e}_N) \tag{3}$$

where k can represent start or end tokens, $\mathbf{H^r} \in \mathbb{R}^{2H \times N}$ is the input hidden state representations of the context, $\mathbf{V} \in \mathbb{R}^{H \times 2H}$, $\mathbf{W^a} \in \mathbb{R}^{H \times H}$, $\mathbf{b^a}, \mathbf{c} \in \mathbb{R}^H$ and $c \in \mathbb{R}$ are trainable parameters, the outer product ($\cdot \otimes \mathbf{e_N}$) produces a matrix or row vector by repeating the vector or scalar on the left for N times, $\mathbf{h^a_{k-1}} \in \mathbb{R}^H$ is the hidden vector of the previous position of the LSTM defined below:

$$\mathbf{h^a_{k-1}} = \overrightarrow{LSTM}(\mathbf{H^r} p_k^T) \tag{4}$$

In this paper, we initialize our hidden vector $\mathbf{h^a_{k-1}}$ as 0. We explore two kinds of design options for the input hidden state context representations, $\mathbf{H^r}$. For the first version, we concatenate the outputs from the attention layer and the modeling layer and feed them into a linear layer, $\mathbf{W}[\mathbf{G}; \mathbf{M}] + \mathbf{b}$, where $\mathbf{W} \in \mathbb{R}^{2H \times 10H}$, $[\mathbf{G}; \mathbf{M}] \in \mathbb{R}^{10H \times N}$. For the second version, we simply concatenate the outputs from the attention layer and the output layer, $[\mathbf{G}; \mathbf{M}] \in \mathbb{R}^{8H \times N}$.

3.2 QANet

1. ***Embedding Layer***

   We use the same embedding layer as BiDAF embedding layer, where we concatenate word embedding vector and character embedding vector after the dropout layers. We set character embedding hidden size, $f = 200$, in QANet.

2. ***Encoder Layer***

   This layer is consisted of the following modules: positional encoder, depthwise separable convolutions layer, self-attention layer and feed-forward layer. The self-attention layer is a multi-head attion instead of single attention. Queries, keys and values are linearly projected and then applied with scaled-dot product attention in parallel to get output values. The three result values are further projected and then concatenated to get the final output.

3. ***Context-Query Attention Layer***

   We use the same attention layer as the BiDAF baseline model.

4. ***Regular Output Layer***

   The regular output layer predicts p_start and p_{end} independently. $\mathbf{p_{start}} = \mathbf{W_1}(\mathbf{M1}; \mathbf{M2})$ and $\mathbf{p_{end}} = \mathbf{W_2}(\mathbf{M1}; \mathbf{M3})$.

5. ***Answer Pointer Layer***

   We explore three kinds of Answer Pointer layers for the QANet. Two out of three take the LSTM structure discussed in Section 3.1. We find that these kinds of LSTM based Answer Pointer layers do not perform really well on the QANet from earlier training epochs. Thus, we decide to borrow the key concept of Answer Pointer output layer, which is to predict p_end based on p_start. Without a LSTM layer, we choose to use fully connected layers to parameterize p_{start} and p_{end}. p_{start} can be formulated as follows:

   $$\mathbf{p_{start}} = \mathbf{W_1}(\mathbf{M_1}; \mathbf{M_2}) \tag{5}$$

   And p_{end} can be computed given p_{start} and formulated as:

   $$\mathbf{p_{end}} = \mathbf{W_3}(\mathbf{W_2}(\mathbf{M1}; \mathbf{M3}), \mathbf{p_{start}}) \tag{6}$$

4 Experiments

4.1 Data

We used the Stanford Question Answering Dataset (SQuAD) 2.0 [4], which combines 100,000 questions in the old dataset, SQuAD 1.0 [3]. This dataset provides over 150,000 answerable and non-answerable questions, setting the benchmark for machine comprehension.

Table 1: Detailed model configurations for QANet experiments

Models	Batch	Hidden	# Heads	# Convs Emb	# Convs Mod	# Blks Emb	# Blks Mod
QANet 1	32	64	4	3	3	1	4
QANet 2	32	128	8	4	2	1	3
QANet3	16	128	8	4	2	1	7

4.2 Evaluation Methods

Quantitatively, we used the official SQuAD evaluation metrics EM and F1 scores to evaluate the performance of our model. AvNA, which stands for Answer vs. No Answer and measures the classification accuracy of our model regarding its prediction on whether the given passage has an answer, and train/ dev loss were also included for debugging purposes.
Qualitatively, we also evaluated our model's performance on specific question types ("why", "who", "where", "how", etc) and answer lengths.

4.3 Experimental Details

We use Nvidia K80 on Microsoft Azure for various BiDAF training and Nivida V100 on Google compute platform for various QANet training.

BiDAF Experiments. We use Adadelta optimizer with a learning rate of 0.5. For all BiDAF experiments, the batch size is 64, and the hidden size is 100. We train every BiDAF model, except BiDAF + Ans-Ptr-v1, for 30 epochs. We further train the BiDAF + Ans-Ptr-v1 for extra 15 epochs and demonstrates the quantitative performance at both epoch 30 and 45.

- ***BiDAF + Character Embedding***
  We train the BiDAF baseline model with additional layer of character-level embedding for 30 epochs, with the kernel size of CNN equals 5 and the hidden size of the character-level embedding equals 50.
- ***BiDAF + Answer Pointer***
  We explore two versions of Answer Pointer on BiDAF, described in Section 3.1. And we train version 1 with extra 15 epochs.
- ***BiDAF + Character Embedding + ReLU MLP Fusion Function***
  We explore two kinds of character embedding hidden size, $f = 50$ and $f = 200$.
- ***BiDAF + Character Embedding + ReLU MLP Fusion Function + Answer Pointer***
  We choose to use Answer Pointer version 1 here since version 1 has better performance than version 2 on the baseline model at epoch 30. We also explore two kinds of character embedding hidden size, $f = 50$ and $f = 200$.

QANet Experiments. Following the trainig details in the QANet paper [2], we use ADAM optimizer with $lr = 0.001, \beta_1 = 0.8, \beta_2 = 0.999, \epsilon = 10^{-7}$. We also apply an exponential moving average (EMA) on all trainable variables with a decay rate 0.9999. Table 1 shows detailed QANet model configurations. We train all of our QANet models for 30 epochs.

- ***QANet 1 - 3*** According to Table 1, we design our QANet model from the most light, QANet 1 to the most complex, QANet 3. As we decrease the batch size, it becomes easier for the models to converge, leading to better performance. In addition, when we use 32 as our batch size, we have to decrease the number of encoder blocks in the modeling layer from 7 to 3. Otherwise, the memory on the GPU will explode.
- ***QANet 1 + Answer Pointer (no-lstm)*** We decide to replace the regular output layer in QANet 1 with our new Answer Pointer output layer (no LSTM). We choose QANet 1 because it is the lightest model among the three QANet configurations we have. From previous experiments on BiDAF, we find that Answer Pointer inspired output layer would improve the model if the model is not very complex.

Ensemble Learning. The main purpose of ensemble learning is to achieve better performance by combining predictions from various models. We believe that diversity of models is a significant factor

Table 2: Quantitative results summary

Models	Dev F1	Dev EM
BiDAF Baseline	60.43	57.08
BiDAF Baseline + Char-Emb (f=50)	63.33	60.17
BiDAF Baseline + Ans-Ptr-v1	61.08	57.82
BiDAF Baseline + Ans-Ptr-v1 (45 epochs)	61.92	58.48
BiDAF Baseline + Ans-Ptr-v2	60.93	57.67
BiDAF Baseline + Char-Emb (f=50) + Fusion	65.26	61.86
BiDAF Baseline + Char-Emb (f=200) + Fusion	**65.69**	**62.02**
BiDAF Baseline + Char-Emb (f=50) + Fusion + Ans-Ptr-v1	63.79	60.07
BiDAF Baseline + Char-Emb (f=200) + Fusion + Ans-Ptr-v1	**64.31**	**60.88**
QANet 1	**64.21**	**61.72**
QANet 1 + Ans-Ptr-No-LSTM	**64.21**	**60.59**
QANet 2	62.53	59.02
QANet 3	**66.91**	**63.15**
Ensemble Learning	**68.02**	**64.56**

to the success of ensemble learning. We would like to ensemble three of our models: QANet 1 with Answer Pointer (no LSTM), QANet 3 and BiDAF baseline with character embedding (f=200) and ReLU MLP fusion function through majority voting. If three models end up with three completely different predictions, we will just the prediction from QANet 3 because it has the best F1 and EM scores.

4.4 Results

Character-Level Embedding. Table 2 shows our model with character embedding result comparison with the baseline model. From Table 2, we can tell the performance improves after adding the character-level embedding layer. This is exactly what we expected because we are able to extract more information via character level tokens. Although the training gets a little overfitting at the end, we will work towards the final report to eliminate overfitting. Figure 2 shows that the model with character embedding begin to overfit around 2M iterations, similar to the baseline model. However, the model with character embedding achieves much less validation loss than the baseline model, which leads to better performance. In addition, we find that character embedding hidden size f has significant influence on the model performance. According to Table 2, models with higher embedding hidden size $f = 200$ tend to perform better than those with lower embeeding hidden size $f = 50$.

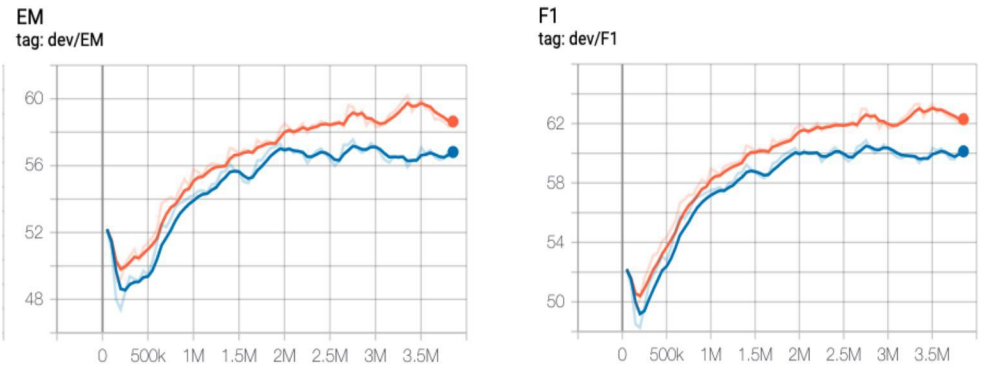

Figure 2: Dev EM and F1 score for baseline model with character-level Embedding (Orange: w/ Char-emb, Blue: Baseline).

Fusion Function. According to Table 2, ReLU MLP fusion function significantly improves the BiDAF performance. The F1 score on validation set increases from 63.33 to 65.26 after the ReLU MLP is added to the attention layer output. This is exactly what we expected since we have additional

MLP to express the attention.

Answer Pointer. According to Table 2, we find that Answer Pointer may not improve the performance on every model. We will disucss and analyze this behavior further in Section 5.

BiDAF [7] vs. QANet [2] QANet models generally outperform the BiDAF models in Table 2. And this is what we expected since QANet is published two year later than BiDAF. And QANet has more moden transformer-like architectures while BiDAF relies on LSTM-based encoders.

Ensemble Learning. Majority voting based ensemble learning, which combines prediction from three different models, achieves the best F1 and EM score on the validation datasets. This is also what we have expected since the resultant combined model has the advantages of all three models.

5 Analysis

Answer Pointer. The purpose of answer pointer is to predict the position of end tokens based on the position of start tokens. We find that Answer Pointer layers in BiDAF and Answer Pointer inspired output layers in QANet have better performance compared with their baseline models when the model itself is not very complex. What's more, we conclude that the Answer Pointer layer itself also needs to be light when the baseline model is extremely complex.
The Answer Pointer layers indeed have better performance when the baseline models are very light. For instance, simply replacing the regular output layer in baseline BiDAF with Answer Pointer increase the F1 score from 60.43 to 61.08 (Table 2). However, if we replace the regular output layer in a more complex model (BiDAF+Char-Emb+Fusion) with Answer Pointer output layer, there will be no improvement or even regression. We interpret this behavior as the cause of too much expressive power. If the model itself originally have a lot of parameters, the Answer Pointer layer will add even more expressive power to the model. And these kinds of expressive power can be too much for the model to generalize well. This is why we see Answer Pointer layers work well on small models than on bigger models.

Another interesting finding is that the the Answer Pointer layer itself needs to be light when the model is extremely complex. This is the case when we switch models from BiDAF to QANet. QANet is known to have much more expressive power than BiDAF because of its transformer-like architecture. When we try to apply LSTM-based Answer Pointer output layers to QANet, none of these layers work. Then, we change our LSTM-based Answer Pointer output layers to MLP-based Answer Pointer inspired output layers and we achieve comparable performance as the model with regular output layers. What's more, these Answer Pointer inspired output layers can only be applied on QANet 1, which has significantly less parameters than QANet 2 and 3. We also try QANet 2 and QANet 3 with MLP-based Answer Pointer Inspired output layer for some epochs. The performance is really poor at the early stage. Thus, we do not include the results in our table. In a nutshell, the Answer Pointer layer itself needs to be light when the model is extremely complex.

Performance on Specific Question Types. Since previous studies reported diminishing model performance with increasing answer length and relatively poor model performance regarding the "Why" questions, the current study analysed the current model's performance with respect to answer length and with respect to specific question types [1]. We examine the performance of our model in terms of the EM and F1 scores with respect to the answer length on the development set. There seem to be a negative relationship between our model performance metrics and answer length (Figure 3). For example, for questions with answers containing 10 tokens, the EM and F1 scores drop to 17.24 and 32.78 as compared to 70.69 and 73.69 for questions with answer composed of a single token. A similar trend is observed in the baseline model, indicating that our current model does not show improvement regarding answering questions with long answers.

Moreover, we also analyze the current model's performance on different groups of questions. We split the questions based on a set of question words we defined, including "what", "when", "who", "how", "why", "where", "which". According to the performance of our model on the development set, our current model achieves the best F1 and EM scores for "when" questions and relatively bad performance for "why" questions (FIgure 4). The diverse performance of our model

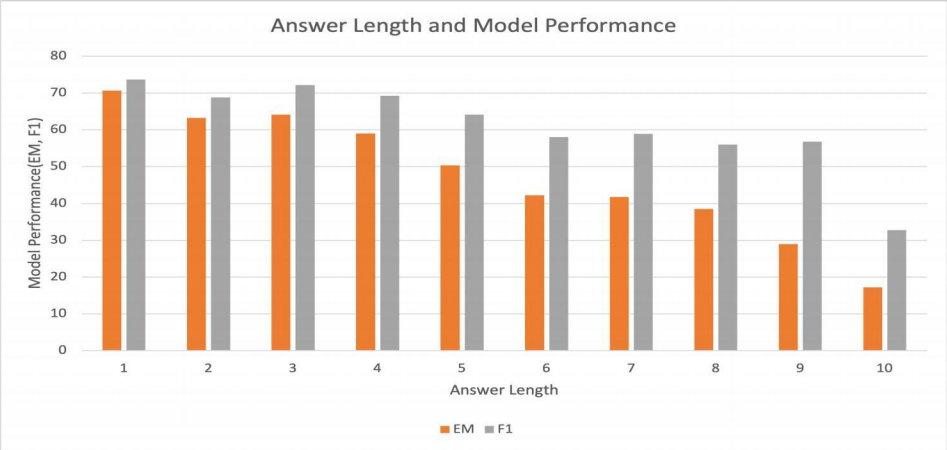

Figure 3: The EM (orange) and F1 (grey) scores of the ensemble model decrease with increasing answer length.

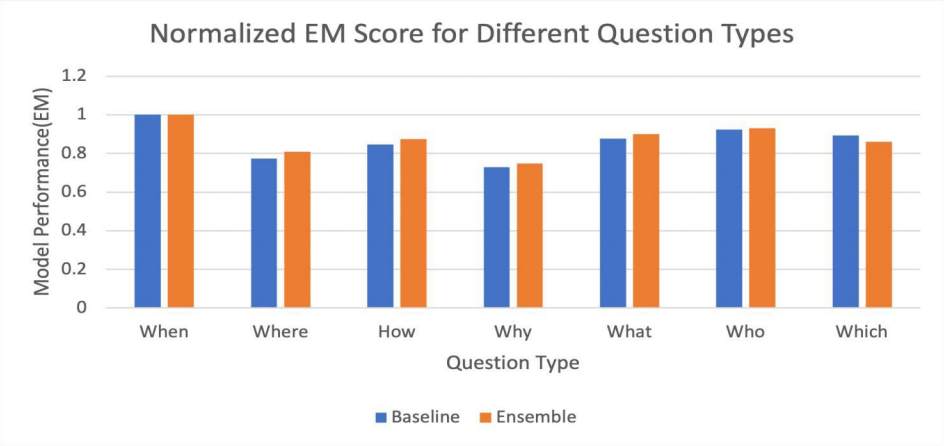

Figure 4: The normalized EM score is higher for "When" questions and lower for "Why" questions for both the baseline (blue) and our ensemble (orange) model

regarding different question types may be explained by the relative ease of extracting temporal information for the "when" questions as compared to extracting the relatively unpredictable phrase types for a "why" questions. We observe the same phenomenon across our models, which is inline with Wang and Jiang's finding [1].

6 Conclusion

In this paper, we propose to replace the regular output layers in both BiDAF and QANet with our specially designed Answer Pointer output layers. We achieve significant improvement using ensemble learning compared with the baseline BiDAF model. We combine predictions from Answer Pointer inspired QANet, regular QANet and enhanced BiDAF through majority voting. The ensemble learning achieves an F1 score of **66.219** and an EM score of **62.840** on the test datasets and an F1 score of **68.024** and an EM score of **64.561** on the validation datasets.

7 Acknowledgement

We would like to thank Prerna Khullar and Chris Waites for their insightful and helpful discussions.

References

[1] Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. *international Conference on Learning Representations (ICLR)*, 2017.

[2] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*, 2018.

[3] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics.

[4] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for SQuAD. In *Association for Computational Linguistics (ACL)*, 2018.

[5] Matthew Richardson, Christopher JC Burges, and Erin Renshaw. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 193–203, 2013.

[6] Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301*, 2015.

[7] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

[8] Shuohang Wang and Jing Jiang. Learning natural language inference with lstm. *Proceedings of the Conference on the North American Chapter of the Association for Computational Linguistics,*, 2016.

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

[10] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, and Ruslan Salakhutdinov. Transformer-xl: Attentive language models beyond a fixed-length context, 2019.