Chained Self-Attention

Jai Gupta
jaigupta@stanford.edu

Abstract

Self-attention based networks (particularly Transformers introduced by Ashish Vaswani [2017]) have been used in a wide variety of tasks for finding context aware representation of input sequences. They have particularly been very successful in language modeling where the context sensitive representations have been proved to be very successful agnostic of the task they are evaluated for [Jacob Devlin, 2018].

In the self-attention layers of a Transformer block, a probability distribution over the complete sequence is generated for every term in the sequence termed as attention distribution of that term. Each of these attention probability matrices are expected to represent some relationship between the sequence terms. Hence, the probability distribution can be visualized as the probability that a direct relationship exists between two terms in the sequence.

With Chained Self-Attention, we are trying to ease the ability of the network to find indirect dependencies between the input sequence terms. Deep multi-layer networks are able to find such dependencies after few layers. Chained Self-Attention aims to provide signals about chains of such relationships at a much earlier stage. A relationship chain exploits the existence of transitivity in some of the relationships to create new attention representations that can take indirect relationships into consideration. Essentially for a given relationship, if $w1$ is related to $w2$, and $w2$ is related to $w3$, then $w1$ may have the same relationship with $w3$ and we provide this explicitly as a signal to the self attention layer. To the best of our knowledge, such a network hasn't been tried before.

We also show comparative results by utilizing chained self-attention in BiDAF and QANet models and show qualitative improvements on SQuAD 2.0 dataset [Pranav Rajpurkar and Liang, 2018] of 0.60 and 1.02 points respectively for F1 metric.

1 Introduction

Interest in various natural language processing and understanding tasks have spiked over the recent years within the research community. Machine reading comprehension and question answering is one such task that has piqued the interest of researchers.

Comprehension and Question answering is particularly interesting since it can bridge the gap between how humans interact with each other and how they interact with artificial systems. In order to build an intelligent system that can pass the Turing test, question answering is the key challenge to solve. Further, DecaNLP [Bryan McCann] has made the community realize the fact that all tasks related to natural language processing can be modeled as a question answering task.

While majority of the systems used to have a complex architectures with many sub components in the past, recent neural net based models with end-to-end training have outperformed them by huge margins. Most of these neural network models consist of RNNs, CNNs, and most recently Transformers networks as the basic building block for processing sequential inputs. Transformer network are the latest and are ubiquitous in language modeling network architectures. They have proved to be very useful in various kinds of NLP tasks. BERT [Jacob Devlin, 2018], a contextual word embedding, that builds on top of transformer networks has proved to be very useful regardless of the task it is being used for. Since these networks are pretty new, there are huge scopes for improvement and there have been many proposed improvements of the BERT model and Transformer networks already, but self-attention networks still exist as a key component. Still being able to improve the representative power of these models is challenging and active area of research since little is known about how the model is internally able to learn these representations.

In this paper we propose a way to improve the efficiency of the Transformer blocks to ease their ability to learn. This is done by providing additional interaction signals between the sequence terms than what is produced by a regular self-attention layer. As mentioned before, it is hard to comprehend how existing models internally represent the provided sequences. In case of multi-head attention models (part of Transformer networks), it was found that each head represent different kinds of relationships between the sequence terms by analyzing the attention distributions across different heads. We hypothesize that some of the relationships are transitive in nature and exploit this property to provide the network more information about indirect relationships between the sequence terms.

The key idea behind this modification is that a self-attention network needs multiple depths to figure out indirect relationships. Sometimes, it might completely fail to figure them. Providing these relationships as explicit signals ensures that they are taken into consideration early on thereby improving the representative ability of the model while barely increasing its complexity.

In order to validate that these modifications are effective, we perform experiments against two baseline models. The first one is a modified version of the BiDAF network [Minjoon Seo, 2017] to include self-attention layers, while the second one is the QANet model [Adams Wei Yu, 2018]. We show a gain of 0.60 and 1.02 points in F1 metrics and 0.60 and 1.27 points in EM metrics for the two models respectively on the SQuAD 2.0 dataset.

2 Related Work

A significant amount of development in the use of end to end neural network models should be accredited to the availability of large amount of dataset to train on. Datasets such as MCTest [Matthew Richardson and Mctest, 2013] were small, and hence needed lot of manual human intervention to train small components that specialized in different language constructs which added up to create the final larger system. But with the availability of the Cloze Test datasets [Karl Moritz Hermann and Blunsom, 2015], Childrens Book Test [Felix Hill and Weston, 2016], and the SQuAD dataset [Pranav Rajpurkar and Liang, 2018], it has become possible to train end to end neural network models without any significant human intervention.

Dzmitry Bahdanau and Bengio [2015] and Minh-Thang Luong and Manning [2015] laid the foundation for the use of attention in Machine Comprehension tasks. Further improvements by using dynamic attention models [Karl Moritz Hermann and Blunsom, 2015], using bi-linear term for computing attention weights [Danqi Chen and Bordes, 2017] and attending query terms from context [Wang and Jiang, 2016] have been proposed. Other modifications include variations in how the attention weights are passed through out the network(Rudolf Kadlec and Kleindienst [2016] , Yiming Cui and Hu [2016]), and the use of multi-hop (Alessandro Sordoni and Bengio [2016], Bhuwan Dhingra and Salakhutdinov [2016]). Significant improvements to the state of art was achieved by BiDAF by using memory-less attention mechanism [Minjoon Seo, 2017] which we use as one of our baselines.

Ashish Vaswani [2017] proposed the use of attention layers to replace all recurrent units which significantly sped up the model's training and inference time. Further improvements to it were proposed in David R. So [2019] where neural search helped find a better stacking of different sub components but the self-attention layer (which we propose to improve in this paper) is left untouched. QANet [Adams Wei Yu, 2018] proposes a model using self-attention for question answering task. We will be using QANet as another baseline in order to improve its self-attention layer.

3 Approach

Transformer networks are ubiquitous and have proved to be very useful in various kinds of NLP tasks. In this paper we provide a way to improve the self attention layer of the Transformer blocks. We first present a description of the self-attention layer which is the key component of Transformer blocks in section 3.1. Further, we provide details about the proposed modification to the self-attention layer

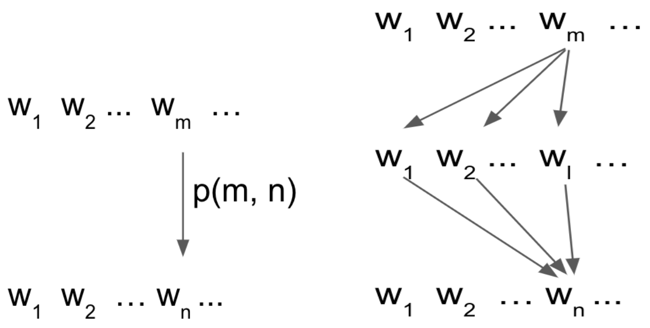


Figure 1: Direct and indirect relationship between terms. On the left is a depiction of direct relationship, while on the right is indirect relationship whose probability for a path is the product of all segments on that path which is summed over all possible paths to find the final probability.

called chained self-attention in section 3.2 with rationale behind why such a modification makes sense. All models mentioned here were coded from scratch taking hints from the papers. We took hints for creation of positional encoding from attention-is-all-you-need github repo [Huang, 2018].

3.1 Self-Attention Layer

A self attention layer tries to represent the probability of a relationship/attention between the terms of a sequence, and then use them to find a new representation for each of the terms in the sequence. A multihead self-attention layer can be considered as evaluating multiple self-attention networks in parallel. The input to the layer is a sequence of terms of embedding dimension size att_dim. Then the following list of transformations are performed in sequence:

- Each of the input terms is passed though a linear transformation to create three outputs K (key), Q (query) and V (value) each of size att_dim.
- If the number of heads is n_heads, each of K, Q and V are split into n_heads smaller representations each of size $\frac{att\_dim}{n\_heads}$. For this reason, att_dim is chosen to be a multiple of n_heads.
- Let K_i, Q_i and V_i represent the key, query and value of the i_{th} head. Let $k (= \frac{att\_dim}{n\_heads})$ represent the dimension size of each head.
- We start with finding scaled dot product between K_i and Q_i for each head as:

  $$P_i = softmax(\frac{QK^T}{\sqrt{k}})$$

  Each head in a multihead attention network is assumed to represent some relationship between the terms in the sequence. Thereby each entry in the P_i matrix represents the probability of the relationship between the terms in the input sequence.
- This probability matrix can also be thought of as the attention each term pays to another term in the sequence for a given relationship. The new representation taking these attentions into consideration is:

  $$Att_i = P_i V_i$$
- The final output of the network is the concatenation of the attention representation base output Att_i from each of the heads to create the output representation V_{att}

3.2 Chained Self-Attention

Chained self-attention network performs a small modification to the self attention network to provide a pretty useful signal to the network. The network utilizes the effect of chaining the attention probability matrix P_i above. As discussed above, $P_i(m, n)$ represents the probability of a relationship between

the m_{th} and n_{th} terms in the input sequence. Lets call this matrix the direct relationship probability matrix.
Now consider the matrix P_i^2 created by multiplying P_i with itself. The entries in this matrix represent the probability of an indirect relationship between any given two terms in the input sequence. $P_i^2(m,n)$ now represents:

$$P_i^2(m,n) = \sum_l P_i(m,l) * P_i(l,n)$$

This can be thought of as the probability of a relationship between m and n through all other terms l in sequence as depicted in Figure 1. Note that we do not specifically remove m and n from the possible values of l to ease out calculations.

Similarly we can evaluate higher order relationship matrices as:

$$P_i^3 = P_i * P_i^2$$
$$P_i^4 = P_i * P_i^3$$

After this, Att_i corresponding to each of these distributions is evaluated as:

$$Att_i^1 = P_i V_i$$
$$Att_i^2 = P_i^2 V_i$$
$$Att_i^3 = P_i^3 V_i$$
$$Att_i^4 = P_i^4 V_i$$

The input to the multihead self-attention layer had n_heads representations. At this point, we have created 4 representations per head thereby creating a total of $4 * n\_heads$ representations. Hence, we add a fully connected layer on top of these representations to reduce to the same number of dimensions as the input.

$$\overline{Att_i} = [Att_i^1 Att_i^2 Att_i^3 Att_i^4]$$
$$Att_i = \overline{Att_i} W_{attn}$$

Self-attention networks are generally stacked with same dimension size. Hence, it was necessary for the output to have the same dimension size as the input.

The max chain length hyper-parameter was chosen as 4 empirically as negligible gains were observed on providing signals for indirect relationships of length greater than 4. This experiment were performed for smaller variations of the qanet model in section 4.5 with just 2 encoder blocks for 5 epochs.

4 Experiments

Our main motive here is to quantify the improvements that chained self-attention networks brings to models over regular self-attention models. In order to do that, we train two baseline models that contain self-attention layers, and then we show the improvement we get by replacing the self-attention layers with chained self-attention. Additionally, in the later sections, we also show how we built a model with further improved performance by borrowing ideas from David R. So [2019] and creating a simple ensemble model. In the following sections, we present the list of models that were trained as part of these experiments.

4.1 BiDAF

BiDAF [Minjoon Seo, 2017] has been a very successful model for SQuAD. With the bidirectional attention flow mechanism, they were able to achieve state of the art performance. This is the model that has been provided as the baseline in the default project handout [Chute, 2019] but has additional modification to add character-level embeddings as input to the embedding layer. In the BiDAF model, the embedding layer consists of embedding lookup followed by 1) a dropout layer, 2) a linear transformation layer, and 3) two layers of the highway network [Rupesh Kumar Srivastava, 2015]. The embedding lookup layer is frozen. The model was trained with embedding size of 128.

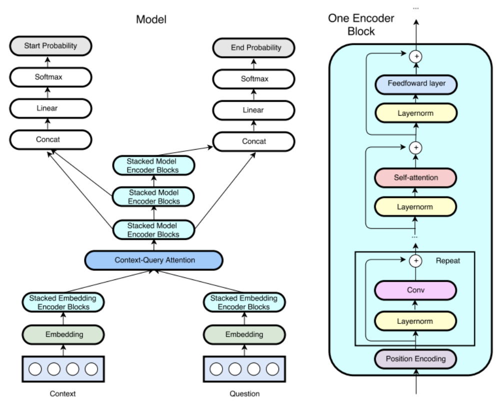


Figure 2: The QANet architecture. See Adams Wei Yu [2018] for detailed architecture description.

The char embedding support was added using a CNN that scans over the character embeddings using window size of 5 and outputs embedding of size 200 for each character. A max over each dimension of the output embeddings is taken over all characters to get the final word-level representation which is a fixed size embedding of size 200. This is the same approach as mentioned in ass5. This word-level representation is concatenated with the word-level embedding from GloVe [Jeffrey Pennington, 2014] before the linear layer reduces the dimension size to the internal embedding size of 128 after which rest of the model is same as the original BiDAF model mentioned in Minjoon Seo [2017].

4.2 BiDAF + Self-attention

The main motive of these experiments is to quantify the gain on replacing self-attention layer with chained self-attention. But the BiDAF model presened in the last section does not have a self-attention component to it. Hence, we create another model that adds self-attention to the BiDAF network from section 4.1. The embedding layer of the model was modified to use three layers of transformer blocks replacing the two layers of highway network. Similar to the baseline model, the embeddings are loaded from the pre-trained GloVe embedding provided with the handout and are frozen during the experiment. This isn't particularly a great model, but creates an artificial baseline for fair evaluation of the impact of chained self-attention layer.

4.3 BiDAF + Chained self-attention

In this model, the self-attention segments of the model above are modified to include signals from chaining of self-attention probabilities. As discussed before, they are intended to provide additional attention probabilities for relationship chains of length 2, 3 and 4. This is simply done by replacing the self-attention layer with chained self-attention layer in the transformer blocks as mentioned in the section 3.2. No other modification was done to the model.

4.4 QANet + Self-attention

QANet [Adams Wei Yu, 2018], as seen from the results mentioned in the paper, outperforms BiDAF by large margins. Further, it inherently makes use of the self-attention, so we can easily quantify improvements to it on using chained self-attention. A flaw with the self-attention integration with the BiDAF model in section 4.3 is that we did not provide positional encoding as input to the layers which is a pretty important signal for self-attention based network. For this QANet model, and the one in section 4.5, we provide positional encoding as an input to the network as described in Ashish Vaswani [2017].

The basic architecture of the QANet model is present in figure 2. Additionally, though it mentions the used of single feed forward network after the self attention layer, we found it to work better with two feed-forward layer with a ReLU unit in between and use that instead in all our implementations. This idea was used borrowed from Transformer blocks in Ashish Vaswani [2017].

4.5 QANet + Chained self-attention

Similar to the experiment in section 4.3 with BiDAF model, the self-attention network in the QANet model in section 4.4 was replaced with the chained self-attention network mentioned above in section 3.2. There were no other changes to the network.

4.6 Data

All experiments use the training data provided by SQuAD 2.0 dataset (Pranav Rajpurkar [2016] and [Pranav Rajpurkar and Liang, 2018]) whose dev set was split as mentioned in the default project handout [Staff, 2019]. Pre-trained word embeddings and character level embeddings were used as mentioned in the handout 2019. No additional data is used in training of any of the models.

4.7 Evaluation method

The experiment results are evaluated against the following standard set of metrics provided in the default project handout: AvNA, EM and F1 scores. Each of these metrics are explained in the handout (2019), so we are skipping any explanation here. Additionally, we provide manual analysis of the obtained results in section 5.

4.8 Experiment details

Each model being evaluated has already been described above. We perform evaluations in two groups. The first group does a comparative study of the three BiDAF models, while the second group does the same for the two QANet models.

4.8.1 BiDAF

We did not perform any hyperparameter tuning. For the BiDAF models, all dropout layers have used a dropout probability of 0.2. Adadelta optimizer is used with a fixed learning rate of 0.5. Batch size of 32 was used. The three versions of the BiDAF model mentioned above were run for approximately 17 epochs (2.5 million iterations), after which the results didn't seem to vary much for either of the models and were sufficient to make conclusive decision. Each epoch of the BiDAF models took approximately 15 minutes.

These models were trained on a standard nv6 Azure machines which is powered by a Nvidia Tesla M60 GPU with a GPU memory of 8GB.

4.8.2 QANet

For the two QANet models, we use the same hyperparameter setting as mentioned in the QANet paper [Adams Wei Yu, 2018]. All dropout layers have a dropout probability of 0.1. The dropout layer on top of character embeddings has a dropout probability of 0.05. We used Adam optimizer with β_1 set to 0.8, β_2 set to 0.999 and ϵ set to 10^{-7}. Learning rate is warmed up from 0 to 0.001 in the first 1000 steps and then kept constant. We did observe that using this warm-up led to better initial learning curve compared to a constant learning rate of 0.001, but the difference seemed to diminish with time. We use l2 weight decay with $\lambda = 3X10^{-7}$ and exponential moving average with a decay rate of 0.9999 for all trainable parameters. Layer normalization and dropout is used between all layers. Stochastic layer dropout was used for encoder block layers with dropout probabiliy increasing linearly from 0 to 0.1 [Gao Huang and Weinberger]. Hyperparameter tuning on these values did not give any significant gain. The two versions of the QANet models were trained for approximately 23 epochs (3 million iterations) with a batch size of 16.

All experiment models were trained on a standard nv12 Azure machines which is powered by two Nvidia Tesla M60 GPUs with a GPU memory of 16GB. Hyperparameter tuning experiments were performed on a machine with Nvidia Quadro P5000 GPU with a memory of 16GB.

4.9 Results

4.9.1 BiDAF model results

The three models evaluated here are (1) BiDAF, (2) BiDAF + self-attention, and (3) BiDAF + chained self-attention. The metrics we found at the end of approx 17 epochs (2.5 million iterations) are presented below:

Metric	BiDAF	BiDAF + self-attention	BiDAF + chained self-attention
AvNA	66.84	67.27	**67.73**
EM	57.06	56.85	**57.45**
F1	60.33	60.37	**60.97**

Plots of the metrics during training, present in figure 3, show that using chained attention layers has a clear advantage for all the metrics compared to the regular self attention layers. Compared to the regular self-attention network, our modification gets a gain of 0.46, 0.60 and 0.60 points for the AvNA, EM and F1 metrics respectively. This experiment performed an initial validation that chained self-attention is indeed useful and motivated us to try the same on QANet.

It can also be observed that BiDAF with self-attention did not perform particularly better than regular BiDAF. The reason we think self-attention could not perform as well is due to the missing positional encoding features. In experiments on QANet, we do provide the positional encodings as input to ensure that self-attention layers don't lack this much required feature.

4.9.2 QANet model results

The two models evaluated here are (1) QANet with self-attention, and (2) QANet with chained self-attention. The metrics we found at the end of 23 epochs are presented below. We stopped at 23 iterations as the models started showing signs of over-fitting after this point:

Metric	QANet + self-attention	QANet + chained self-attention
AvNA	74.72	**75.54**
EM	65.37	**66.64**
F1	68.89	**69.91**

We observe a gain of 0.82, 1.27 and 1.02 points in the AvNA, EM and F1 metrics respectively on replacing the self-attention layers with chained self-attention layer. Clearly chained version of self-attention is able to beat the baseline by a huge margin for each of the metrics. The plot for these metrics over the dev dataset during training is present in figure 4. Due to increase in the number of computations, a increase of approximately 2.3% was observed on the average training and inference time for the model. The number of trainable parameters in the model increased by a mere 0.11% (1.755 million vs 1.753 million).

4.9.3 Further improvements

Most of the additional improvements were borrowed from "The Evolved Transformer" [David R. So, 2019] and the first change was to add a gated linear unit prior to applying the self attention layer. Additionally, the QANet model consisted of two feed forward units (with a ReLU activation in between) where output and input of each layer was same as the hidden embedding dimension used throughout the network. Following the idea from David R. So [2019] the first feedforward unit was modified to double the number of dimension from 128 to 256. Once this was passed through the ReLU activation, the second feedfoward unit brought down the dimension size back to 128. Additionally, Leaky ReLU were used to replace all ReLU units throughout the model and a slight improvement in the model learning curve was observed. On training the improved model for approximately 30 epoch. We got EM of 67.552 and F1 score of 70.676 (dashboard id: "Chained Self-Attention") which topped the dev dashboard.

Analysis presented below in section (5) showed that the model would benefit from using ensemble. A regular QANet model uses pointer network to find a probability distribution for the start and end indexes. At inference time, the start and end positions are chosen such that $p_{start\_pos} * p_{end\_pos}$ is maximized.

Due to resource constraints and observations in the analysis section 5 below, no training was performed. Two models were loaded and while calculating the value of $start\_pos$ and end_pos such that $p_{start\_pos} * p_{end\_pos}$ is maximized, we look for the max in the output distribution over both the models. This lead to a F1 score of 71.04, much higher than both the individual models. Ensemble over four such models lead to a F1 score of 72.06.

Note that all metrics presented here are on the dev dataset since analysis on the test dataset is pretty difficult due to limited number of allowed submissions.

5 Analysis

An analysis of the output of the two models showed that the two model performed good for a different set of questions. It was clear that the two were converging at a pretty different minimas. Retraining the models also showed similar convergence pattern. On careful analysis of examples where only one of them was correct, there were a good fraction of questions where the correct answer had very high confidence from one model while the other model had a low confidence on all positions. Hence, it was easy to observe that the results might benefit significantly by taking a max over the two outputs. As mentioned in the section 4.9.3, this lead to huge gains with F1 of 72.06. Perhaps using a ensemble over more models can further improve gain due to coverage of more number of such minimas.

On analyzing the erroneous outputs of our models, two important class of errors that we found were:

- The model is biased to prefer span of text from context as output that are closer to terms similar to terms from the question. While this works in most of the cases pretty well, we could see examples where we believe this bias caused the model to output erroneous values. For example, the dev dataset has an example where context had the sentence "The rotational inertia of planet Earth is what fixes the constancy of the length of a day and the length of a year.", and the question was "The rotational inertia of planet Mars is what fixes the what?". Due to multiple terms from the question matching this sentence, the model outputs "constancy of the length of a day and the length of a year" as answer, while the answer is not present in the context.
- Since most of the answers are noun phrases, and noun phrases can be a composition of other noun phrases, the boundary of what should be the right phrase to output may differ in opinion even between humans. For example, given the context, "I am the Queen of the Dothrakis, Daenerys", and the question, "Who am I?", both the noun phrases "Queen of the Dothrakis, Daenerys" and "Daenerys" are good answers. The choices differ across examples which may confuse the model. Additionally, there are also cases when the model has trouble deciding the boundaries where human may have close to unanimous opinion. Taking an example from the dev dataset, the context had the sentence, "The experimental measurements made by Watt on a model steam engine led to the development of the separate condenser.", and the question was, "What was developed from Watt's measurements on a model steam engine?". The correct answer here is "condenser", but the model's output was "separate condenser". Similarly there are many other cases where the models response is either a noun phrase which contains the correct noun phrase as a child, or vice versa. Essentially the model is able to identify the where the answer lies, but finding the correct noun phrase boundary is still difficult.

From the above two observations, it looks like the model is performing significant amount of pattern matching while answering these questions. It is possible that a majority of the questions are a result of performing pattern matching of the context with the query to find the a distribution of matches followed by extraction of nearby noun phrase. Hence, it is still questionable whether the model has a good understanding of language's construct and more evaluation is needed to ensure that the internal computations of such models align with our expectations about how the model should answer these questions.

6 Conclusion

We have conclusively validated the effectiveness of chained self-attention layers over regular self-attention layers. Self-attention layers are used today across a wide variety of tasks, and this improvement should work across all such models that make use of self-attention layers.

We also explored few other ideas about how to further improve the model using ideas from David R. So [2019], using Leaky ReLU and creating a simple ensemble model.

We also presented an analysis of the model outputs and it raises serious concerns whether such models are really able to understand the construct of the language like we intend them to do.

The chaining of attention presented in this paper is for a single attention head. If P_1 is the attention distribution for one head and P_2 is the distribution for another head, then P_1P_2 will denote a cross head relationship chain. Finding an efficient way to chain relationships across different attention heads is a future work that we would like to explore.

References

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

Phillip Bachman Alessandro Sordoni and Yoshua Bengio. Iterative alternating neural attention for machine reading. 2016.

Niki Parmar-Jakob Uszkoreit Llion Jones Aidan N. Gomez Lukasz Kaiser Illia Polosukhin Ashish Vaswani, Noam Shazeer. Attention is all you need. *Neural Information Processing Systems (NIPS)*, 2017.

William W Cohen Bhuwan Dhingra, Hanxiao Liu and Ruslan Salakhutdinov. Gated-attention readers for text comprehension. *arXiv preprint arXiv:1606.01549*, 2016.

Caiming Xiong Richard Socher Bryan McCann, Nitish Shirish Keskar. The natural language decathlon: Multitask learning as question answering.

Chris Chute. Squad Github Repo, 2019.

Jason Weston Danqi Chen, Adam Fisch and Antoine Bordes. Reading wikipedia to answer opendomain questions. *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers, pp. 1870–1879*, 2017.

Quoc V. Le David R. So, Chen Liang. The evolved transformer. *arXiv:1901.11117*, 2019.

Kyunghyun Cho Dzmitry Bahdanau and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *ICLR*, 2015.

Sumit Chopra Felix Hill, Antoine Bordes and Jason Weston. The goldilocks principle: Reading children's books with explicit memory representations. *ICLR*, 2016.

Zhuang Liu Daniel Sedra Gao Huang, Yu Sun and Kilian Q. Weinberger. Deep networks with stochastic depth.

Yu-Hsiang Huang. https://github.com/jadore801120/attention-is-all-you-need-pytorch, 2018.

Kenton Lee Kristina Toutanova Jacob Devlin, Ming-Wei Chang. Bert: Pre-training of deep bidirectional transformers for language understanding. *CoRR, abs/1810.04805*, 2018.

Christopher D. Manning Jeffrey Pennington, Richard Socher. Glove: Global vectors for word representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543*, 2014.

Edward Grefenstette Lasse Espeholt Will Kay Mustafa Suleyman Karl Moritz Hermann, Tomas Kocisky and Phil Blunsom. Teaching machines to read and comprehend. *NIPS*, 2015.

Christopher JC Burges Matthew Richardson and Erin Renshaw. Mctest. Mctest: A challenge dataset for the open-domain machine comprehension of text. *EMNLP*, 2013.

Hieu Pham Minh-Thang Luong and Christopher D. Manning. Effective approaches to attentionbased neural machine translation. *EMNLP*, 2015.

Ali Farhadi Hannaneh Hajishirzi Minjoon Seo, Aniruddha Kembhavi. Bidirectional attention flow for machine comprehension. *ICLR*, 2017.

Konstantin Lopyrev Percy Liang Pranav Rajpurkar, Jian Zhang. Squad: 100,000+ questions for machine comprehension of text. *CoRR, abs/1606.05250*, 2016.

Robin Jia Pranav Rajpurkar and Percy Liang. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*, 2018.

Ondrej Bajgar Rudolf Kadlec, Martin Schmid and Jan Kleindienst. Text understanding with the attention sum reader network. *ACL*, 2016.

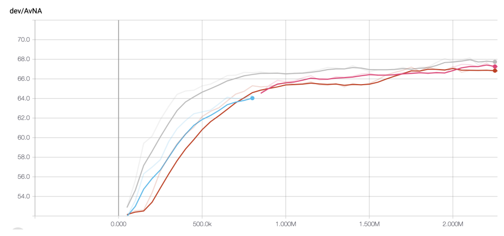
Jürgen Schmidhuber Rupesh Kumar Srivastava, Klaus Greff. Highway networks. *arXiv:1505.00387v2*, 2015.

CS224N Teaching Staff. CS224N, Default Project Handout, 2019.

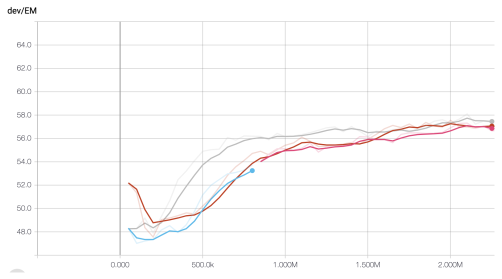
Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. *CoRR, abs/1608.07905*, 2016.

Si Wei Shijin Wang Ting Liu Yiming Cui, Zhipeng Chen and Guoping Hu. Attention-overattention neural networks for reading comprehension. *arXiv preprint arXiv:1607.04423*, 2016.

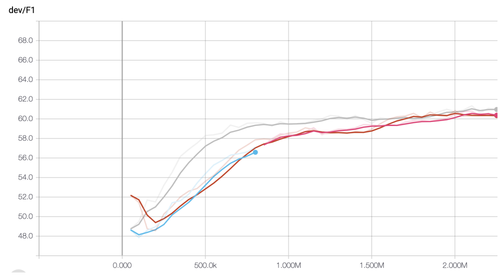
7 Appendix



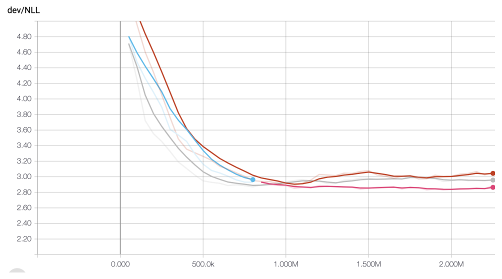
(a) AvNAA



(b) EM

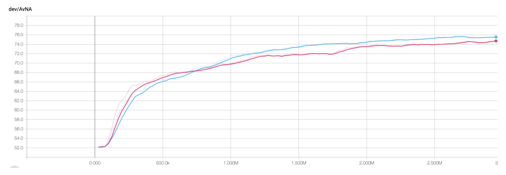


(c) F1

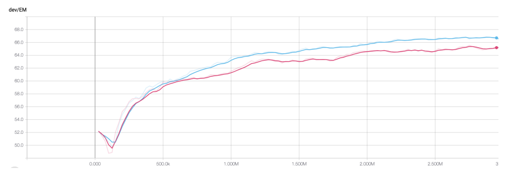


(d) NLL

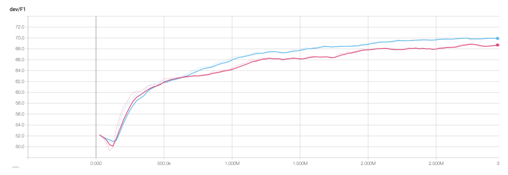
Figure 3: Comparison of metrics for the three BiDAF models. (1) The brown line represents the BiDAF model (baseline1), (2) the blue line and its continuation pink represents BiDAF model with self-attention, and (3) the grey line represents the BiDAF model with chained self-attention.
. 2 is broken into two lines because we had to stop and restart training of the model in between.



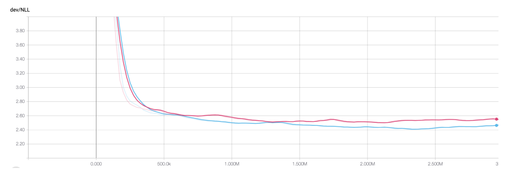
(a) AvNAA



(b) EM



(c) F1



(d) NLL

Figure 4: Comparison of metrics for the two QANet models. (1) The blue line represents the baseline QANet model (with self-attention), and (2) the orange line represents QANet model with chained self-attention.