Project Report : E1-246
Reading Comprehension System for SQuAD 2.0

Aadesh Magare
aadeshmagare@iisc.ac.in

Abhishek Kumar
abhishekkumar@iisc.ac.in

Abstract

Reading comprehension is an exciting problem for Natural language uderstanding. Lots of deep learning models have been built for the SquAD 2.0 (Rajpurkar et al., 2018) dataset, which has become very popular in the community. Given a context passage and a question, a model must predict the start and end indices in the context as an answer, or the model predict no answer. We have used QANnet (Yu et al., 2018) architecture to solve the problem which replaces RNNs with Transformer-like encoder blocks.

1 Introduction

Question Answering is a popular sub-field in the domain of Natural Language Understanding. In particular, Reading Comprehension (RC), the ability to read text and answer questions about it is a challenging task as it involves understanding the language along with some knowledge about the world. Stanford Question Answering Dataset (SQuAD 2.0) is a reading comprehension dataset. The model is given a paragraph and questions about that paragraph as input. The baseline model uses BiDAF (Seo et al., 2016) based architecture to predict if the question can be answered or not based on the information from the comprehension but training these models from scratch requires lots of data and computational resource and time. This motivated us to look into methods avoiding recurrence and were intrigued by QANet which avoids RNNs completely. Our major focus is understanding the limitations of QANet against SQuAD 2.0 dataset and investigating methods to augment the architecture.

Problem Statement: The reading comprehension task considered in this paper, is defined as follows. Given a context paragraph with n words $C = \{c_1, c_2, ..., c_n\}$ and the query sentence with m words $Q = \{q_1, q_2, ..., q_m\}$ output a span S = $\{c_i, c_{i+1}, ..., c_{i+j}\}$ from the original paragraph C which represents answer to the query sentence. x is used to denote both the original word and its embedded vector, for any x $\in$ C, Q.

2 Related Work

The architecture of our model comes from QANet [1] paper. Prior to QANet, major question answering systems primarily contained either of two key ingredients (1) recurrent units such as LSTM for capturing sequential input and (2) exploiting attention mechanism for capturing long-term interaction. Our baseline BiDAF [2] model uses both of these techniques. It is a hierarchical multi-stage end-to-end network which takes inputs as word embedding to obtain a query aware context representation using memoryless context-to-query (C2Q) and query-to-context (Q2C) attention. We have used multi-head attention mechanism which is based on the “Attention is All You Need”[3] paper.

3 Datasets and Metrics

3.1 Dataset

Stanford Question Answering Dataset (SQuAD) is a reading comprehension dataset, consisting of questions posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage, or the question might be unanswerable.

SQuAD2.0 (Rajpurkar et al., 2018) combines the 100,000 questions in SQuAD1.1 with over

[1] QANet: Combining Local Convolution With Global Self-Attention for reading Comprehension : arxiv.org/pdf/1804.09541.pdf

[2] Bidirectional Attention Flow for Machine Comprehension : arxiv.org/pdf/1611.01603.pdf

[3] Attention Is All You Need : arxiv.org/pdf/1706.03762.pdf

50,000 new, unanswerable questions written adversarially by crowdworkers to look similar to answerable ones. The typical length of the paragraphs is around 250 while the question is of 10 tokens although there are exceptionally long cases. To do well on SQuAD2.0, systems must not only answer questions when possible, but also determine when no answer is supported by the paragraph and abstain from answering.

The official SQuAD 2.0 dataset has three splits: train, dev and test. The train and dev sets are publicly available and the test set is entirely secret. To compete on the official SQuAD leaderboards, researchers submit their models, and the SQuAD team runs the models on the secret test set.

The SQuAD dataset contains many (context, question, answer) triples. Each context (sometimes called a passage, paragraph or document) is an excerpt from Wikipedia. The question (sometimes called a query) is the question to be answered based on the context. The answer is a span (i.e. excerpt of text) from the context.

Data Preprocessing: NLTK tokenizer is used to preprocess the data. The maximum context length is set to 400 and any paragraph longer than that would be discarded. During training, the examples are batched by length and dynamically padded the short sentences with special symbol <PAD>.

The maximum answer length is set to 30. We use the pretrained 300-D word vectors GLoVe (Pennington et al., 2014), and all the out-of-vocabulary words are replace with <UNK> whose embedding is updated during training. Each character embedding is randomly initialized as a 200-D vector, which is updated in training as well.

3.2 Metrics

Performance is measured via two metrics: **Exact Match (EM)** and **F1 score**.

- **Exact Match** is a binary measure (i.e. true/false) of whether the system output matches the ground truth answer exactly.
- **F1** is a less strict metric, F1 measures the portion of overlap tokens between the predicted answer and groundtruth, it is the harmonic mean of precision and recall.
- When a question has no answer, both the F1 and EM score are 1 if the model predicts NoAnswer, and 0 otherwise.

The EM and F1 scores are averaged across the entire evaluation dataset to get the final reported scores.

4 Baseline

The baseline [4] model is a based on BiDAF (Seo et al., 2016), unlike the original BiDAF model, this implementation does not include a character-level embedding layer. The baseline model has the following architecture:

4.1 Embedding Layer (layers.Embedding)

Given some input word indices $w_1, ..., w_k \in N$ the embedding layer performs an embedding look up to convert the indices into word embeddings $v_1, ..., v_k \in R^D$. This is done for both the context and the question, producing embeddings $c_1, ..., c_N \in R^D$ for the context and $q_1, ..., q_M \in R^D$ for the question. it uses highway network (Srivastava et al., 2015) to further refine the learned word embedding. Given an input vector $\boldsymbol{h_i} \in \boldsymbol{R_H}$, a one way highway network computes:

$$\begin{aligned} \boldsymbol{g} &= \boldsymbol{\sigma(W_g h_i + b_g) \in R^H} \\ \boldsymbol{t} &= \boldsymbol{ReLU(W_t h_i + b_t) \in R^H} \\ \boldsymbol{h'_i} &= \boldsymbol{g \odot t + (1-g) \odot h \in R^H} \end{aligned}$$

We use a two-layer highway network to transform each hidden vector h_i, which means the model apply the above transformation twice.

4.2 Encoder Layer (layers.RNNEncoder)

The encoder layer uses a bidirectional LSTM to allow the model to incorporate temporal dependencies between time steps of the embedding layers output. The encoded output is the RNNs hidden state at each position:

$$\begin{aligned} \boldsymbol{h'_i, fwd} &= \boldsymbol{LSTM(h'_{i-1}, h_i) \in R^H} \\ \boldsymbol{h'_i, rev} &= \boldsymbol{LSTM(h'_{i+1}, h_i) \in R^H} \\ \boldsymbol{h'_i} &= \boldsymbol{[h'_i, fwd; h'_i, rev] \in R^{2H}}. \end{aligned}$$

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

[4] http://web.stanford.edu/class/cs224n/project/default-final-project-handout.pdf

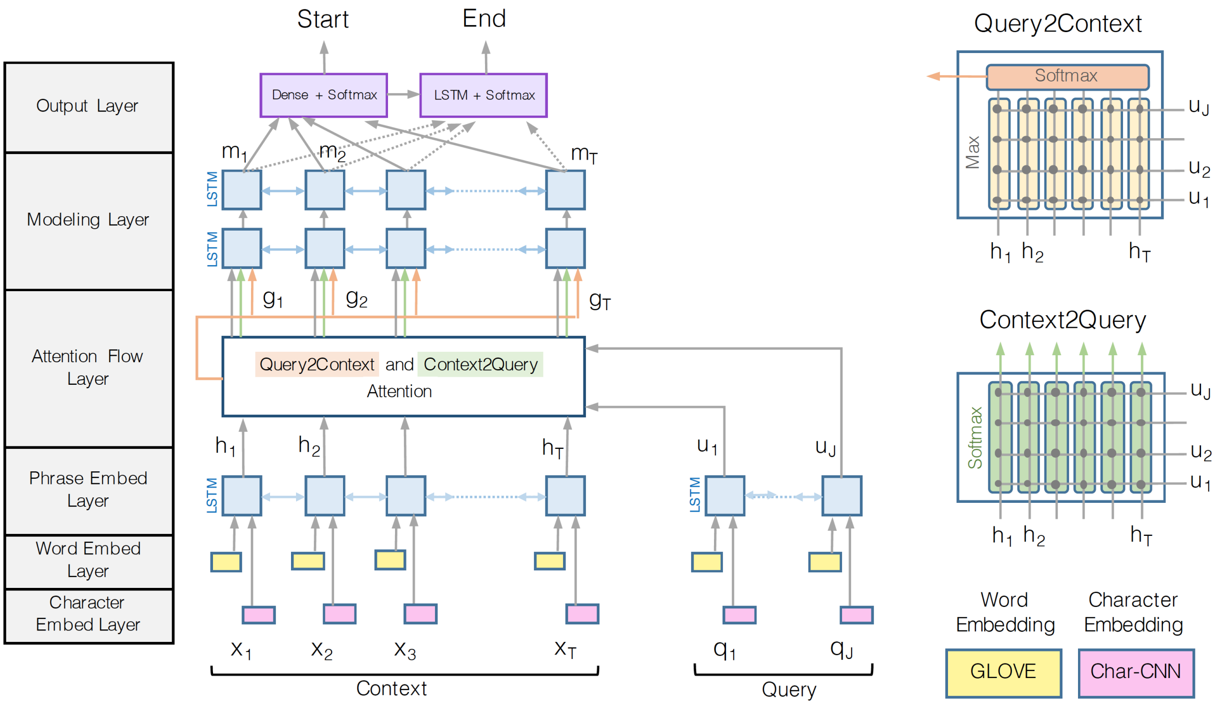


Figure 1: BiDAF Architecture

4.3 Attention Layer (layers.BiDAFAttention)

The core part of the BiDAF model is the BiDirectional Attention Flow layer. The main idea is that attention should flow both ways from the context to the question and from the question to the context.

- First it computes the similarity matrix $S \in R^{NxM}$, which contains a similarity score S_{ij} for each pair (c_i, q_j) of context (c_i) and question (q_j) hidden states.
- Next it performs Context-to-Question (C2Q) Attention and Question-to-Context(Q2C) Attention.
- Lastly, for each context location $i \in 1, ..., N$ it obtain the output g_i of the Bidirectional Attention Flow Layer by combining the context hidden state c_i , the C2Q attention output a_i , and the Q2C attention output b_i

$\boldsymbol{g_i = [c_i; a_i; c_i \cdot a_i; c_i \cdot b_i] \in R^{8H} \forall i \in \{1, ..., N\}}$ where $\cdot$ represents elementwise multiplication.

4.4 Modeling Layer (layers.RNNEncoder)

The modeling layer is tasked with refining the sequence of vectors after the attention layer. It integrates the temporal information between context representations conditioned on the question. Similar to the Encoder layer, It uses a bidirectional LSTM.

4.5 Output Layer (layers.BiDAFOutput)

The output layer is tasked with producing a vector of probabilities corresponding to each position in the context i.e., $p_{start}, p_{end} \in R^N$. p_{start} (i) is the predicted probability that the answer span starts at position i, and similarly p_{end} (i) is the predicted probability that the answer span ends at position i. To allow the model to make no-answer predictions it prepend a OOV token to the beginning of each sequence and When discretizing a prediction, if $p_{start}(0)\ \ p_{end}(0)$ is greater than any predicted answer span, the model predicts no-answer. Otherwise the model predicts the highest probability span as usual.

5 Method

The high level structure of the model is similar to most existing models that contain five layer architecture: an embedding layer, an embedding encoder layer, a context-query attention layer, a model encoder layer and an output layer, as shown in Figure 1.

These are the standard building blocks for most reading comprehension models. However, the major differences in our approach is that for both the embedding and modeling encoders, we only use convolutional and self-attention mechanism, discarding RNNs, which are used by most of the existing reading comprehension models.

5.1 Input Embedding Layer

The embedding of each word w is obtained by concatenating its word embedding and character embedding. The word embedding is p1 = 300 dimensional pre-trained GloVe (Pennington et al., 2014) word vectors, which are fixed during training. All the OOV words are mapped to an <UNK> token, whose embedding is trainable with random initialization.

Characters are 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. A two-layer highway network is also adopted on top of this representation.

5.2 Embedding Encoder Layer

The encoder layer is a stack of the convolution layer, self attention layer and feed forward layer.

In the convolution layer the kernel size is 7, the number of filters is d = 128 and the number of convolution layers within a block is 3. For the self attention layer, multi-head attention mechanism defined in (Vaswani et al., 2017) is used. In which, for each position in the input, called the query, computes a weighted sum of all positions in the input based on the similarity between the query and key as measured by the dot product. Each of these basic operations (convolution/self attention/ffn) is placed inside a residual block.

5.3 Context-Query Attention Layer

C and Q is used to denote the encoded context and query. Similarities between each pair of context and query words is calculated, rendering a similarity matrix $S \in R^{n*m}$. We then normalize each row of S by applying the softmax function, getting a matrix S'. Then the context to query attention is computed as

$$\boldsymbol{A = S' * Q^T \in R^{n*d}}$$

The column normalized matrix S'' of S is computed by softmax function, and the query to context attention is

$$\boldsymbol{B = S' * S''^T * C^T}$$

5.4 Model Encoder Layer

The input of this layer at each position is $[c, a, c * a, c * b]$ similar to (Seo et al., 2016), where a and b are respectively a row of attention matrix A and B. The layer parameters are the same as the Embedding Encoder Layer except that convolution layer number is 2 within a block and the total number of blocks are 7.

5.5 Output layer

An example in SQuAD is labeled with a span in the context containing the answer. The probability of each position in the context being the start or end of an answer span is calculated as:

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

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. The score of a span is the product of its start position and end position probabilities.

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.

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

where y^1_i and y^2_i are respectively the ground truth starting and ending position of example i. At inference time, the predicted span (s, e) is chosen such that $p^1_s * p^2_e$ is maximized and $s \leq e$.

6 Experiments

In this section, we describe the experiments conducted by us to evaluate the performance of the QANet model on the SQuAD [2.0] dataset.

The hardware used for the experiment is NC6_DEMO Virtual Machine of GPU machines provided by Microsoft Azure For Students with 12GB of GPU memory. It was observed that close 11.9 GB of GPU memory was used. The QANet model was manually stopped after 5 epochs in the interest of training time. The model was run with batch size of 8 due to memory constraints as opposed to the size of 32 in paper.

Also, as specified in the paper, we use different regularization techniques.

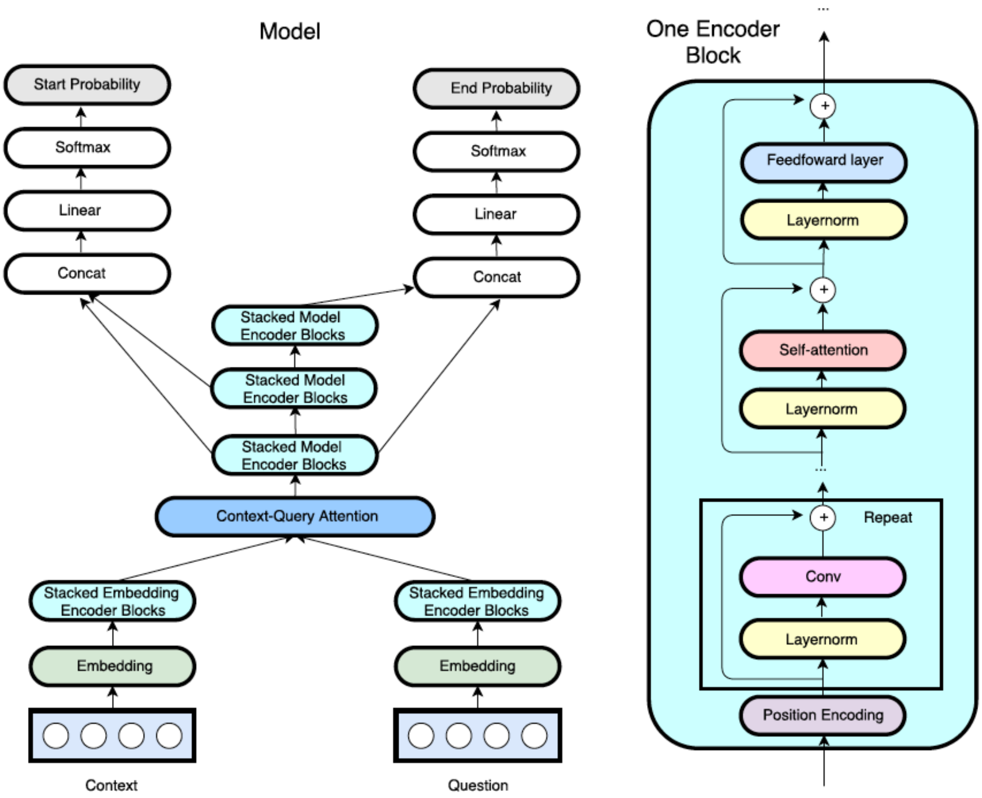


Figure 2: QANet Architecture.

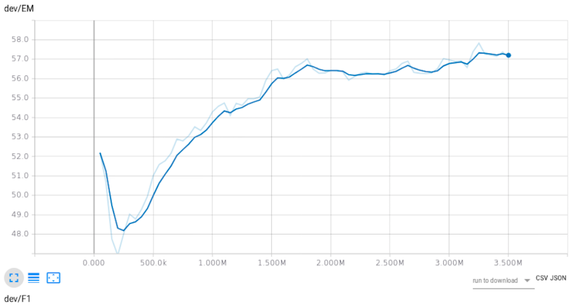


Figure 3: QANet Dev EM Score

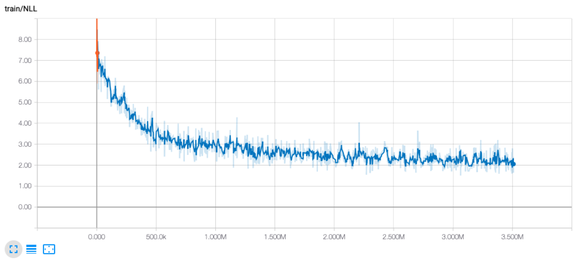
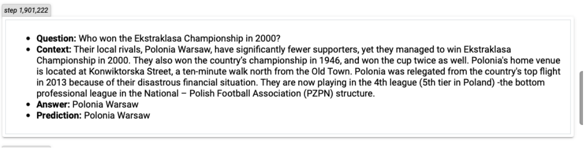


Figure 4: QANet Training Loss


step 1,901,222

- **Question:** Who won the Ekstraklasa Championship in 2000?
- **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:** Polonia Warsaw
- **Prediction:** Polonia Warsaw

Figure 5: QANet: Example

1. L2 weight decay on all trainable kernels with $\lambda = 3e^{-7}$

2. Dropout with drop_prob = 0.1 after every other layer. Additionally dropout is applied word embedding and character embedding with drop_prob of 0.1 and 0.05 respectively.

7 Results

The model achieves scores of 57.77 / 71.19 of EM and F1 respectively on dev dataset on SQuAD [2.0]. The improvement over baseline model is not by a large margin, as claimed in paper, we suspect that this is due to a combination of early stopping of training due to lack of resources and some sub-optimal implementation. Comparison of results can be seen in the 1.

Model	EM Score	F1 Score
Baseline-BiDAF	56.57	60.25
QANet	57.77	71.19

Table 1: Result comparison

8 Discussions

The main highlight of the model is increase in training speed, on the SQuAD dataset, the model is 5x faster in training than the baseline. while achieving an equivalent accuracy. The speed-up allows to train the model with much more data. QANet authors had experimented with data generated by back-translation from a neural machine translation model, and shown to have significant impact on overall performance of model.

9 Future Work

QANet is a fast and accurate end-to-end model, for machine reading comprehension. The core innovation was to completely remove the recurrent networks in the encoder. The resulting model is fully feed-forward, composed entirely of separable convolutions, attention, linear layers, and layer normalization, which is suitable for parallel computation.

One limitation is the memory requirement on the GPUs in training. In the future, we could look for ways to improve the memory efficiency of the model and also try out different ensemble of models. Lastly, it may be premature to completely do away with recurrent models, it can be incorporated before or after transformers and can help with training and perhaps performance as well. It can be worthwhile to experiment with it.

10 Acknowledgements

We would like to thank Chris Chute for providing the starter kit [5] of BiDAF model and Azure tips handout, also Prof Partha Talukdar, the course instructor for E1-246 and Sawan, course TA.

The code for project is available at GitHub. [6].

[5] https://github.com/chrischute/squad

[6] https://github.com/Aadesh-Magare/QANet-for-SQuAD-2.0

References

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *In EMNLP*.

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

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

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *CoRR*, abs/1505.00387.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. 2018. Qanet: Combining local convolution with global self-attention for reading comprehension. *CoRR*, abs/1804.09541.