Question Answering on SQuAD 2.0

Stanford CS224N Default Project, Option 3

Kevin Han
Department of Statistics
Stanford University
kevinwh@stanford.edu

Han Wu
Department of Statistics
Stanford University
hanwu71@stanford.edu

Yuqi Jin
Department of Computer Science
Stanford University
yuqijin@stanford.edu

Abstract

In this project, we implemented several variants of Question Answering models and compare their performance on SQuAD 2.0. We also implemented a data-augmentation method that enlarges the training set and compared the model trained on augmented dataset with the original model. We found that the QANet model trained on augmented dataset gives us the best performance in terms of both EM and F1 metrics. We achieved 65.21 F1 score and 61.80 EM score on dev set, 62.08 F1 score and 58.58 EM score on test set.

1 Introduction

Question answering is concerned with building systems that automatically answer questions posed by humans in a natural language [Wikipedia, 2004]. It has been a popular and crucial topic in the field of natural language processing for a long time, as researchers try to make this important human intelligence be automated by the machine. Despite the seemingly easy nature from a human point of view, it remains to be an open question and a complex challenge. With the rise of deep learning based models and more and more public available datasets, we have seen significant progress on this task. In this project we focus on one particular dataset, Stanford SQuAD2.0 [Rajpurkar et al., 2018]. SQuAD2.0 improves over SQuAD1.1 by adding roughly half of unanswerable questions, which gives the system extra layer of complication.

Attention mechanism is indispensable in recent successful models. In this project, we investigate two kinds of attention based models, BiDAF and QANet. We improve upon the baseline BiDAF by adding additional trainable character embeddings. Inspired by the work of [Wang et al., 2017], we also add a self-matching attention layer to effectively aggregate the information across the whole passage. We also implement the QANet architecture [Yu et al., 2018] which bypasses RNN entirely and uses convolution instead. We perform detailed ablation study and hyperparameter tuning. We also do data augmentation using back translation, which is introduced in [Yu et al., 2018], and adversarial questions, which aims to enlarge unanswerable questions.

2 Related Work

There are many existing models for question answering task in natural language processing literature. In BiDAF model [Seo et al., 2016], the authors use a context-query attention layer on top of embedding layers and encoder layers. R-net [Wang et al., 2017] is another successful model on question answering task. The key ingredient in their model is a self-matching attention layer. The intuition behind the self-matching attention layer is that we want the model to encode the information from the context when predicting the answer. Transformer [Vaswani et al., 2017] has already become a popular model in all kinds of natural language processing tasks now. QANet [Yu et al., 2018] utilizes transformer block in both encoder layer and modeling layer. The authors demonstrate significant speed up in their paper which gives the possibility to use an augmented training dataset for training. Another work in past CS224n offering [Dhoot and Gupta] tried to modify the embedding

by incorporating the cosine similarities between context words and query words. They showed the advantage of using this extra information.

3 Models

We implement our Question Answering models basing on BiDAF and QANet. We train them under different hyperparameters and also increase our amount of training data by data augmentation.

3.1 BiDAF based models

The baseline model, which is provided in the CS224N default project handout, is a slightly modified BiDAF model compared to the model in the original paper [Seo et al., 2016]. As most existing models, it consists of five layers: an embedding layer, an encoder Layer, an attention Layer, a modeling Layer and an output layer. The illustration is in figure 1. We detailed these and our modifications in the following subsections.

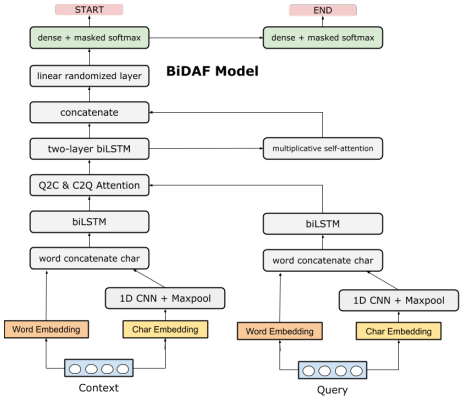

Figure 1: BiDAF Architecture

3.1.1 Embedding Layer

We combine both word-level embeddings and character-level embeddings for each word in this embedding layer. Given a word $w = [c_1, ...c_l] \in Z^l$, we use pretrained GloVe [Pennington et al., 2014] vectors to get its 300 dimensional word embedding, which is fixed during training. To get the character embedding, we initialize the character vectors to be the pretrained values, which gives us a $l \times H$ matrix for each word. We then pass this matrix into a convolution layer followed by max pooling to get a H-dimensional vector. We concatenate this with the word vectors followed by a projection and two-layer highway network [Srivastava et al., 2015]. The final output of this layer will be a 2H-dimensional vector for each word.

3.1.2 Encoder Layer

The encoder layer consists of a bi-directional LSTM to help us capture temporal dependencies, the output will be the concatenation of forward and backward hidden states. The output is a matrix $C \in R^{N \times 4H}$ and a matrix $Q \in R^{M \times 4H}$ obtained by the following equations,

$$c_i = \text{bi-LSTM}(h_i, W_i^C) \in R^{4H}$$
$$q_j = \text{bi-LSTM}(h_j, W_j^Q) \in R^{4H}$$

3.1.3 Context-Query Attention Layer

This layer is the core part of BiDAF model where we perform bi-directional attention between context and query. Given context hidden states $c_1, ...c_N \in R^{4H}$ and query hidden states $q_1, ...q_M \in R^{4H}$ we first compute the similarity matrix S.

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

Context-to-Query (C2Q) Attention We take row-wise softmax of S and get question aware context representation, i.e.

$$\bar{S}_{i,:} = \text{softmax}(S_{i,:}) \in R^M \ \forall i \in \{1, ..., N\}$$
$$a_i = \sum_{j=1}^{M} \bar{S}_{i,j} q_j \in R^{4H} \ \forall i \in \{1, ..., N\}$$

Question-to-Context (Q2C) Attention We first take column-wise softmax of S to get $\bar{\bar{S}}$ and multiply this to $\bar{S}$ to get the new weight matrix S', which is used to get Q2C attention output:

$$\bar{\bar{S}}_{:,j} = \text{softmax}(\bar{S}_{:,j}) \in R^M \ \forall j \in \{1, ..., M\}$$
$$S' = \bar{S}\bar{\bar{S}}^T \in R^{N \times N}$$
$$b_i = \sum_{j=1}^{N} S'_{i,j} c_j \in R^{4H} \ \forall i \in \{1, ..., N\}$$

The final output layer will be

$$g_i = [c_i; a_i; c_i \circ a_i; c_i \circ b_i] \in R^{16H} \ \forall i \in \{1, ...N\}$$

3.1.4 Modeling Layer

The modeling layer is again a bi-directional LSTM which integrates between temporal information in the context condition on question. However we reduce dimension to 4H.

3.1.5 Self-Matching Attention Layer

On top of the modeling layer, we add a self-attention layer mentioned in R-net [Wang et al., 2017]. We implemented this layer because we want to emphasize the related part in the context regarding the query. We deleted the bi-RNN structure and replaced the additive attention to multiplicative attention due to memory constraint and speed concern. We kept the gate structure and added a linear projection layer with bias before output to make the dimension match. Specifically, given the question aware context representations $v_t \in R^{4H}, t = 1, ..., N$, we get an attention based vector c_t by:

$$s_j^t = v_j^T W v_t$$
$$a_i^t = \exp(s_i^t) / \sum_{j=1}^{N} \exp(s_j^t)$$
$$c_t = \sum_{i=1}^{N} a_i^t v_i$$

The output will be

$$m_t = \text{sigmoid}(W[v_t, c_t])$$
$$m_t = W'(m_t \circ [v_t, c_t]) \in R^{4H}$$

3.1.6 Output Layer

In this layer we further aggregate all the information. Suppose we have output matrix G from the BiDAF attention layer and matrix M from the self-matching attention layer. We pass M into a bi-directional LSTM to get a matrix M' and then we predict p_{start} and p_{end} by

$$p_{start} = \text{softmax}(W_{start}[G; M]) \quad p_{end} = \text{softmax}(W_{end}[G; M'])$$

3.2 QANet

QANet shares some components with BiDAF. The embedding layer and context-query attention layer stay the same. The main differences are how we handle encoder layer and modeling layer. This is where the transformer block comes in. We replace the recurrent architecture in BiDAF by convolution-layer and transformer block. Specifically, both the encoder layer and modeling layer is a stack of the basic encoder blocks shown in the right side of Figure 2. For an overview of the QANet model, please refer to the left side of Figure 2. Next, we introduce the key ingredients in one single encoder block.

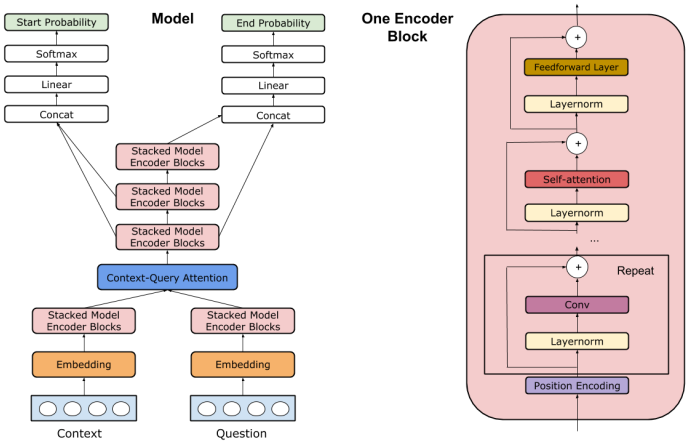

Figure 2: Overview of QANet Architecture

3.2.1 Positional Encoding

In [Vaswani et al., 2017], they add an positional encoding to the input embeddings to encode the information of the position of tokens in the sentence. We use the same positional encoding as shown below:

$$\text{PE}_{(pos,2i)} = sin(pos/10000^{2i/d_{model}}) \tag{1}$$

$$\text{PE}_{(pos,2i+1)} = cos(pos/10000^{2i/d_{model}}) \tag{2}$$

3.2.2 Depthwise separable convolutions

The idea of depthwise separable convolutions come from [Kaiser et al., 2018]. It consists of a depthwise convolution, i.e. a spatial convolution performed independently over every channel of an input, followed by a pointwise convolution, i.e. a regular convolution with kernel size 1. According to [Kaiser et al., 2018], this is a powerful simplification to a regular 2D or 3D convolution under the assumption that the 2D or 3D inputs that convolutions operate on will feature both fairly independent channels and highly correlated spatial locations.

3.2.3 Layer Normalization

This is first introduced in [Ba et al., 2016]. One of the challenges of deep learning is that the gradients with respect to the weights in one layer are highly dependent on the outputs in the previous layer. Batch normalization is designed to fix this problem. However, it is hard to apply to recurrent neural networks [Ba et al., 2016]. Layer normalization was designed to overcome the drawback of batch normalization. Suppose H is the number of hidden units of a layer and l denotes one layer in our model. The layer normalization computes

$$\mu^l = \frac{1}{H}\sum_{i=1}^{H} a_i^l \quad \sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H}(a_i^l - \mu^l)^2} \tag{3}$$

and then normalize the inputs by

$$\hat{\mathbf{a}}^l = \frac{\mathbf{a}^l - \mu^l}{(\sigma^l)^2 + \epsilon}, \tag{4}$$

where ϵ is a small positive number to avoid dividing by 0 problem.

3.2.4 Multi-head attention

This is the key part of the transformer block. The multi-head attention mechanism first appeared in [Vaswani et al., 2017]. For each head i, we first project keys, querys and values through linear projections to d_k, d_q and d_v dimensions and then we calculate

$$\text{HEAD}_i = \text{ATTENTION}(QW_i^Q, KW_i^K, VW_i^V), \tag{5}$$

where we have

$$\text{ATTENTION}(Q, K, V) = \text{SOFTMAX}(\frac{QK^T}{\sqrt{d_k}})V. \tag{6}$$

Finally we concatenate all the heads and pass it through another linear projection:

$$\text{MULTIHEAD}(Q, K, V) = \text{CONCAT}(\text{HEAD}_1, \cdots, \text{HEAD}_h)W^O \tag{7}$$

Please refer to the original paper [Vaswani et al., 2017] for more details. The self-attention layer utilizes the multi-head attention mechanism by using the same inputs for keys, queries and values.

3.2.5 Feedforward layer

This feedforward layer consists of two linear layers with the same input and output sizes. We apply a ReLU activation and then dropout to the ouputs of the first linear layer.

3.3 Keywords-awared Embeddings

Inspried by [Dhoot and Gupta], we implemented a slightly modified version of embedding layers by incorporating keywords information. In this paper, they add the maximum element-wise cosine similarities between context and query to the original word embeddings. We adapted their idea but calculated the cosine similarities between only context words and keywords in the query. We selected the keywords by using StanfordCoreNLP dependency parser [Manning et al., 2014]. Specifically, we select those words that are characterized as 'ROOT' or 'nsubj' or 'nsubjpass' or 'ccomp' or 'xcomp' or 'nummod' or 'nmod' or 'advmod' or 'compound' or 'conj'. Suppose $[c_1, \cdots, c_{l_c}]$ represents the context paragraph and $[q_1, \cdots, q_{l_k}]$ represents the keyword. For each c_i in the context paragraph and q_j in the keyword set, we calculate

$$\frac{c_i^T q_j}{\|c_i\|\|q_j\|} \tag{8}$$

and append the biggest three cosine similarities to the word embeddings for context words and then resize to the original hidden size. In our experiments, we found that although in [Dhoot and Gupta], the authors claimed relative advantage of this kind of embedding, it hurts the performance of our model on SQuAD 2.0 dataset. Hence, we remove the discussion of this particular variant of our model in the later discussion.

3.4 Data Augmentation

Considering the importance of data in training a good model, we implemented data augmentation in two steps to create a more various dataset on the basis of the original provided training data-set. Notice that we only modified the training dataset, without do any modifications on the development dataset to ensure a fair evaluation on our model's performance. We introduce some notations here to simplify our explanation: Assume for each query, we have a tuple (c,q,a), where c represents the context, q represents the query and a represents the answer.

3.4.1 Backtranslation

In this step, we only modified q and keep c & a unchanged. By observing the SQuAD2.0 dataset, we found that the meaning of question is very important in determining the answers. So we decided to use the backtranslation strategy to get a different expression for each question in the dataset without changing its meaning. We utilized the Cloud Translation API provided by Google to finish the translation part of work. We first translated from English to Spanish and then translated from Spanish to English to obtain paraphrases of questions. The reason why we chose Spanish as the intermediate language is Spanish has a similar language structure as English to try avoiding the errors caused by different word orders. After finishing this step, the augmented dataset was twice the size of the original dataset.

3.4.2 Adversarial Questions

Regarding the next step, we generated more unanswerable questions to deal with the new feature of SQuAD2.0 dataset that roughly half of the questions are unanswered. For every originally answerable

queries, we modified them and turned them into unanswerable queries, so a new tuple (c, q', a') was generated. First, to ensure the correctness of the resulting adversarial sentence, we applied semantics-altering perturbations to the question([Jia and Liang, 2017]. Next, we used antonyms from WordNet [Soergel, 1998] to replace the nouns and adjectives. Finally we replaced named entities and numbers with the nearest word in GloVe word-vector space[Pennington et al., 2014]. The augmentation process is illustrated in Figure 3.

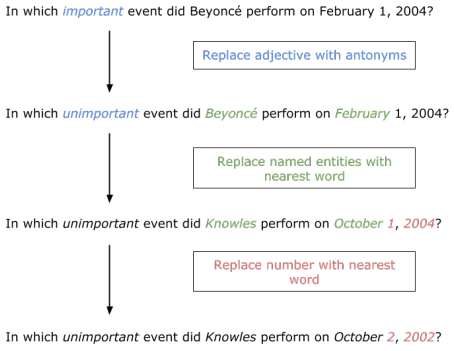

Figure 3: Illustration of Adversarial Questions

After finishing this step, the augmented dataset was twice the size of the original dataset. the final augmented dataset we got was 200% larger than the original training dataset.

4 Experiments

4.1 Data

We use SQuAD2.0, a question answering data set proposed in Rajpurkar et al. [2018]. The data set consists of context, question and answer pairs. The answer is either N/A which means the question is not answerable according to the context or an excerpt from the passage.

4.2 Evaluation method

The training loss used was the sum of negative log probabilities of the true start and end word indices given by the predicted distributions. We evaluate our model performance by two measures, EM and F1. EM stands for exact match, which gives us a 0 or 1 depending on whether or not our predicted answer matches one of the correct answers given by human. F1 is the harmonic mean of precision and recall when we answers as bag of words. This is considered to be more reliable and taken to be primary.

4.3 Experimental details

For the baseline model, we configure our model as the default starter code, namely hidden size of 100 with no dropout. For the modified model, we use dropout rate 0.1 in character embedding part and all other layers except self-matching attention, which has a higher dropout rate of 0.3. We continue to use 100 as the hidden size. For the optimizer, we use Adadelta [Zeiler, 2012]with a constant 0.5 learning rate. We fix this throughout the three models to get a fair comparison. For all these three models we ran experiments on one NV6 promo GPU and they take roughly 20, 25, 30 minutes to train per epoch. We train our models for 30 epochs and checkpoint every 50k steps. During evaluation, we pick the best saved model.

As for the QANet based models, we tried several configurations. To start with, we tried the following versions of QANet: 1 head for the self-attention layer with hidden size 96, 1 head for the self-attention

layer with hidden size 128 and 2 heads for self-attention layer with hidden size 128. We followed exactly the same set-up as in the paper for the model architecture. For all these three versions, we trained the model using Adadelta optimizer with constant learning rate 0.5 and dropout rate 0.1 for all layer dropouts. The character dropout rate is 0.05 as described in the QANet paper [Yu et al., 2018]. The other hyperparameters are exactly the same as in the starter code. The average training times for 1 epoch differ because of the hidden size. For the first model, it took us about 50 minutes to finish 1 epoch. However, with hidden size increasing to 128, the average time increased to around 1 hour. When we implemented the above three versions of QANet models, we did not adopt the stochastic depth method [Huang et al., 2016] mentioned in the paper, rather, we applied dropout layer with probabilities same as the survival probabilities in the paper. After we are done with the above three models, we found that the F1 scores and EM scores are not high enough, we thus implemented the stochastic depth method. Also, we changed the optimizer to Adam optimizer with $\beta_1 = 0.8$, $\beta_2 = 0.999$, $\epsilon = 10^{-7}$. Following the details in the QANet paper, we also implemented a warm-up scheme with an inverse exponential increase from 0.0 to 0.001 in the first 1000 steps and then keep the learning rate constant. Finally, we trained our first model (1 attention head with hidden size 96) on our augmented dataset. Since our training set is now three times larger than the original one, our average training time for 1 epoch is now 2.5 hours.

5 Results

Table 1: BiDAF-Based Model Results on DEV Dataset

Case	Char-Embedding	Self-Attention	Dev-F1	Dev-EM
1	×	×	61.62	58.11
2	✓	×	62.79	59.25
3	✓	✓	63.75	60.63

Table 2: QANet-Based Model Results on DEV Dataset

Case	Head #	Hidden Size	Augmentation	Adam+Stochastic Depth	Dev-F1	Dev-EM
4	1	96	×	×	64.37	60.91
5	1	128	×	×	63.43	59.45
6	2	128	×	×	63.54	59.96
7	1	96	✓	×	65.21	61.80
8	1	96	×	✓	64.98	61.44

Table 3: Model Results on TEST Dataset

Case	Model Description	Test-F1	Test-EM
7	QANet w/ Data Augmentation	61.83	57.90
8	QANet w/ Adam + Stochastic Depth Method	62.08	58.58
4	QANet	60.04	56.28

Table 1 shows our results on dev set for BiDAF-based models. We see that char embedding and self-attention do help improve the model performance. However, the improvements are less significant than what we expect. One possible explanation is that by using more complex models, hyperparameter tuning and optimizer choice becomes more important. Table 2 shows our results on dev set for different variants of QANet models. We see that increasing the number of attention heads and increasing the hidden size do not help. However, adding the stochastic depth method and using Adam optimizer together gives us better performance in terms of both F1 and EM. Moreover, training our model on augmented dataset gives us an increase in both F1 and EM. The results fit our intuition. By using a very large model like QANet, careful training and more data do help us improve the performance of our model. Table 3 shows our three best submissions on the test leaderboard. We see that although the model trained with data augmentation shows the best performance on the dev set, it is not the case on test set. One explanation is that our augmented data may not be good since we use

backtranslation and some simple word substitutions. Hence, our model may not generalize well on the real test set.

6 Analysis

6.1 QANet with Data Augmentation

Question: Economy, Energy and Tourism is one of the what?

Context: Subject Committees are established at the beginning of each parliamentary session, and again the members on each committee reflect the balance of parties across Parliament. Typically each committee corresponds with one (or more) of the departments (or ministries) of the Scottish Government. The current Subject Committees in the fourth Session are: Economy, Energy and Tourism; Education and Culture; Health and Sport; Justice; Local Government and Regeneration; Rural Affairs, Climate Change and Environment; Welfare Reform; and Infrastructure and Capital Investment.

Answer: current Subject Committees

QANet w/ Data Augmentation Prediction: N/A

Analysis: We found that our QANet model with data augmentation predicted the answer to be N/A for an anwerable questions. The possible reason might be we augmented the unanswerable questions. Originally, there were roughly half of the questions to be unanswerable, but after data augmentation, two of the thirds of the questions were unanswerable, which might made the model have more bias to those unanswerable ones and thus have more possibility to generate N/A answers. A possible solution is re-balance the spread of unanswerable questions in the dataset.

6.2 QANet

Question: How many miles is Montpellier from Paris?

Context: Montpellier was among the most important of the 66 "villes de sûreté" that the Edict of 1598 granted to the Huguenots. The city's political institutions and the university were all handed over to the Huguenots. Tension with Paris led to a siege by the royal army in 1622. Peace terms called for the dismantling of the city's fortifications. A royal citadel was built and the university and consulate were taken over by the Catholic party. Even before the Edict of Alès (1629), Protestant rule was dead and the ville de sûreté was no more.[citation needed]

Answer: N/A

Prediction: 66

Analysis: We see that QANet sometimes also gives answer for unanswerable questions. In this case, 66 is not the distance from Montpellier to Paris but the model gives this answer. One possible explanation is that the model cannot understand the query quite well. Although there is no answer to this query based on the context, the model is still trying to generate some answer. If we think about the problem in this way, one possible direction is to better model the semantic relation between the query and the context, i.e. we should let the model learn if the query and the context are related.

7 Conclusion

In this project, we worked on SQuAD challenge and tried to improve model performance on the SQuAD2.0 dataset. We studied and implemented both BiDAF and QANet based models, and did various experiments on our models, such as adapting different data augmentation strategies. We are able to achieve better results than the baseline models on the dataset.

We fully admit that there are still much space for improvement on our model, and the future work can be the fine-tuning of the hyperparameters, or more trials on models with different data augmentation strategies. We are also interested to see why our QANet model did not achieve the high F1 and EM scores claimed by other teams. We learned a lot on NLP during the whole semester and would like to express our sincere gratitude to all teaching staffs.

8 References

References

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

Anand Dhoot and Anchit Gupta. Iterative reasoning with bi-directional attention flow for machine comprehensionn. https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1184/reports/6908250.pdf.

Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep networks with stochastic depth. *CoRR*, abs/1603.09382, 2016. URL http://arxiv.org/abs/1603.09382.

Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems, 2017.

Lukasz Kaiser, Aidan N. Gomez, and Francois Chollet. Depthwise separable convolutions for neural machine translation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=S1jBcueAb.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014. URL http://www.aclweb.org/anthology/P/P14/P14-5010.

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. doi: 10.3115/v1/D14-1162. URL https://www.aclweb.org/anthology/D14-1162.

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

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2016. URL http://arxiv.org/abs/1611.01603.

Dagobert Soergel. Wordnet. an electronic lexical database. 10 1998.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *CoRR*, abs/1505.00387, 2015. URL http://arxiv.org/abs/1505.00387.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 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)*, pages 189–198, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1018. URL https://www.aclweb.org/anthology/P17-1018.

Wikipedia. Question answering — Wikipedia, the free encyclopedia, 2004. URL https://en.wikipedia.org/wiki/Question_answering. [Online; accessed 22-July-2004].

Adams Wei Yu, David Dohan, Quoc Le, Thang Luong, Rui Zhao, and Kai Chen. Fast and accurate reading comprehension by combining self-attention and convolution. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B14TlG-RW.

Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012. URL http://arxiv.org/abs/1212.5701.

A Appendix

We attached our results pictures from tensorboard on the dev dataset for the best three models.

A.1 Model Case #: 7

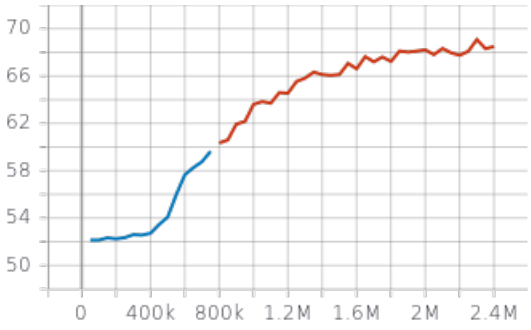

Figure 4: AvNA for Model 7

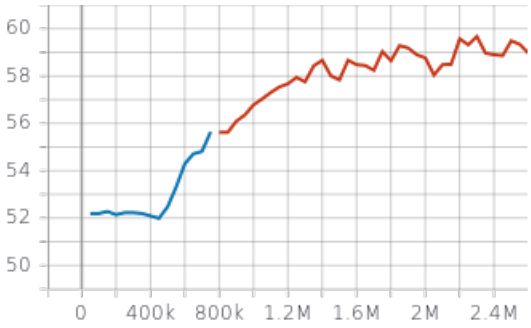

Figure 5: EM for Model 7

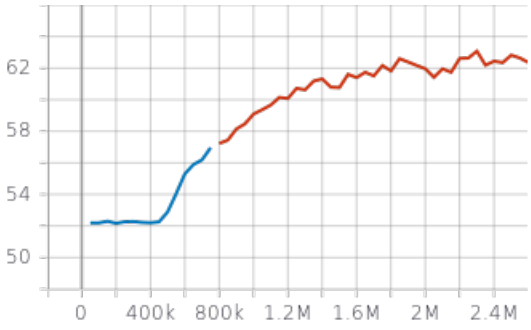

Figure 6: F1 for Model 7

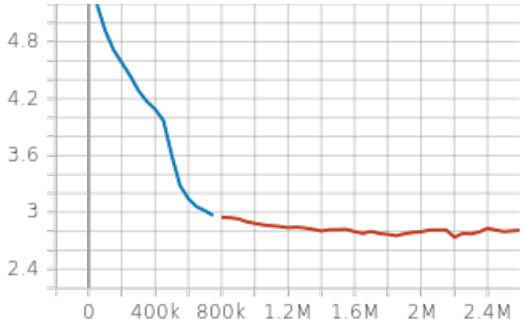

Figure 7: NLL for Model 7

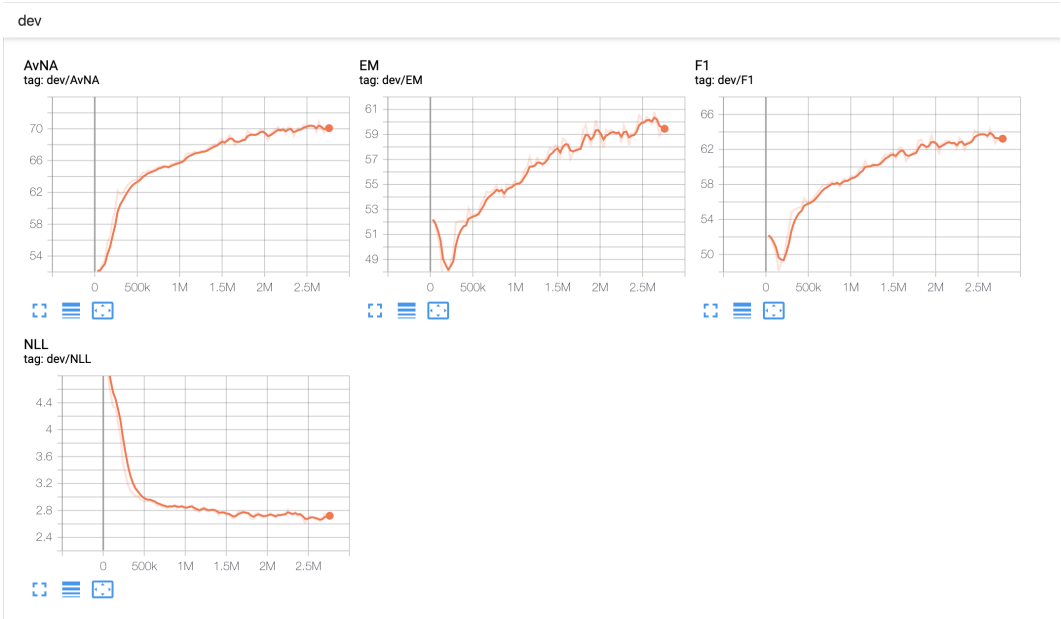

Figure 8: Dev Statistics for Model 8