IID SQuAD track: Comparing QANet with BiDAF

Stanford CS224N Default Project

Bingqi Sun
Department of Computer Science
Stanford University
sbq@stanford.edu

Ruochen(Chloe) Liu
Department of Computer Science
Stanford University
ruochenl@stanford.edu

Shanduojiao Jiang
Department of Computer Science
Stanford University
sj99@stanford.edu

Abstract

In the task of reading comprehension or question answering, a model will be given a paragraph and a question about that paragraph as inputs, and the goal is to answer the question correctly. This is an interesting task as it could be viewed as how well a model can "understand" text. Current end-to-end machine reading and question answering models such as BiDAF [1] and QANet [2] can achieve relatively good results. In this work, we (1) improved on the given BiDAF-based model, (2) implemented QANet, (3) explored data augmentation and model ensembling to further improve the performance of the QA system, and (4) investigated and analyzed the difference between BiDAF and QANet. For our final ensemble model, we achieved an F1 score of 70.09, an EM score of 67.50, and an AvNA score of 74.36 on dev set - all three of these metrics showed around 10% improvement compared to the baseline model. We released the source code of our project on GitHub.

1 Key Information to include

TA Mentor: Michihiro Yasunaga

2 Introduction

Question answering has been one of the most active research areas in NLP because of its wide applications such as search engine optimization. Earlier models were only able to provide answers in the form of a document or a segment of text selected from a vast amount of available documents, which is not concise nor precise enough as the users would still have to read through the documents to locate specific answers to their questions. With the emergence of QA datasets such as SQuAD or SQuAD 2.0 (which includes unanswerable questions compared to SQuAD)[3], recent models can select the span of text in the paragraph that answers the question.

One of the most successful recent models is Bidirectional Attention Flow model (BiDAF), which employs a recurrent model to process sequential inputs, and an attention component to cope with long term interactions. Another one of these models is QANet, which eliminates the recurrent component of BiDAF and combines the local convolution with global self attention.

Given the different architecture designs of BiDAF and QANet, we suspected that the two models have different strengths and weaknesses. Thus, in addition to improving the given BiDAF-based model, and to re-implementing QANet from scratch, we investigated and analyzed the different performances

of the two models for different scenarios, and proposed an ensemble model that improved the baseline performance by 10% because of its ability to utilize the strengths of each individual model.

3 Related Work

Prior to the deployment of deep neural networks, research in the task of question answering primarily focused on the linguistic resources, such as part-of-speech tagging, parsing, named entity extraction, and semantic relations. For example, Brill et al.[4] explored the power of surface text patterns. Specifically, a tagged corpus is built from the internet in a bootstrapping process by providing a few hand-crafted examples of each question type. Patterns are then automatically extracted from the returned documents and standardized, which are applied to find answers to new questions.

However, systems that rely solely on linguistic resources have limited ability to narrow down the content. One of the key factors to the advancement has been the use of neural attention mechanisms, which enables the system to focus on a targeted area within a context paragraph. Xiong et al.[5] proposed several improvements to dynamic memory network's memory and input modules, and was able to improve the performance on both visual question answering dataset and text question answering dataset.

With more attention being drawn to machine comprehension and question answering, BiDAF and QANet were introduced, as discussed in the previous section. Later on, more powerful transformer-based models such as BERT[6], which stands for Bidirectional Encoder Representations from Transformers, achieved state-of-the-art results for the SQuAD 2.0 dataset. BERT is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers, and thus it can be fine-tuned with just one additional output layer to show great results for a wide range of tasks. However, due to the requirement of this project, we are not going to use BERT.

4 Approach

In this section, we describe the following five stages of our project in detail: (1) Baseline BiDAF model, (2) Improving the BiDAF model with character embedding, (3) QANet implementation, (4) Data Augmentation, and (5) Model ensembling.

4.1 Baseline BiDAF Model

We will use the given BiDAF-based model as our baseline for the project. The model architecture is discussed in the project handout as well as the original BiDAF paper[1].

4.2 Improving the BiDAF Model with Character Embedding

The given baseline model uses only pre-trained GloVe embeddings to get the vector representation of words in the Query and the Context. Even if GloVe covers millions of words, it is still possible to encounter a word in our training set that is not present in GloVe's vocabulary. GloVe deals with these words by simply assigning them random vector values, which could be confusing to our model. Therefore, to better handle these out-of-vocabulary words, we added character-level embeddings. Following the original BiDAF paper[1], we obtained the characterlevel embedding of each word using one-dimensional Convolutional Neural Network (CNN). Characters are embedded into vectors and fed into the CNN. The outputs of the CNN are max-pooled over the entire width to obtain a fixed-size vector for each word. The character-level embeddings are then concatenated with the word vectors and passed to a two-layer Highway Network [7]. In our QANet implementation detailed below, we adopted this techniques again to obtain the character-level embeddings.

4.3 QANet implementation

We re-implemented QANet from scratch using PyTorch. The model architecture is shown in Figure 1. Specifically, there are five layers in this model:

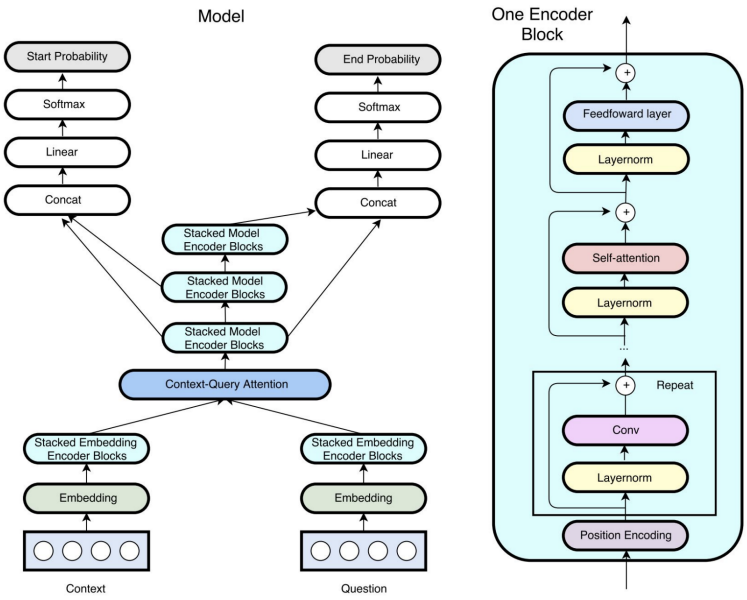

Figure 1: QANet Model Architecture and Encoder Block Zoom-in

- **Input Embedding Layer**: In this layer, we obtain the embedding for each word by concatenating its word embedding (which is fixed during training and initialized from the p_1 dimensional pre-trained GloVe word vectors [8]) and character embedding (which is represented as a trainable vector of dimension $p2$). The output of a given word x from this layer would be $[x_w; x_c] \in R^{p_1+p_2}$, where x_w and x_c are the word embedding and the convolution output of character embedding of x respectively.
- **Embedding Encoder Layer**: This layer is a stack of the following encoder block: [convolution-layer + self-attention-layer + feed-forward-layer] as shown on the right side of Figure 1. We followed the original paper and used depth-wise separable convolutions (which deals not just with the spatial dimensions, but with the depth dimension as well) in an attempt to make the model memory efficient and more generalizable.[9] We used the multi-head attention mechanism in the self-attention-layer such that for each position in the input (query), it computes a weighted sum of all positions (keys) in the input based on the dot-product similarity between the query and key [10]. Each of these basic operations is placed inside a residual block.
- **Attention Layer**: This is a standard context-query attention layer in almost every reading comprehension models that signifies which query words are most relevant to each context word. It uses a similarity matrix $S \in R^{n \times m}$ computed from context C and query Q. Then, softmax function is applied to normalize each row of S to obtain $\overline{S}$. We then computed the context-to-query attention as $A = \overline{S} \cdot Q^T \in R^{n \times d}$. Following BiDAF, we also used query-to-context attention which signifies which context words have the closest similarity to one of the query words and are hence critical for answering the query. We take the softmax of the columns of get S to get $\overline{\overline{S}}$. Then we multiply $\overline{S}$ into $\overline{\overline{S}}$ and use the result to take weighted sums of the hidden states c_j to get the query-to-context attention output.
- **Model Encoder Layer**: The input of this layer at each position is $[c, a, c \odot a, c \odot b]$, where a and b represent the context-to-query and query-to-context attention output respectively, and $\odot$ represents element-wise multiplication. The weights are shared between each of the 3 repetitions of the model encoder.
- **Output Layer**: In this layer, the probabilities of the starting and ending position are modeled as $p^1 = softmax(W_1[M_0; M_1])$, and $p^2 = softmax(W_2[M_0; M_2])$ where W_1 and W_2 are two trainable variables and M_0, M_1, and M_2 are the outputs of the three model encoders from bottom to top. The score of the span is calculated as the product of its start position

and end position probabilities. Let y_i^1 and y_i^2 be the groundtruth starting and ending position of example i, we define the objective function as: $L(\theta) = -\frac{1}{N}\sum_i^N [log(p_{y_i^1}^1) + log(p_{y_i^2}^2)]$.

At inference time, the predicted span (s, e) is chosen such that $p_s^1 p_e^2$ is maximized and $s \leq e$, where s is the starting index and e is the ending index of the answer, respectively.

4.4 Data Augmentation

In machine learning, more training data usually helps improve the model performance. Therefore, we adopted a simple data augmentation technique to enrich our training data. We used a pretrained multi-task t5-base model[11][12] which is trained for question answering and question generation tasks. Given an input text, the model is able to generate multiple questions simultaneously related to the input text and their corresponding answer as illustrated in Figure 2. This is an end-to-end question generation model which is capable of generating questions without providing the answers. With all the context available in the train set, we were able to generate around 80K new questions. Since we did not directly implement this model, we will not discuss the model architecture and training process in detail.

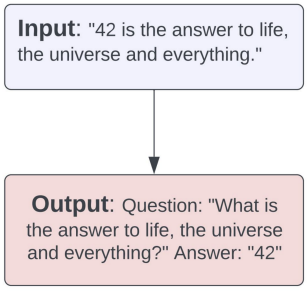

Figure 2: Data Augmentation Example

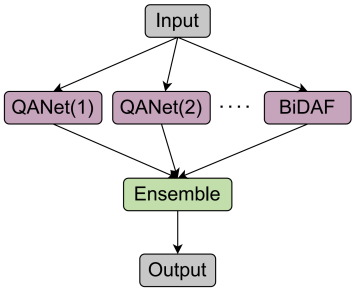

Figure 3: Ensemble Pipeline

4.5 Model ensembling

To further improve the model performance, we investigated in several ensemble techniques. Figure 3 illustrates our ensemble pipeline: an input is passed through different BiDAF and QANet models at the same time, and based on the rules we define, the ensemble model will output the final result. The first ensemble method we tried was averaging the softmax score p^1 and p^2 predicted by different models, and using the averaged predictions to identify the answer span when testing. We also explored the majority voting approach: for each question, we picked the answer that the majority of models agreed on, and broke ties by choosing the prediction from the model with the highest F1 score on the dev set. Finally, after conducting a comprehensive error analysis on BiDAF and QANet models, we came up with a list of heuristics when ensembling two models (BiDAF and QANet): (1) When the question is of "why" type, use the output from BiDAF model; (2) Choose the shorter answer as output between the two models; and (3) Mark question as "unanswerable" when either model outputs "unanswerable". We will discuss what each rule means and how we obtained the three rules in the analysis section in detail.

5 Experiments

5.1 Data

There are two parts for our dataset. The first part is the provided SQuAD 2.0 dataset, which consists of approximately 130K training data, 6K development data, and 6K test data. One data entry is a (question, context, answer) triple, where the question is either impossible to answer using the provided context (unanswerable), or is answerable using a chunk of text taken directly from the paragraph. The other part of our data comes from data augmentation, where we generated 80K more training data with the same format using the t-5 base model as discussed in the previous section. To

further understand the dataset, we investigated the distribution of question types, the distribution of question length, the distribution of context length, and the distribution of answer length as shown in Figure 4.

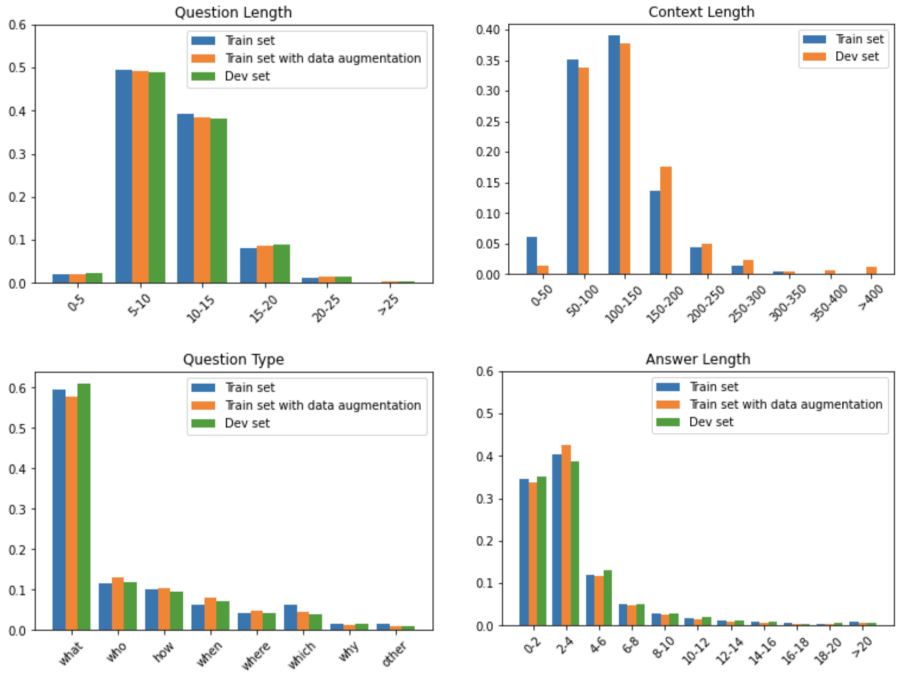

Figure 4: Data Distribution

Based on the distribution plots, we can see that: (1) The majority of the questions ask about "what". E.g. "What is the answer to life, the universe and everything?"; (2) The majority of the contexts, questions, and answers are of shorter length; (3) Data augmentation did not change the question length, question type, and answer length distribution too much.

5.2 Evaluation method

We use three metrics to evaluate the model. The first one is Exact Match (EM), which is a binary measure of whether the output matches the ground truth answer exactly. The second one is F1 score, which is the harmonic mean of precision and recall - calculated as $2 \times$ prediction $\times$ recall / (precision + recall). In addition to these two metrics, we introduced answer vs. no-answer (AvNA) score, which measures the classification accuracy of the model when only considering its answer (any span predicted) vs. no-answer predictions.

5.3 Experimental Details

We mostly followed the instructions of the BiDAF paper[1] in our training. We used an Adadelta optimizer, a learning rate of 0.5, and a batch size of 64. As for the QANet, we used an Adam optimizer, a learning rate of 0.001, and a batch size of 16. For both models, we used a dropout rate of 0.2 and an exponential moving average on all trainable variables with a decay rate of 0.999. We also experimented with a few variants of the each model by modifying the hidden size, number of attention heads, recurrent component architecture as explained in Table 1 and Table 2.

5.4 Results

Figure 3 summarizes the results that all of our models achieved on the dev set. We can see that the best performing model with an EM score of 67.50 and an F1 score of 70.09 comes from ensemble (average). Looking back, there are two major improvements to the performance of our QA system.

	Hidden Size	# Head	Char.Emb
QANet(1)	128	1	Yes
QANet(2)	128	4	Yes
QANet(3)	128	8	Yes
QANet(4)	256	1	Yes

Table 1: QANet Variants

	Char.Emb	RNN
BiDAF(1)	No	LSTM
BiDAF(2)	Yes	LSTM
BiDAF(3)	Yes	GRU

Table 2: BiDAF Variants

The first one is a result of changing the model from BiDAF to QANet, and the second one is a result of the deployment of model ensembling.

	F1	EM	AvNA
BiDAF(1)	60.42	56.74	67.7
BiDAF(2)	63.79	60.34	70.07
BiDAF(2) + Data Augmentation	62.75	59.15	69.55
BiDAF(3)	61.84	59.06	69.45
QANet(1)	68.46	64.86	74.47
QANet(2)	64.76	62.58	68.78
QANet(3)	61.55	59.22	67.17
QANet(4)	66.13	62.49	73.25
Ensemble (average)	70.09	67.50	74.36
Ensemble (majority vote)	54.56	50.89	N/A
Ensemble (heuristics)	69.96	67.36	N/A

Table 3: Comparison of models' performances on the dev set

We made the following discoveries based on Table 3: (1) data augmentation did not improve our system's performance; (2) BiDAF with character embeddings and LSTM as the recurrent component achieved the best results out of all the BiDAF variants; (3) QANet with hidden size = 128 and number of attention head = 1 achieved the best results out of all the QANet variants; (4) Ensemble (average) achieved the best results out of all the ensemble model variants.

For the test set, we achieved 65.56 for the EM, and 68.28 for the F1. The differences in the scores between the dev set and the test set could be attributed to slight differences in data distribution that will be elaborated in the next section.

6 Analysis

6.1 Dataset Analysis

In table 3, one of the interesting findings is that adding data augmentation did not improve our model's performance on development set. Therefore, we plotted the data distribution before and after data augmentation in Figure 5 to investigate what would have caused the loss of performance. We can see that the percentage of unanswerable questions was 33.38% in the original training set, and 52.11% in the original development set. However, since the current t-5 base model that we used for data augmentation was only able to generate answerable questions, the percentage of unanswerable questions in the new training set was diluted to 20.79% after the data augmentation, which may have caused the model to prefer answering the question even though the question might be unanswerable. Therefore, since the percentage of unanswerable questions is more than half in the dev set, the model trained on a biased augmented dataset could not perform well on the dev set.

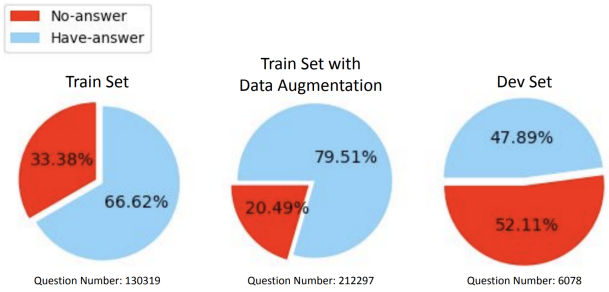

Figure 5: Data Distribution after Data Augmentation

6.2 QANet vs. BiDAF

6.2.1 Answerable vs. Unanswerable predictions

One feature of SQuAD 2.0 is that it has questions that are impossible to answer based on the given context. We want to assess our models' ability to identify these unanswerable questions.

Table 4 shows the performance comparison between BiDAF and QANet on the answerable and unanswerable questions. We can see that for questions that QANet predicts as answerable, $\frac{37.42\%}{37.42\%+15.09\%} \approx 71.26\%$ are correct, while for BiDAF this number is $\frac{39.1\%}{39.1\%+21.17\%} \approx 64.87\%$. For questions that QANet predicts as unanswerable, $\frac{37.05\%}{37.05\%+10.44\%} \approx 78.02\%$ are actually unanswerable, while this number is $\frac{30.97\%}{8.75\%+30.97\%} \approx 77.9\%$ for BiDAF. QANet has less false positives while BiDAF is more likely to predict answers for unanswerable questions. We conclude that generally QANet performs better in terms of differentiating these two types of questions, which could lead to its higher F1 and EM scores on the dev set.

	QANet + Char.Emb Prediction		BiDAF + Char.Emb Prediction	
	Answerable	Unasnwerable	Answerable	Unanswerable
Answerable	37.42% (TP)	10.44% (FN)	39.1% (TP)	8.75% (FN)
Unanswerable	15.09% (FP)	37.05% (TN)	21.17% (FP)	30.97% (TN)

Table 4: Model Performance Comparison on Answerable vs. Unanswerable Questions

6.2.2 Performance by Question Type

We hope to gain a deeper understanding about how our models concretely work, so we calculated the F1, EM, and AvNA scores for each model on a subset of the dev set which only contains certain types of questions as shown in Figure 6. We noticed that among all the question types, both models showed the best performance on "When" questions and also achieved a reasonably good result on "Which" and "What" questions. This is probably because these questions are inherently easier for the models since their answers are often a single word which is easy to retrieve from the context. However, when it comes to "Why" questions, we witnessed a significant decrease in both models' performances. This is probably because "Why" questions require better "understanding" of the context rather than simple string pattern matching.

In terms the comparison between BiDAF and QANet, it is not surprising to see that QANet outperformed BiDAF on almost all types of questions. However, we discovered that on the hardest "Why" questions, BiDAF actually outperformed QANet. To find out the reason behind this result, we inspected many of the "Why" questions. We found that compared to QANet, BiDAF overall is more likely to predict an answer instead of predicting NAN (which is also shown in 6.2.1). For "Why" questions, since the questions are more complex and require logic reasoning, the three gold answers provided by human are often different. In many cases, the BiDAF prediction happens to match one of the three gold answers. We suspect that in this situation, predicting something is generally better then predicting nothing. This discovery helped us design the rules for our ensemble model.

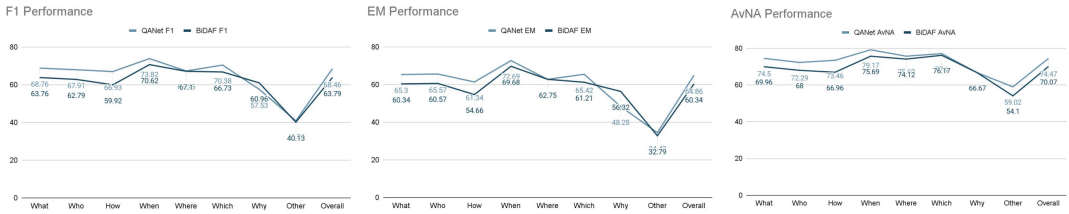

Figure 6: Model Performance Comparison for Different Question Types

6.2.3 Performance by Answer Length

We wondered if we could develop useful heuristics for ensemble models based on the model predictions. So we analyzed the relationship between metric scores and the length of predicted answers and Figure 7 shows the result. The x-axis represents answers categorized by their length, the bars show the total count in each categories, and the line chart denotes corresponding scores for each evaluation metric. We observe that both models achieved higher performance with shorter answers. This is expected as shorter answers make up for the majority of the data. Also, EM score is close to F1 score when the answer is short, but the gap becomes larger as the answer length increases, which makes sense if we consider how this two metrics are calculated: shorter answers are easier to be an exact match of the gold answer. When we compare the performance between QANet and BiDAF, there are also several findings worth mentioning: QANet achieved higher performance for all answer lengths, and QANet is more likely to predict shorter and unanswerable questions while BiDAF is more likely to predict longer answers.

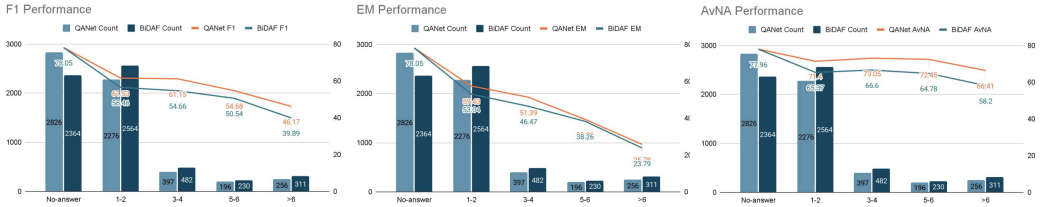

Figure 7: Model Performance Comparison for Different Answer Lengths

6.3 Ensemble Model

As mentioned earlier in the Approach section, we have tried different model ensembles, among which Ensemble (average) and Ensemble (heuristics) were able to boost the performance significantly. Recall that Ensemble (average) takes the average of the softmax score p^1 and p^2 predicted by different models, and uses the averaged predictions to identify the answer span for inference. We believe that the ensemble model was able to achieve good results because it aggregates information from all the individual models. Each single model might rely heavily on a subset of features, and it may suffer from high variance. Ensemble model can overcome these shortcomings.

Based on the comparison analysis between BiDAF and QANet above, we were able to derive the three rules mentioned earlier in the Approach section for our Ensemble (heuristics) model. One of the reasons why Ensemble (heuristics) model did not perform as well as Ensemble (average) could be that it did not generalize to unseen data as we defined our rules based on the available dev data. However, it still achieve second highest score on dev set comparing with all other models, so we would conclude that it still works well.

7 Conclusion

In summary, we implemented, evaluated and analyzed different models for the task of question answering (QA) on SQuAD 2.0 dataset. Specifically, we first improved the given BiDAF-based model by modifying its embedding and attention layers. Then, we used a pretrained model for data augmentation. Next, we re-implemented QANet from scratch and experimented with different hidden

layer sizes and number of heads. The adoption of QANet showed a great improvement on both the F1 and EM scores. After training both models, we did an extensive analysis between the two models. The analysis led us to derive the rules for our ensemble model, which showed another significant improvement for the model performance.

Overall, we were able to achieve relatively good results with an F1 score of 70.09, and an EM score of 67.50, but there is still room for improvement. One potential future work is to explore other data augmentation models that can generate unanswerable questions, or various question types.

Acknowledgement

We would like to thank the teaching team of CS224N for providing guidance, and Microsoft Azure for providing GPU access throughout this project.

References

[1] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2016.

[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. *CoRR*, abs/1804.09541, 2018.

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

[4] Deepak Ravichandran and Eduard Hovy. Learning surface text patterns for a question answering system. In *Proceedings of the 40th Annual meeting of the association for Computational Linguistics*, pages 41–47, 2002.

[5] Caiming Xiong, Stephen Merity, and Richard Socher. Dynamic memory networks for visual and textual question answering. *CoRR*, abs/1603.01417, 2016.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

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

[8] 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)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

[9] Francois Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

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

[11] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683, 2019.

[12] Suraj Patil. Question Generation using transformers.