Question Answering System on The SQuAD Dataset (Default Project)

Gevorg Khondkaryan
gevorg@stanford.edu

Abstract

The idea of the project is to have a question answering system which works on SQuAD. SQuAD is a reading comprehension dataset.The input is a paragraph, and a question about that paragraph.The goal is to answer the question correctly.This task provides measure for how well systems can understand text. The paragraphs in SQuAD are from Wikipedia. The questions and answers were crowdsourced using Amazon Mechanical Turk. It has about 150k questions and about the half of questions cannot be answered using the provided paragraph.

1 Introduction

While the amount of knowledge available as linked data grows, so does the need for providing end users with access to this knowledge. Especially question answering systems are receiving much interest, as they provide intuitive access to data via natural language and shield end users from technical aspects related to data modeling, vocabularies and query languages. Question answering is a good compromise between intuitiveness and expressivity, which has attracted the attention of researchers from different communities. Machine comprehension (MC), answering a query about a given context paragraph, requires modeling complex interactions between the context and the query. Recently, attention mechanisms have been successfully extended to MC. In this project I will concentrate on reading comprehension and question answering. For majority speakers the goal will be to answer questions correctly and majority will agree on both provided question and answers and will wont contain irrelevant information. As a data source I am using Stanford Question Answering Dataset(SQuAD)[1] version 2. This dataset contain more that 150K+ question-answer taken from Wikipedia dataset and to answer every question will be a segment of text, or span, from the context paragraph.Model shell select span or segment of the text to answer to the question. To evaluate the performance of the model F1 and Exact Match (EM) metrics are used. F1 is given by the harmonic mean of precision and recall and Exact Match measures the exact string match between the predicted answer.

2 Related Work

The leaderboard on the SQuAD website shows many deep learning models created for this dataset. And many researchers have come up with different architectures. First models for SQuAD 2.0 were evaluated by Rajpurkar et al. These are BiDAF-No-Answer(BNA) model proposed by Levy et al.(2017) [6], and two versions of the DocumentQA No-Answer (DocQA) model from Clark and Gardner (2017) [7], namely versions with and without ELMo (Peters et al., 2018) [8]. These models all learn to predict the probability that a question is unanswerable, in addition to a distribution over answer choices. From these three the best model DocQA + Elmo, achieves only 66.3 F1 on the test set. Most of the architecture solutions participated in this challenge have ELMo and BERT architectures. ELMo stands for Embedding from Language Models and BERT stands for Bidirectional Encoder Representations from Transformers. And the core idea of ELMo and BERT

is to represent a piece of text using word embeddings that depend on the context in which the word appears in the text. Currently the first place takes BERT+MMFT+ADA(ensemble) created by Microsoft researchers which has EM=85.082 and F1=87.615 scores.

3 Approach

For the project 3 approaches were taken, Bert, Bidaf, QANet.

3.1. BERT model architecture (which stands for Bidirectional Encoder Representations from Transformers) [2].

Bert is designed to pre-train deep bidirectional representations by jointly conditioning on both left and right context in all layers. The pre-trained BERT representations can be fine tuned with just one additional output layer to create model for question answering system. BERT's model architecture is a multi-layer bidirectional Transformer encoder based on the original implementation described in Vaswani et al.(2017) [3]. BERT is designed to pre-train deep bidirectional representations by jointly conditioning on both left and right context in all layers. BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE benchmark to 80.4 % (7.6% absolute improvement), MultiNLI accuracy to 86.7% (5.6% absolute improvement) and the SQuAD v1.1 question answering Test F1 to 93.2 (1.5 absolute improvement), outperforming human performance by 2.0. BERT addresses the previously mentioned unidirectional constraints by proposing a new pre-training objective: The "masked language model" (MLM), inspired by the Cloze task (Taylor, 1953). The masked language model randomly masks some of the tokens from the input, and the objective is to predict the original vocabulary id of the masked word based only on its context. Unlike left-to-right language model pre-training, the MLM objective allows the representation to fuse the left and the right context, which allows to pre-train a deep bidirectional Transformer. In addition to the masked language model, introduced a "next sentence prediction" task that jointly pre-trains text-pair representations. An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key. Instead of performing a single attention function with dmodel-dimensional keys, values and queries, its beneficial to linearly project the queries, keys and values h times with different, learned linear projections to dk, dk and dv dimensions, respectively. On each of these projected versions of queries, keys and values then perform the attention function in parallel, yielding dv-dimensional output values. The encoder contains self-attention layers. In a self-attention layer all of the keys, values and queries come from the same place, in this case, the output of the previous layer in the encoder. Each position in the encoder can attend to all positions in the previous layer of the encoder. The number of layers (i.e., Transformer blocks) as L, the hidden size as H, and the number of self-attention heads as A. In all cases its set the feed-forward/filter size to be 4H, i.e., 3072 for the H = 768 and 4096 for the H = 1024.

3.2. Bi-Directional Attention Flow (BIDAF) network[4].

A hierarchical multi-stage architecture for modeling the representations of the context paragraph at different levels of granularity (Figure 1). BIDAF includes character-level, word-level, and contextual embeddings, and uses bi-directional attention flow to obtain a query-aware context representation. Attention mechanism offers following improvements to the previously popular attention paradigms. First, attention layer is not used to summarize the context paragraph into a fixed-size vector. Instead, the attention is computed for every time step, and the attended vector at each time step, along with the representations from previous layers, it is allowed to flow through to the subsequent modeling layer. This reduces the information loss caused by early summarization. Second, uses a memory-less attention mechanism. That is, while iteratively computed attention through time as in Bahdanau et al. (2015), the attention at each time step is a function of only the query and the context paragraph at the current time step and does not directly depend on the attention at the previous time step. Its

hypothesize that this simplification leads to the division of labor between the attention layer and the modeling layer.

The machine comprehension model is a hierarchical multi-stage process and consists of six layers.

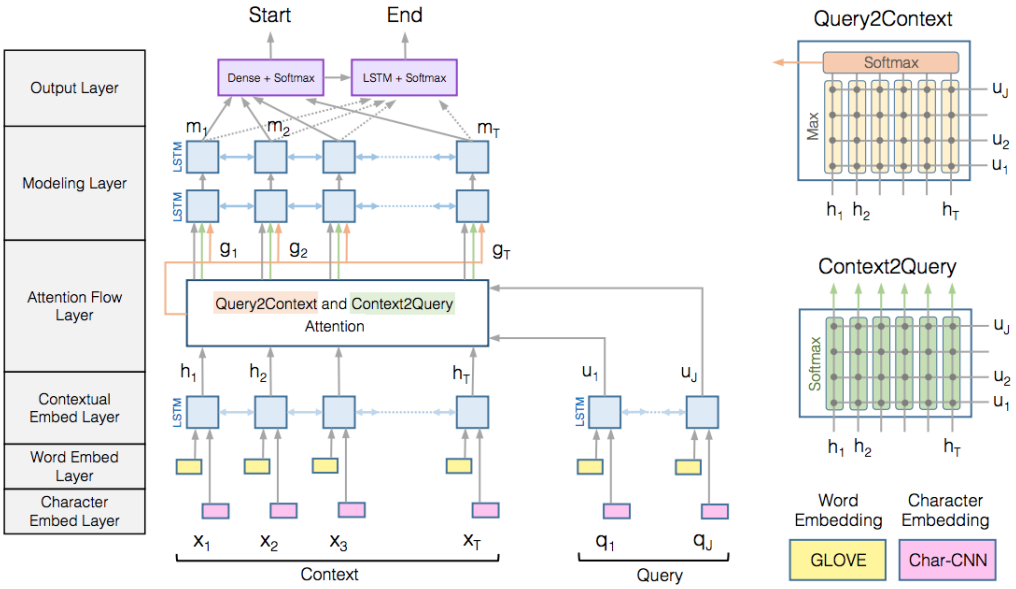


Figure 1: BiDirectional Attention Flow Model *(best viewed in color)*

Figure 1: BiDAF model.

1. Character Embedding Layer maps each word to a vector space using character-level CNNs.

$$g = \sigma(W_g h_i + b_g) \in R^H$$
$$t = ReLU(W_t h_i + b_t) \in R^H$$
$$h'_i = g \odot t + (1 - g) \odot h_i \in R^H,$$

   where $W_g, W_t \in R^{H \times H}$ and $b_g, b_t \in R^H$ are learnable parameters (g is for 'gate' and t is for 'transform').

2. Word Embedding Layer maps each word to a vector space using a pre-trained word embedding model.

$$h'_{i,fwd} = LSTM(h'_{i-1}, h_i) \in R^H$$
$$h'_{i,rev} = LSTM(h'_{i+1}, h_i) \in R^H$$
$$h'_i = \left[h'_{i,fwd}; h'_{i,rev}\right] \in R^{2H}$$

   h'_i is of dimension $2H$, as it is the concatenation of forward and backward hidden states at timestep i.

3. Contextual Embedding Layer utilizes contextual cues from surrounding words to refine the embedding of the words. These first three layers are applied to both the query and context.

4. Attention Flow Layer couples the query and context vectors and produces a set of queryaware feature vectors for each word in the context. Context-to-Question (C2Q) Attention. Take the row-wise softmax of S to obtain attention distributions $\bar{S}$, which used to take weighted sums of the question hidden states q_j , yielding C2Q attention outputs a_i.

$$\bar{S}_{i,:} = softmax(S_{i,:}) \in R^M \qquad \forall_i \in \{1, ..., N\}$$

$$a_i = \sum_{j=1}^{M} \bar{S}_{i,j} q_j \in R^{2H} \qquad \forall_i \in \{1, ..., N\},$$

5. Modeling Layer employs a Recurrent Neural Network to scan the context.

$$m_{i,fwd} = LSTM(m_{i-1}, g_i) \in R^H$$

$$m_{i,rev} = LSTM(m_{i+1}, g_i) \in R^H$$
$$m_i = \Big[m_{i,fwd}; m_{i,rev}\Big] \in R^{2H}$$

The Modeling layer differs from the Encoder layer in that used a one-layer LSTM in the Encoder layer, whereas used a two-layer LSTM in the Modeling layer.

6. Output Layer provides an answer to the query.

$$m'_{i,fwd} = LSTM(m'_{i-1}, m_i) \in R^H$$
$$m'_{i,rev} = LSTM(m'_{i+1}, m_i) \in R^H$$
$$m'_i = \Big[m'_{i,fwd}; m'_{i,rev}\Big] \in R^{2H}$$

To finally produce p_{start} and p_{end}, the output layer computes

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

where $W_{start}, W_{end} \in R^{1\times 10H}$ are learnable parameters.

Self Attention[13].

Most competitive neural sequence transduction models have an encoder-decoder structure [9,10,11]. Here, the encoder maps an input sequence of symbol representations $(x_1, ..., x_n)$ to a sequence of continuous representations $z = (z_1, ..., z_n)$. Given z, the decoder then generates an output sequence $(y_1, ..., y_m)$ of symbols one element at a time. At each step the model is auto-regressive [12], consuming the previously generated symbols as additional input when generating the next. The Transformer follows this overall architecture using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder, shown in the left and right halves of Figure 2, respectively.

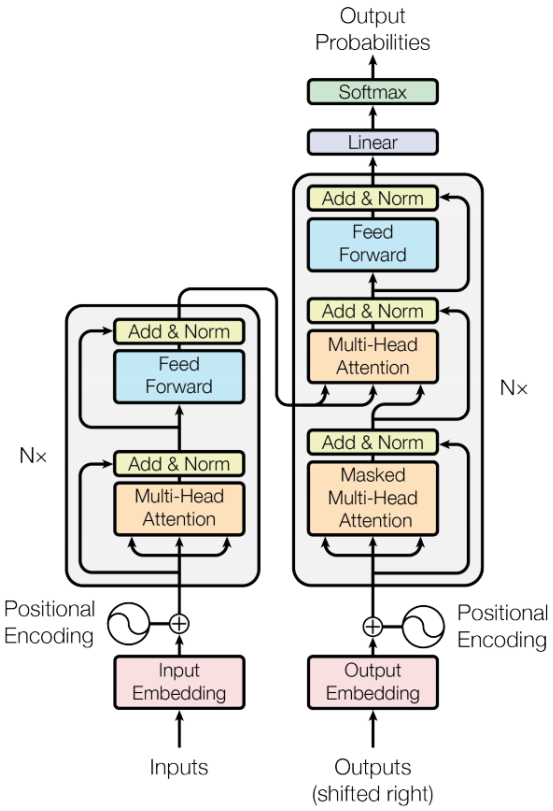


Figure 2: Transformer model.

An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key. "Scaled Dot-Product Attention" input consists of queries and keys of dimension d_k, and values of dimension d_v. Computed the dot products of the query with all keys, divide each by $\sqrt{d_k}$, and apply a softmax function to obtain the weights on the values. Computed the attention function on a set of queries simultaneously, packed together into a matrix Q. The keys and values are also packed together into matrices K and V . Computed the matrix of outputs as:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

3.3. QANET: COMBINING LOCAL CONVOLUTION WITH GLOBAL SELF-ATTENTION FOR READING COMPREHENSION[5].

The high level structure of the model is similar to most existing models that contain five major components: an embedding layer, an embedding encoder layer, a context-query attention layer, a model encoder layer and an output layer, as shown in Figure 2. These are the standard building blocks for most, if not all, existing reading comprehension models. However, the major differences between the approach and other methods are as follow: For both the embedding and modeling encoders, only used convolutional and self-attention mechanism, discarding RNNs, which are used by most of the existing reading comprehension models. As a result, model is much faster, as it can process the input tokens in parallel. Note that even though self-attention has already been used extensively in Vaswani et al. (2017a), the combination of convolutions and self-attention is novel, and is significantly better

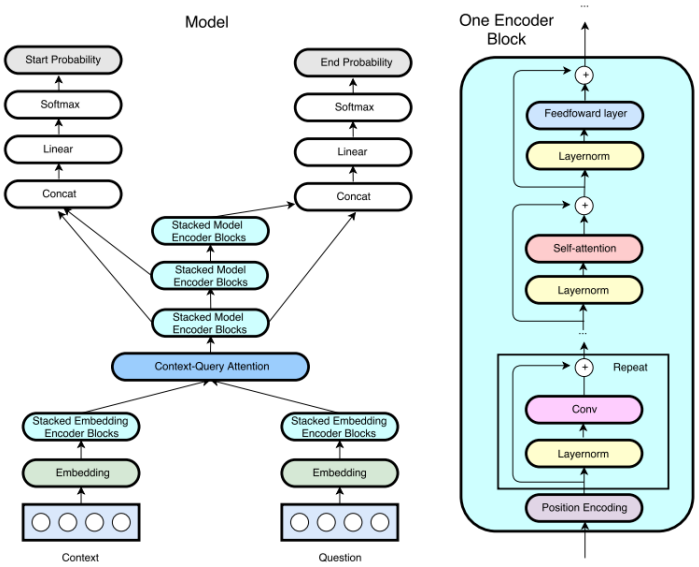


Figure 3: An overview of the QANet architecture (left) which has several Encoder Blocks. The same Encoder Block (right) used throughout the model, only varying the number of convolutional layers for each block.Also used layernorm and residual connection between every layer in the Encoder Block. Also shared weights of the context and question encoder, and of the three output encoders. A positional encoding is added to the input at the beginning of each encoder layer consisting of sin and cos functions at varying wavelengths, as defined in (Vaswani et al., 2017a). Each sub-layer after the positional encoding (one of convolution, self-attention, or feed-forward-net) inside the encoder structure is wrapped inside a residual block.

than self-attention alone and gives 2.7 F1 gain in main experiments. The use of convolutions also allows to take advantage of common regularization methods in ConvNets such as stochastic depth (layer dropout) (Huang et al., 2016), which gives an additional gain of 0.2 F1 in the experiments.

In detail, the model consists of the following five layers:

1. Input Embedding Layer.

All the out-of-vocabulary words are mapped to an <UNK> token, whose embedding is trainable with random initialization. The character embedding is obtained as follows: Each character is represented as a trainable vector of dimension p2 = 200, meaning each word can be viewed as the concatenation of the embedding vectors for each of its characters. The length of each word is either truncated or padded to 16. We take maximum value of each row of this matrix to get a fixed-size vector representation of each word. Finally, the output of a given word x from this layer is the concatenation $[x_w; x_c] \in R^{p_1+_2}$ where x_w and x_c are the word embedding and the convolution output of character embedding of x respectively.

2. Embedding Encoder Layer. For the self-attention-layer, adopted the multi-head attention mechanism defined in (Vaswani et al., 2017a) which, for each position in the input, called the query, computes a weighted sum of all positions, or keys, in the input based on the similarity between the query and key as measured by the dot product. The number of heads is 8 throughout all the layers. Each of these basic operations (conv/self-attention/ffn) is placed inside a residual block, shown lower-right in Figure 1. For an input x and a given operation f, the output is $f(layernorm(x)) + x$, meaning there is a full identity path from the input to output of each block, where layernorm indicates layer-normalization proposed in (Ba et al., 2016).

3. Context-Query Attention Layer.

The context-to-query attention is constructed as follows: First computer the similarities between each pair of context and query words, rendering a similarity matrix $S \in R^{n \times m}$. Then normalized each

row of S by applying the softmax function, getting a matrix $\bar{S}$. Then the context-to-query attention is computed as $A = \bar{S} \cdot Q^T \in R^{n \times d}$. The similarity function used here is the trilinear function (Seo et al., 2016):

$$f(q, c) = W_0[q, c, q \odot c]$$

More concretely, computed the column normalized matrix $\bar{\bar{S}}$ of S by softmax function, and the query-to-context attention is $B = \bar{S} \cdot \bar{\bar{S}}^T \cdot C^T$.

4. Model Encoder Layer.

Similar to Seo et al. (2016), the input of this layer at each position is $[c, a, c \odot a, c \odot b]$, where a and b are respectively a row of attention matrix A and B.

5. Output layer. This layer is task-specific. Each example in SQuAD is labeled with a span in the context containing the answer. The probabilities of the starting and ending position are modeled as.

$$p^1 = softmax(W_1[M_0; M_1]), p^2 = softmax(W_2[M_0; M_2]),$$

where W_1 and W_2 are two trainable variables and M_0, M_1, M_2 are respectively the outputs of the three model encoders, from bottom to top. Finally, the objective function is defined as the negative sum of the log probabilities of the predicted distributions indexed by true start and end indices, averaged over all the training examples:

$$L(\theta) = -\frac{1}{N} \sum_{1}^{N} \Big[log(p^1_{y^1_i}) + log(p^2_{y^2_i}) \Big]$$

4 Experiments

In the first stage I trained the base model for BiDaf and then I added character level embedings to the BiDaf model, also I did fine tuning of Bert. Results can be seen under the experiment details section.

4.1 Data

For this project the official SQuAD 2.0 dataset will be used [1]. It has three splits: train, dev and test. The train and dev sets are publicly available and the test set is entirely secret. The data contains the following splits

- train (129941 examples): from official SQuAD 2.0 training set.
- dev (6078 examples): about half of the official dev set, selected randomly.
- test (5921 examples): the remaining examples from the dev set, plus hand-labeled examples.

For this training has been used the datasets which were generated by squad load script from official datasets.

4.2 Evaluation method

To evaluate the models two metrics will be used. The EM score, which indicates whether there is an exact match between the prediction and the ground truth, and the F1 score, which rewards partial matches between the prediction and the ground truth.Will chose model parameters which are maximizing the F1 score on the development set.

4.3 Experimental details

- BiDAF baseline training run with. Evaluation steps 50,000, Learning rate. 0.5, L2 weight decay. 0, Number of epochs for which to train. 30, Probability of zeroing an activation in dropout layers. 0.2, Name of dev metric to determine best checkpoint. F1, Maximum number of checkpoints to keep on disk. 5, Maximum gradient norm for gradient clipping. 5.0, Random seed for reproducibility. 224, Decay rate for exponential moving average of

parameters. 0.999, Maximum length of a predicted answer. 15, Number of sub-processes to use per data loader. 4, Batch size per GPU. Scales automatically when multiple GPUs are available. 64, Number of features in encoder hidden layers. 100, Number of examples to visualize in TensorBoard. 10. Training process took around 8-10 hours.

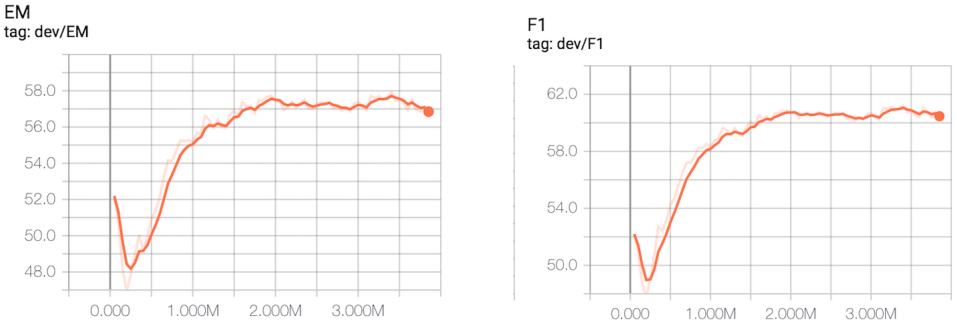
- BiDAF with char embedings training run with. Evaluation steps 50,000, Learning rate. 0.5, L2 weight decay. 0, Number of epochs for which to train. 30, Probability of zeroing an activation in dropout layers. 0.2, Name of dev metric to determine best checkpoint. F1, Maximum number of checkpoints to keep on disk. 5, Maximum gradient norm for gradient clipping. 5.0, Random seed for reproducibility. 224, Decay rate for exponential moving average of parameters. 0.999, Maximum length of a predicted answer. 15, Number of sub-processes to use per data loader. 4, Batch size per GPU. Scales automatically when multiple GPUs are available. 64, Number of features in encoder hidden layers. 100, Number of examples to visualize in TensorBoard. 10. Training process took around 8-10 hours.

- BiDAF with char embedings with self attention training run with. Evaluation steps 50,000, Learning rate. 0.5, L2 weight decay. 0, Number of epochs for which to train. 30, Probability of zeroing an activation in dropout layers. 0.2, Name of dev metric to determine best checkpoint. F1, Maximum number of checkpoints to keep on disk. 5, Maximum gradient norm for gradient clipping. 5.0, Random seed for reproducibility. 224, Decay rate for exponential moving average of parameters. 0.999, Maximum length of a predicted answer. 15, Number of sub-processes to use per data loader. 4, Batch size per GPU. 32, Number of features in encoder hidden layers. 100, Number of examples to visualize in TensorBoard. 10. Training process took around 8-10 hours.

4.4 Results

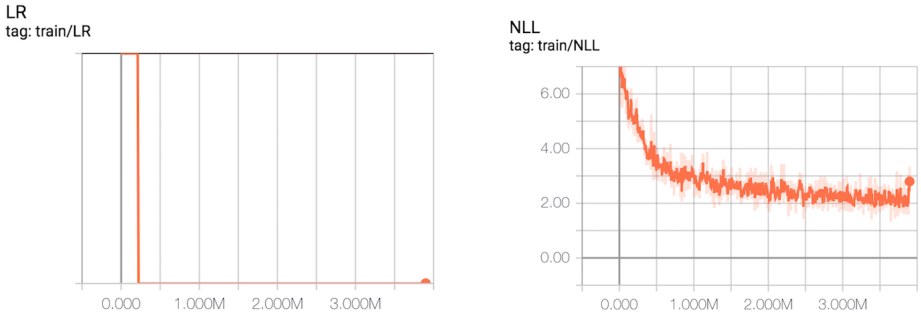
1. Bidaf baseline.

Name	Split	Description
EM	Dev	57.92
F1	Dev	61.19



(a) EM Score (b) F1 Score

Figure 4: Dev Dataset

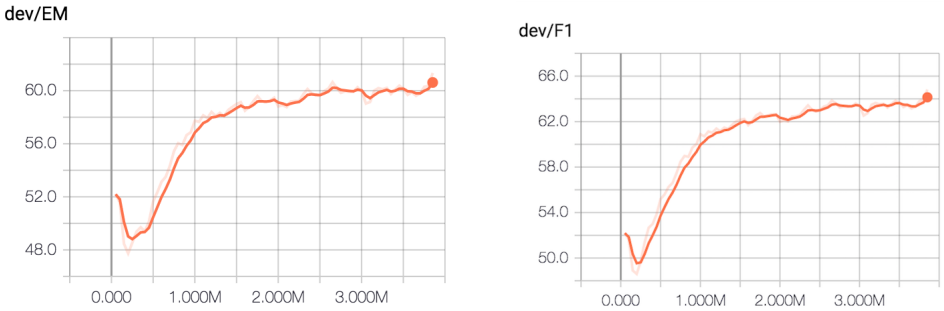


(a) LR (b) NLL

Figure 5: Train Dataset

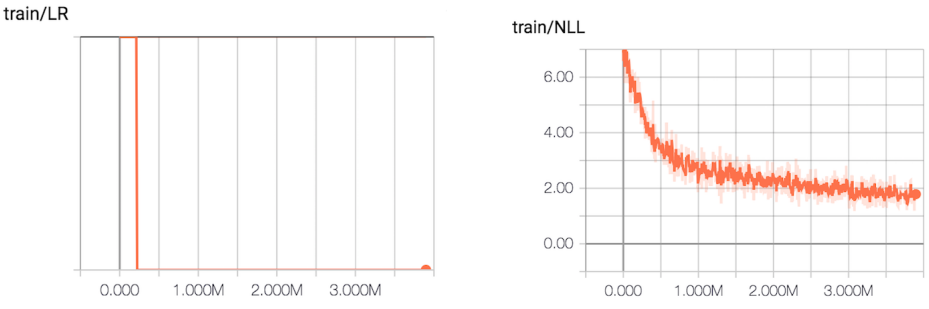
2. Bidaf with char embedings.

Name	Split	Description
EM	Dev	61.334
F1	Dev	64.822



(a) EM Score (b) F1 Score

Figure 6: Dev Dataset

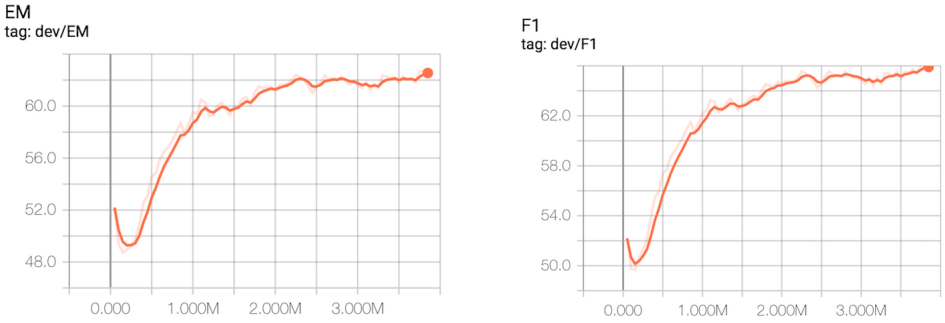


(a) LR (b) NLL

Figure 7: Train Dataset

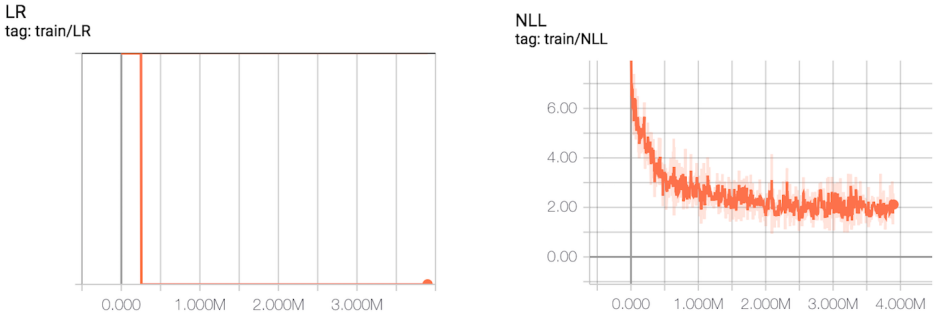
2. Bidaf with char embedings with self attention.

Name	Split	Description
EM	Dev	62.847
F1	Dev	66.267



(a) EM Score (b) F1 Score

Figure 8: Dev Dataset

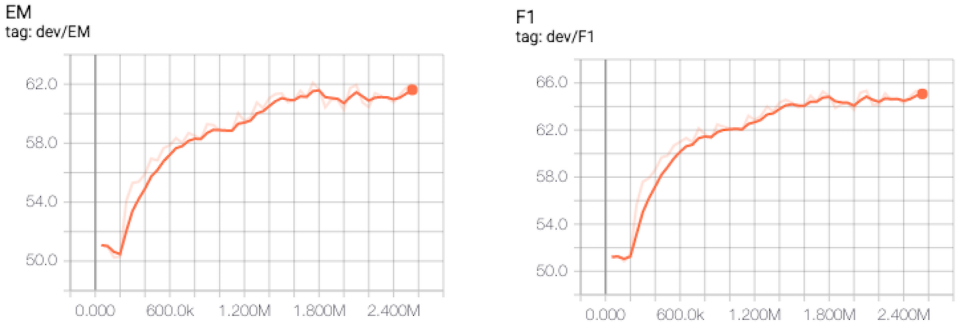


(a) LR (b) NLL

Figure 9: Train Dataset

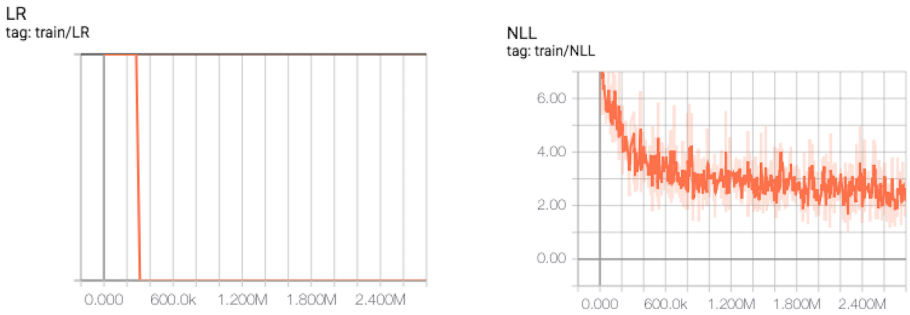
3. QANet.

Name	Split	Description
EM	Dev	58.71
F1	Dev	62.97



(a) EM Score (b) F1 Score

Figure 10: Dev Dataset



(a) LR (b) NLL

Figure 11: Train Dataset

4. Bert.

Name	Split	Description
EM	Dev	72.836
F1	Dev	76.068

5 Result Analysis

I get the following results on my best model F1=64.771, EM=61.302 on test set. I did my analysis step by step first I run baseline BiDAF model and achieved the following results F1=61.19, EM=57.92 then I tried the following versions.

1. BiDAF with char embedding (F1=64.822, EM=61.334). 2. BiDAF with char embedding and self attention (best result) (F1=66.267, EM=62.847). 3. BiDAF with char embedding and multi head attention (not very good results) (F1=61.89, EM=58.78). 4. QANet (F1=62.97, EM=58.71). 5. BERT (F1=76.068, EM=72.836).

I tried a few versions of multi head attention but not get good results, also I tried QANet with different params but still didn't get good results comparing to my best model, also I used BERT model which of course has better results but BiDAF gave me more place for research so I continued on non PCE models.

5.1 Analysis

Adding char embedding gave me about 2.22% improvement above baseline, and adding self attention gave me about 3.1% improvement over baseline the following parameters were used to train BiDAF with char embedding and self attention model.

Evaluation steps 50,000, Learning rate. 0.5, L2 weight decay. 0, Number of epochs for which to train. 30, Probability of zeroing an activation in dropout layers. 0.2, Name of dev metric to determine best checkpoint. F1, Maximum number of checkpoints to keep on disk. 5, Maximum gradient norm for gradient clipping. 5.0, Random seed for reproducibility. 224, Decay rate for exponential moving average of parameters. 0.999, Maximum length of a predicted answer. 15, Number of sub-processes to use per data loader. 4, Batch size per GPU. 32, Number of features in encoder hidden layers. 100, Number of examples to visualize in TensorBoard. 10.

5.2 Error analysis

I analyzed randomly selected 200 examples from which 78 has errors I categorized the errors.

1. Non precise answer boundaries, about 19% of errors falls into this category here some examples.

- **Question:** Where is Polonia's home venue located?
- **Context:** Their local rivals, Polonia Warsaw, have significantly fewer supporters, yet they managed to win Ekstraklasa Championship in 2000. They also won the country's championship in 1946, and won the cup twice as well. Polonia's home venue is located at Konwiktorska Street, a ten-minute walk north from the Old Town. Polonia was relegated from the country's top flight in 2013 because of their disastrous financial situation. They are now playing in the 4th league (5th tier in Poland) -the bottom professional league in the National – Polish Football Association (PZPN) structure.
- **Answer:** at Konwiktorska Street
- **Prediction:** Konwiktorska Street

- **Question:** Where did one of Triton's daughters decide she wanted to hang out and stay?
- **Context:** The origin of the legendary figure is not fully known. The best-known legend, by Artur Oppman, is that long ago two of Triton's daughters set out on a journey through the depths of the oceans and seas. One of them decided to stay on the coast of Denmark and can be seen sitting at the entrance to the port of Copenhagen. The second mermaid reached the mouth of the Vistula River and plunged into its waters. She stopped to rest on a sandy beach by the village of Warszowa, where fishermen came to admire her beauty and listen to her beautiful voice. A greedy merchant also heard her songs; he followed the fishermen and captured the mermaid.
- **Answer:** coast of Denmark
- **Prediction:** on the coast of Denmark

- **Question:** Telnet was sold to
- **Context:** Telenet was the first FCC-licensed public data network in the United States. It was founded by former ARPA IPTO director Larry Roberts as a means of making ARPANET technology public. He had tried to interest AT&T in buying the technology, but the monopoly's reaction was that this was incompatible with their future. Bolt, Beranack and Newman (BBN) provided the financing. It initially used ARPANET technology but changed the host interface to X.25 and the terminal interface to X.29. Telenet designed these protocols and helped standardize them in the CCITT. Telenet was incorporated in 1973 and started operations in 1975. It went public in 1979 and was then sold to GTE.
- **Answer:** Telenet was incorporated in 1973 and started operations in 1975. It went public in 1979 and was then sold to GTE
- **Prediction:** GTE

These predictions are usually right from human point of view but if we want to predict exactly need to have a few answers for the question.

2. Syntax ambiguities.

In this example for human is also difficult to define “Where did Molcalm slip away to attack”. And for model predicted that he slip away in Albany and then led the successful attack ion Oswego. In this case to avoid ambiguities maybe need to have more clarified Context or add additional input features to the model that will take account ambiguities.

- **Question:** Where did Moncalm slip away to attack, left largely unprotected?
- **Context:** The new British command was not in place until July. When he arrived in Albany, Abercrombie refused to take any significant actions until Loudoun approved them. Montcalm took bold action against his inertia. Building on Vaudreuil's work harassing the Oswego garrison, Montcalm executed a strategic feint by moving his headquarters to Ticonderoga, as if to presage another attack along Lake George. With Abercrombie pinned down at Albany, Montcalm slipped away and led the successful attack on Oswego in August. In the aftermath, Montcalm and the Indians under his command disagreed about the disposition of prisoners' personal effects. The Europeans did not consider them prizes and prevented the Indians from stripping the prisoners of their valuables, which angered the Indians.
- **Answer:** Oswego
- **Prediction:** Albany

3. Predicted as N/A.

About 24% predicted as "Not have an answer N/A" but they have answers.

- **Question:** What two member nations of the Holy Roman Empire received Huguenot refugees?
- **Context:** The bulk of Huguenot émigrés relocated to Protestant European nations such as England, Wales, Scotland, Denmark, Sweden, Switzerland, the Dutch Republic, the Electorate of Brandenburg and Electorate of the Palatinate in the Holy Roman Empire, the Duchy of Prussia, the Channel Islands, and Ireland. They also spread beyond Europe to the Dutch Cape Colony in South Africa, the Dutch East Indies, the Caribbean, and several of the English colonies of North America, and Quebec, where they were accepted and allowed to worship freely.
- **Answer:** Electorate of Brandenburg and Electorate of the Palatinate
- **Prediction:** N/A

Maybe need to improve "Predicting no-answer" approach to allow low probability spans with some threshold to participate in answer creation.

6 Conclusion

For this project I have reimplemented the BiDAF layer adding also self-attention layer. I showed how important was attention, because its increased scores comparing with baseline.I also trained BERT and QANET, I didnt have much improvements there because fine tuning process faced with memory issues, especially increasing batch size immediately giving not enough memory issue in GPU.Overall I am rather satisfied by the model as it has achieved good results. There are still many improvements to be made, and I have trained with no such many tuned hyperparameters. My approach was to train with different models.

References

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

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018. https://arxiv.org/pdf/1810.04805.pdf

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In Advances in Neural Information Processing Systems, pages 6000–6010.

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

[5] Adams Wei Yu, David Dohan, ,Minh-Thang Luong weiyu@cs.cmu.edu, ddohan,thangluong@google.com, Carnegie Mellon University, Google Brain Rui Zhao, Kai Chen, Mohammad Norouzi, Quoc V. Le Google Brain. QANET: COMBINING LOCAL CONVOLUTION WITH GLOBAL SELF-ATTENTION FOR READING COMPREHENSION.

[6] O. Levy, M. Seo, E. Choi, and L. Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In Computational Natural Language Learning (CoNLL)

[7] C. Clark and M. Gardner. 2017. Simple and effective multi-paragraph reading comprehension. arXiv preprint arXiv:1710.10723 .

[8] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. 2018. Deep contextualized word representations. In North American Association for Computational Linguistics (NAACL).

[9] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. CoRR, abs/1406.1078, 2014.

[10] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. CoRR, abs/1409.0473, 2014.

[11] Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Sequence to sequence learning with neural networks. In Advances in Neural Information Processing Systems, pages 3104–3112, 2014.

[12] Alex Graves. Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850, 2013.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin Attention Is All You Need. arXiv:1706.03762