BiDAF and QANet Model Variations on SQuAD2.0

Stanford CS224N Default Project IID SQuAD track

Feiyang Liu
ICME
Stanford University
liuf9@stanford.edu

Zhuofan Xi
ICME
Stanford University
zfxi@stanford.edu

Huanzhong Xu
ICME
Stanford University
xuhuanvc@stanford.edu

Abstract

In this paper we build two models for the question-answering task on SQuAD 2.0 dataset: BiDAF (Bi-Directional Attention Flow) network and QANet. Based on vanilla BiDAF baseline model we first adopt coattention mechanism from Dynamic Coattention Network into BiDAF. Secondly we re-implement QANet. These two models get 61.855/62.309 EM and 65.309/65.595 F1 on development set respectively. Fusing Performer technique into QANet is also explored. Finally, we use a voting scheme ensemble model to achieve **F1 = 68.35** on dev set and **F1 = 65.69** on test set[1].

1 Key Information to include

- Mentor: Yian Zhang
- External Collaborators (if you have any): No
- Sharing project: No

2 Introduction

The task of question answering (QA) is to correctly answer a question to a paragraph given as context. The task is so crucial in natural language process that it attracts the interests of both research work and real-life applications. It's an interesting research topic since it's a good measurement of how well an end-to-end system can "understand" a given paragraph. From a practical perspective, a machine learning system than can understand questions from human can serve information need much broader than simple keyword searching, as we can see in Figure 1.

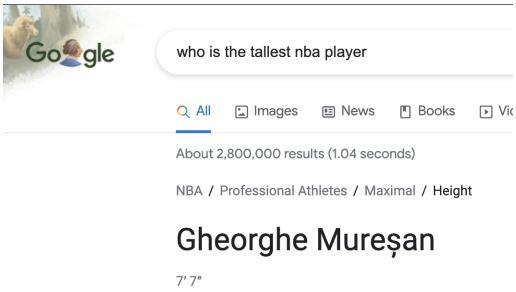

Figure 1: Google has implemented a pretty good online QA system.

There was a lack of datasets both of large scale and high quality for models to train on [1], restricting the application of deep learning models in this field. Recently a number of datasets have been

[1] Our implementation can be found at https://github.com/zhuofanx/cs224nproj

developed to solve this problem. One example is SQuAD [2], which contains paragraphs from Wikipedia and questions crowdsourced by Amazon Mechanical Turk. SQuAD2.0, the latest version of SQuAD, contains about 150k questions in total, and roughly half of them do not have an answer given the context, while the other half answerable questions are guaranteed to have a span of text in the given paragraphs as their answers. We will use this dataset to measure the performances of models developed in our wok.

With the development of datasets, more deep learning methods have shown their potentials in this task. Specifically, the attention mechanism brings significant performance improvement. Successful models include BiDAF [3], Dynamic Coattention Network [4], QANet [5], and Performer [6]. In this work, we will explore an ensemble model which brings together the key ideas from these three papers. Namely, we will use a simple version of BiDAF as our baseline model, and borrow the Coattention idea from DCN to modify the architecture and achieve a better performance. We also re-implement the QANet from scratch and perform some minor hyperparameter searching. Both the Coattention Layer and QANet model bring an approximately 10% improvement to the F1 score from around 58 to around 65. To expedite the QANet without losing much performance, we also replace the attention layer of QANet by adopting the FAVOR idea from Performer. Finally, we use a straightforward ensemble model (a voting scheme) to achieve F1 scores of 68.35 on dev set and 65.69 on test set.

3 Related Work

Prior to 2016, the attention mechanisms applied to end-to-end machine comprehension and question answering models usually involve a fixed-size vector on the summarization of a small context paragraph with a uni-directional attention. In 2017, Seo et al. [3] introduced the Bi-Directional Attention Flow (BiDAF) network, which uses bi-directional attention flow mechanism (i.e., query-to-context and context-to-query attention) to obtain a query-aware context representation that allows the unfixed attended vectors computed at every time step flowing into subsequent layers while each attention is computed independently based on the query and the context paragraph only at the current time step. This mechanism reduces the information by making the attention layer not to summarize the context to be a fixed-size vector, at the same time, the memory-less attention computed at each time step helps to enforce the attention layer focusing on the bidirectional interaction of context and query at the current time step and not attending to some false information at the previous time steps. Earlier in the same year, Xiong et al. [4] suggests the Dynamic Coattention Network (DCN) which consists of a coattentive encoder that also utilizes a bi-directioanl attention to capture the interactions between the query and the context, as well as a dynamic pointing decoder that iteratively predicts the start and end points of the answer span, which helps the model to recover from the initial incorrect answer spans.

In 2018, Yu et al.[5] takes advantage of the newly-introduced self-attention by Vaswani et al. [7] and proposes a new Q&A architecture, QANet, which eschews the previous recurrent networks and exclusively uses convolution for the local structures of the texts and self-attention for global interactions between each pair of words, while still preserving the query-aware context representation (i.e., context query attention) from BiDAF. The model outperforms the existed models on the SQuAD 1.0 dataset and also boosts the speed on training and evaluation in comparison to BiDAF. However, self-attention requires quadratic cost $O(T^2)$ on each pair of the words where T represents the sequence length, there are different techniques such as low-rank matrix representation for key $\boldsymbol{K}$ and value $\boldsymbol{V}$ by projection (Linformer) [8], local-sensitive hashing and reversible residual layers (Reformer) [9], sparse attention mechanism by combining of different types of attentions (BigBird) [10], etc. In 2020, Choromanski et al. [6] provides a practical but accurate approximation of regular full-rank self-attention matrix in linear time without relying on any priors such as sparsity by using a novel *Fast Attention Via positive Orthogonal Random features* approach (FAVOR+). One can incorporate the linear-cost (or subquadratic-cost) self-attention approaches with QANet to further enhance the speed.

4 Approach

In this section we detail the models we implement to solve the question answering task. Our ensemble method has two components: an improved BiDAF model with Dynamic Coattention Layer and a

QANet model expedited by an idea adopted from Performer to approximate the quadratic attention computation in linear time.

4.1 Baseline Model

We use a vanilla BiDAF model as our baseline. In general, a BiDAF has an Embedding Layer, an RNN Encoder Layer, a bidirectional Attention Layer, another Encoder Layer, and a final Output Layer. Note that unlike the model specified in the paper, we do not include a character embedding layer, which will be described in the next section.

4.2 Character Level Embedding

Character embedding layer is used for both models (BiDAF and QANet) we implement. It is responsible for mapping each word to a 200-dimensional space to explore the internal word structure (*morphology*) and handle out-of-vocabulary words.

We implement character-level embedding of each word using Convolutional Neural Networks (CNN). The original randomly initialized character embedding is passed through a 2D CNN with kernel size $k = (1, 3)$ which outputs a vector $\boldsymbol{x}_c \in \mathbb{R}^{200}$. It's then concatenated with the pretrained word embedding and passed through a high way network to generate the final embedding vector.

4.3 Dynamic Coattention

In this section we discuss the Coattention Layer [4], which serves as a substitution to the original Attention Layer in our baseline model. This idea may bring improvement since it provides a more complicated attention mechanism, i.e., it also attends over representations that are attentions themselves.

Let $\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots, \boldsymbol{c}_N \in \mathbb{R}^H$ denote the context hidden states, and $\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_M \in \mathbb{R}^H$ denote the question hidden states. Here we adopt the idea proposed by Merity et al. [11] to also append sentinel vectors $\boldsymbol{c}_\emptyset$ and $\boldsymbol{q}_\emptyset$ to context and question states respectively, which allows us to attend to none of the hidden states. Note that we now have $N + 1$ context hidden states and $M + 1$ question hidden states. We denote the hidden state matrix $\boldsymbol{D} = [\boldsymbol{c}_1 \quad \boldsymbol{c}_2 \quad \ldots \quad \boldsymbol{c}_N \quad \boldsymbol{c}_\emptyset] \in \mathbb{R}^{H\times(N+1)}$ and $\boldsymbol{Q} = [\boldsymbol{q}_1 \quad \boldsymbol{q}_2 \quad \ldots \quad \boldsymbol{q}_M \quad \boldsymbol{q}_\emptyset] \in \mathbb{R}^{H\times(M+1)}$.

To allow for variation between question encoding space and document encoding space, we first apply a non-linear projection layer to question encoding

$$\boldsymbol{Q}' = \tanh\left(\boldsymbol{W}\boldsymbol{Q} + \boldsymbol{b}\right)$$

where $\boldsymbol{W} \in \mathbb{R}^{H\times H}$ and $\boldsymbol{b} \in \mathbb{R}^H$. Then we use the usual inner-product attention to compute the similarity matrix $\boldsymbol{L} = \boldsymbol{D}^\top \boldsymbol{Q}$. As in BiDAF, we normalize over $\boldsymbol{L}$ row-wise to produce attention weights $\boldsymbol{A_Q}$ for each word in the question, and column-wise to produce attention weights $\boldsymbol{A_D}$ for each word in the document:

$$\boldsymbol{A_Q} = \text{softmax}(\boldsymbol{L}) \in \mathbb{R}^{(N+1)\times(M+1)}, \boldsymbol{A_D} = \text{softmax}(\boldsymbol{L}^\top) \in \mathbb{R}^{(M+1)\times(N+1)},$$

then we get Context-to-Question (C2Q) Attention matrix $\boldsymbol{D}\boldsymbol{A_Q} \in \mathbb{R}^{H\times(M+1)}$ and Question-to-Context (Q2C) matrix $\boldsymbol{Q}\boldsymbol{A_D} \in \mathbb{R}^{H\times(N+1)}$. Finally, Coattention Layer gives the second-level attention output by using the C2Q distributions as weight to take the sum of Q2C attention outputs. Namely we have $\boldsymbol{S} = \boldsymbol{D}\boldsymbol{A_Q}\boldsymbol{A_D} \in \mathbb{R}^{H\times(m+1)}$. Let $\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_N$ denote the first N columns of $\boldsymbol{S}$. Note that the original paper recommends concatenating $\boldsymbol{s}_i$ with columns C2Q attention outputs, and feed the concatenated sequence through a bidirectional LSTM. However, in our own experiments we find such architecture is not only time-consuming but also bring little, if any, improvement. Therefore here we replace the bidirectional by a simple linear layer. It turns out the resulting performance is still good as shown in Section 5.

4.4 QANet

Our QANet implementation is mainly based on the original paper by Yu et al. [5] except for some small modifications. QANet shares many similar components with BiDAF: they both start with word

and character embedding layer; they use the same bi-directional context-query attention architecture; and the output layers apply linear transform and masked softmax to the output of model encoders to produce start and end position predictions. However their encoding mechanisms are different: BiDAF uses recurrent technique while QANet uses convolution and self-attention exclusively as building blocks of encoders.

Embedding Layer QANet adapts pretrained 300-dimensional GloVe word vectors as fixed word embedding and uses the same 2D convolution + ReLU + maxpooling as we do in BiDAF to obtain 200-dimensional learnable character embedding. Two embeddings are concatenated together before fed into a two-layer highway network introduced by Srivastava et al. [12] and finally linearly projected from $d = 500$ to $d_{\text{model}} = 128$.

Embedding Encoder Layer An encoder block is in the form [positional encoding + (layernorm + convolution layer) $\times$ n + layernorm + self-attention + layernorm + feed-forward-layer] with residual connections between each pair of layernorms. We use learnable positional encoding

$$\text{Encode}(\boldsymbol{x}) = \text{x} + \text{pos}[:, \text{x.size}(1), \text{x.size}(2)]$$

where pos is learnable parameters. QANet uses depthwise separable convolutions (Kaiser et al. [13]) for convolution layer:

$$\text{PointwiseConv}(\boldsymbol{W}, \boldsymbol{y})_{(i,j,k)} = \sum_{m}^{M} \boldsymbol{W}^{(k)}[m]\boldsymbol{y}[i,j,m]$$

$$\text{DepthwiseConv}(\boldsymbol{W}, \boldsymbol{y})_{(i,j,k)} = \sum_{i,j}^{M,N} \boldsymbol{W}^{(k)}[i,j] * \boldsymbol{y}[i,j,k]$$

$$\text{SepConv}(\boldsymbol{W}_p, \boldsymbol{W}_d, \boldsymbol{y}) = \text{PointwiseConv}(\boldsymbol{W}_p, \text{DepthwiseConv}(\boldsymbol{W}_d, \boldsymbol{y}))$$

and these can be easily translated into pytorch functions: conv1d with kernel_size=1 for pointwise convolution and conv1d with (kernel_size=7, padding=3 and groups=128) for depthwise convolution. Self attention is adopted from multi-head attention by ([7])

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_k}}\right)\boldsymbol{V}$$

$$\text{head}_i = \text{attention}(\boldsymbol{Q}\boldsymbol{W}_i^{\boldsymbol{Q}}, \boldsymbol{K}\boldsymbol{W}_i^{\boldsymbol{K}}, \boldsymbol{V}\boldsymbol{W}_i^{\boldsymbol{V}}), \quad i = 1, \ldots, h$$

$$\text{MultiHead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{concat}(\text{head}_1, \ldots, \text{head}_h)\boldsymbol{W}^{\boldsymbol{O}}$$

where projections are parameters $\boldsymbol{W}_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $\boldsymbol{W}_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $\boldsymbol{W}_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$, $\boldsymbol{W}^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$. In this project we use $h = 8$, $d_k = d_v = d_{\text{model}}/h = 16$. Feedforward network has the form

$$f(x) = \text{ReLU}(\boldsymbol{x}\boldsymbol{W}_1 + \boldsymbol{b}_1)\boldsymbol{W}_2 + \boldsymbol{b}_2$$

Residual connections wrap pairs of layernorm and nonlinear layer inside

$$\text{Residual}(f, \boldsymbol{x}) = f(\text{layernorm}(\boldsymbol{x})) + \boldsymbol{x}$$

Embedding Encode Layer contains **one** encoder block.

Context-Query Attention Layer Attention mechanism in QANet is the same as in BiDAF. Note that since the output has shape (seq_len, $4d_{\text{model}}$), we need a linear layer to project output back to d_{model}.

Model Encoder Layer Model encoder uses the same encoder block as embedding encoder layer but with some different parameters. We use 7 consecutive encoder block instead of 1. Kernel size of depth-wise convolution is 5. Each block contains 2 convolution layers instead of 4. We propagate through the 7 blocks three times, and denote them as $\boldsymbol{M}_1$, $\boldsymbol{M}_2$ and $\boldsymbol{M}_3$.

Output Layer Each example in SQuAD is labeled with a span in the context containing the answer. We predict the probability of each position being the start or end of the answer.

$$\boldsymbol{p}_1 = \text{softmax}\left(\boldsymbol{W}_1[\boldsymbol{M}_1; \boldsymbol{M}_2]\right), \quad \boldsymbol{p}_2 = \text{softmax}\left(\boldsymbol{W}_2[\boldsymbol{M}_1; \boldsymbol{M}_3]\right)$$

4.5 Performer

We discuss the mechanism of Performer [6] in this section. As a variation of the vanilla self-attention, we require $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{L\times d}$ from self-attention component. We first generate m samples $\omega_1, \ldots, \omega_m \overset{\text{i.i.d.}}{\sim} \mathcal{D}$ for $\omega_1, \ldots, \omega_m \in \mathbb{R}^d$ from an isotropic distribution (usually $\mathcal{D} = \mathcal{N}(0, \boldsymbol{I}_d)$). Then we use Gram-Schmidt process to make $\omega_1, \ldots, \omega_m$ exactly orthogonal such that the variance of softmax/Gaussian kernel estimators is reduced for large d. We then construct the kernel function $\phi \in \mathbb{R}^d \to \mathbb{R}_+^{ml}$ for function $f_1, \ldots, f_l : \mathbb{R} \to \mathbb{R}$ and $h : \mathbb{R}^d \to \mathbb{R}$ such that

$$\phi(\boldsymbol{x}) = \frac{h(\boldsymbol{x})}{\sqrt{m}}(f_1(\omega_1^\top \boldsymbol{x}), \ldots, f_1(\omega_m^\top \boldsymbol{x}), f_l(\omega_1^\top \boldsymbol{x}), \ldots, f_l(\omega_m^\top \boldsymbol{x})).$$

In specific, the softmax-kernel admits a positive random feature map approximation with either of the following conditions:

$$\begin{cases} h(\boldsymbol{x}) = \exp(-\frac{||\boldsymbol{x}||^2}{2}), l = 1, f_1(u) = \exp(u), \\ h(\boldsymbol{x}) = \frac{1}{\sqrt{2}}\exp(-\frac{||\boldsymbol{x}||^2}{2}), l = 2, f_1(u) = \exp(u), f_2(u) = \exp(-u). \end{cases}$$

Then let $r = ml$ and compute $\boldsymbol{Q}', \boldsymbol{K}' \in \mathbb{R}_+^{L\times r}$ where without loss generality, consider $\boldsymbol{Q}$ and $\boldsymbol{Q}'$:

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{q}_1^\top \\ \boldsymbol{q}_2^\top \\ \vdots \\ \boldsymbol{q}_L^\top \end{bmatrix}, \boldsymbol{Q}' = \begin{bmatrix} \phi(\boldsymbol{q}_1^\top)^\top \\ \phi(\boldsymbol{q}_2^\top)^\top \\ \vdots \\ \phi(\boldsymbol{q}_L^\top)^\top \end{bmatrix}.$$

Finally, construct $\hat{\boldsymbol{D}} = \text{diag}(\boldsymbol{Q}'((\boldsymbol{K}')^\top \boldsymbol{1}_L)) \in \mathbb{R}^{L\times L}$ as a normalization matrix and

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) \overset{\mathbb{E}}{=} \hat{\boldsymbol{D}}^{-1}(\boldsymbol{Q}'((\boldsymbol{K}')^\top \boldsymbol{V})).$$

(Please refer to the whole derivations in [6]. Intuitively, for the softmax operation in self-attention, we need to compute $\exp(\boldsymbol{q}^\top \boldsymbol{k})$ and

$$\begin{aligned} \exp(\boldsymbol{q}^\top \boldsymbol{k}) &= \exp(-\frac{||\boldsymbol{q}||^2 + ||\boldsymbol{k}||^2}{2})\exp(\frac{||\boldsymbol{q}+\boldsymbol{k}||^2}{2}) \\ &= \exp(-\frac{||\boldsymbol{q}||^2 + ||\boldsymbol{k}||^2}{2})\mathbb{E}_{\omega\sim\mathcal{N}(0,\boldsymbol{I}_d)}\left[\exp(\omega^\top \boldsymbol{q})\exp(\omega^\top \boldsymbol{k})\right] \\ &\approx \frac{\exp(-\frac{||\boldsymbol{q}||^2}{2})}{\sqrt{m}}\begin{pmatrix} \exp(\omega_1^\top \boldsymbol{q}) \\ \exp(\omega_2^\top \boldsymbol{q}) \\ \vdots \\ \exp(\omega_m^\top \boldsymbol{q}) \end{pmatrix} \cdot \frac{\exp(-\frac{||\boldsymbol{k}||^2}{2})}{\sqrt{m}}\begin{pmatrix} \exp(\omega_1^\top \boldsymbol{k}) \\ \exp(\omega_2^\top \boldsymbol{k}) \\ \vdots \\ \exp(\omega_m^\top \boldsymbol{k}) \end{pmatrix}, \end{aligned}$$

for large sample size m.)

Notice that if the cost of the last computation is dominant, then $\boldsymbol{A} = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d}}) \in \mathbb{R}^{L\times L}$ and $\boldsymbol{A}\boldsymbol{V}$ takes $O(L^2 d)$ operations, while $\boldsymbol{Q}'((\boldsymbol{K}')^\top \boldsymbol{V})$ takes $O(Lrd + Lrd + L) = O(Lrd)$. The time complexity is reduced when $r \ll L$.

5 Experiments

5.1 Data

We use Stanford Question Answering Dataset (SQuAD) 2.0 dataset [14] for our project. For the train set, around 68% of the questions are answerable, while for the dev set, only around 48% of the questions are answerable.

5.2 Evaluation method

We primarily evaluate the model performance by F1 score as F1 score is the performance metric for rank submissions in the leaderboard and we have known dev F1 score (~ 58) for the baseline model. We also leverage Exact Match (EM) score and AvNA as reference metrics to compare models with different variants.

5.3 Experimental details

5.3.1 Common Training hyperparameters

Default settings are the same as in the starter code: optimizer = Adadelta, learning_rate = 0.5, weight_decay = 0, dropout_prob = 0.2, ema_decay = 0.999, batch_size = 32, hidden_size = 128, char_dim = 200

5.3.2 QANet

In QANet we use learned parameters to achieve positional embedding. So during training we set maximum sequence length to 800 to account for potential long sequences in test set.

5.3.3 Emsemble

For QANet model with Performer, we have tested with several reduced dimensions $m \leq d_{model}/h = 16$ and find that the current re-implementation has a relatively large overhead on Gram-Schmidt process, the computation time is not comparable to direct computation even with small m. We did not run the model with full number of iterations as the cost becomes unaffordable to us. On the other hand, the performance of QANet with Performer is inferior to the performance of vanilla QANet because it originally intends to sacrifice the accuracy of computing the exact self-attention with a promisingly faster speed. Therefore, we decide not to include QANet with Performer model in the ensemble model.

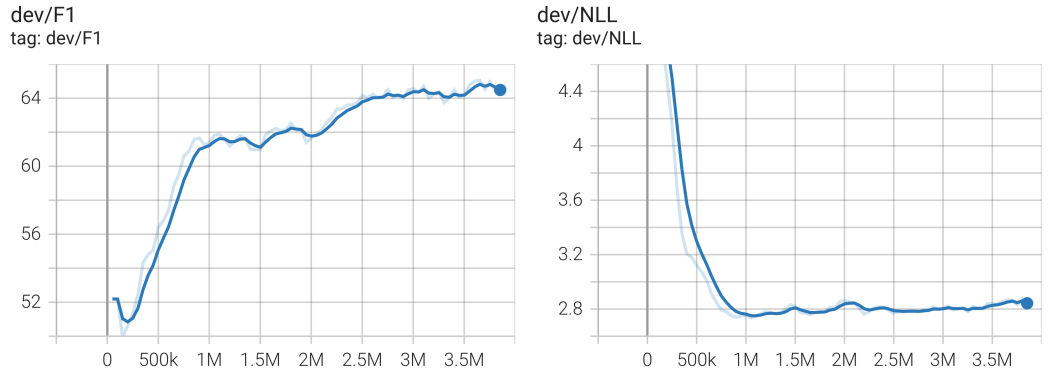

Figure 2: The dev plots of BiDAF with DCN.

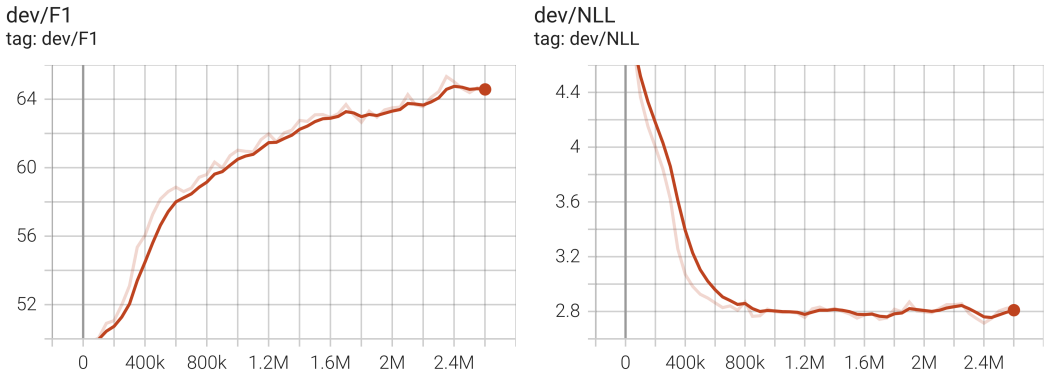

Figure 3: The dev plots of QANet with self attentioner.

5.4 Results

Models scores on dev and test leaderboards are listed in 1. The leaderboard name is XXL.

Model	EM	F1
BiDAF with DCN (dev)	61.855	65.309
BiDAF with DCN (test)	60.845	64.600
QANet (dev)	62.309	65.595
Ensemble (test)	62.688	65.690

Table 1: Experiment Results

6 Analysis

In the following plots 4, we can notice that for the ensemble model evaluating on the dev set, there are around 9.4% of the answerable examples that are classified wrongly as unanswerable questions (False Negative) and around 16.8% of the unanswerable examples that are classified wrongly as answerable questions (False Positive) which is nearly 2 times of the false negative rate. It shows that one of the main goals that we have to work on is still abstaining the model from answering unanswerable questions. Within True Positive Examples, around 35.5% examples have issues on the entire different answer spans (described as disjoint in the plots), which contributes around 92.7% of the wrong answers. This suggests that we may focus on advanced technique of answer pointer to further improve our model.

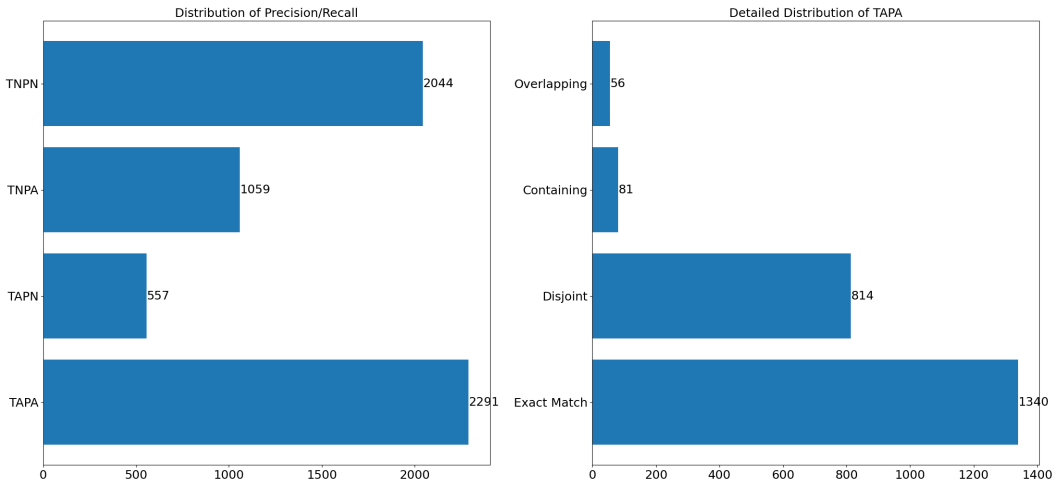

Figure 4: Precision and Recall Barplot of Result for the Ensemble Model at the Dev Set.

7 Conclusion

In this project, we start from the baseline model BiDAF by adding character-level embeddings and then replace the original bi-directional attention layer by a re-implemented Coattention Layer introduced in DCN model, which enhances the performance from the baseline model. After that, we shift gears to re-implement the QANet model and achieve a comparative performance as the modified BiDAF model. Additionally, we endeavor to re-implement FAVOR+ approach of the Performer model from scratch. At last, we construct an ensemble model with BiDAF and vanilla QANet model to maximize the final performance. In the future, we may consider changing our kernel generation mechanism for the Performer to enhance the speed or alternatively, applying data augmentation technique to enrich the training data for performance or exploring some advanced pointer decoder techniques to more effectively recover the answer span and thus, gradually improving the model.

References

[1] 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.

[2] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

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

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

[5] 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.

[6] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[8] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

[9] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[10] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33:17283–17297, 2020.

[11] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

[12] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

[13] Lukasz Kaiser, Aidan N Gomez, and Francois Chollet. Depthwise separable convolutions for neural machine translation. *arXiv preprint arXiv:1706.03059*, 2017.

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