R-NET based Neural Network for Machine Reading Comprehension

Balaji Kollu (vbkollu) Abhishek Bharani (abharani)

March 21, 2018

1 Introduction

Question answering is a special task of NLP , where computer system is presented with context paragraph and based on the query or question, the system predicts the answer. Here, we focus on reading comprehension style question answering system mainly focused on SQuAD [2], large scale dataset for reading comprehension and question answering. SQuAD requires answer to be present in the context paragraph. With the introduction of attention mechanism in the recurrent neural networks, it is possible to build systems able to comprehend the text. The Squad challenge is very competetive and many novel methods are proposed with their performance near to human level performance.

For this assignment we tried Simple Attention Model(Baseline), Bidirectional attention flow(BiDAF), Coattention and Self-attention (R-NET). We provide performance analysis of three models, visualization of data , attention output, effect of different dropouts and explain in deep our best performing model (R-NET) out of three.

2 Related Work

There has been lot of research done and many end to end neural network models proposed. Task of reading comprehension by machine, answering to the questions from the context, is really challenging for the machines and has received much attention in recent years. Since the release of SQuAD dataset by Rajpurkar [2], which provides large number of real questions and answers, lot of work has been done on building deep learning systems.

A standard neural network consists of a series of non-linear transformation layers and produces hidden representations for whole sentence. For tasks with large input spaces, it makes hard to control the interaction between components. Wang and Jiang [4] built end-to-end neural network based on match-LSTM to generate question aware context representations. Natural Language Computing Group, Microsoft Research Asia proposed R-NET [1], new end-to-end neural network model, which includes self-attention mechanism to question aware context representations where context is matched against itself to refine context representations. We are inspired by R-net model, and implemented self-attention and gating mechanism onto baseline model.

3 Dataset

Dataset used for this task is Stanford Question Ansering Dataset(SQuAD). The reading passages in SQuAD are from high-quality wikipedia articles, and cover a diverse range of topics across a variety of domains, from music celebrities to abstract concepts. A passage is a paragraph from an article, and is variable in

[1]https://www.microsoft.com/en-us/research/wp-content/uploads/2017/05/r-net.pdf

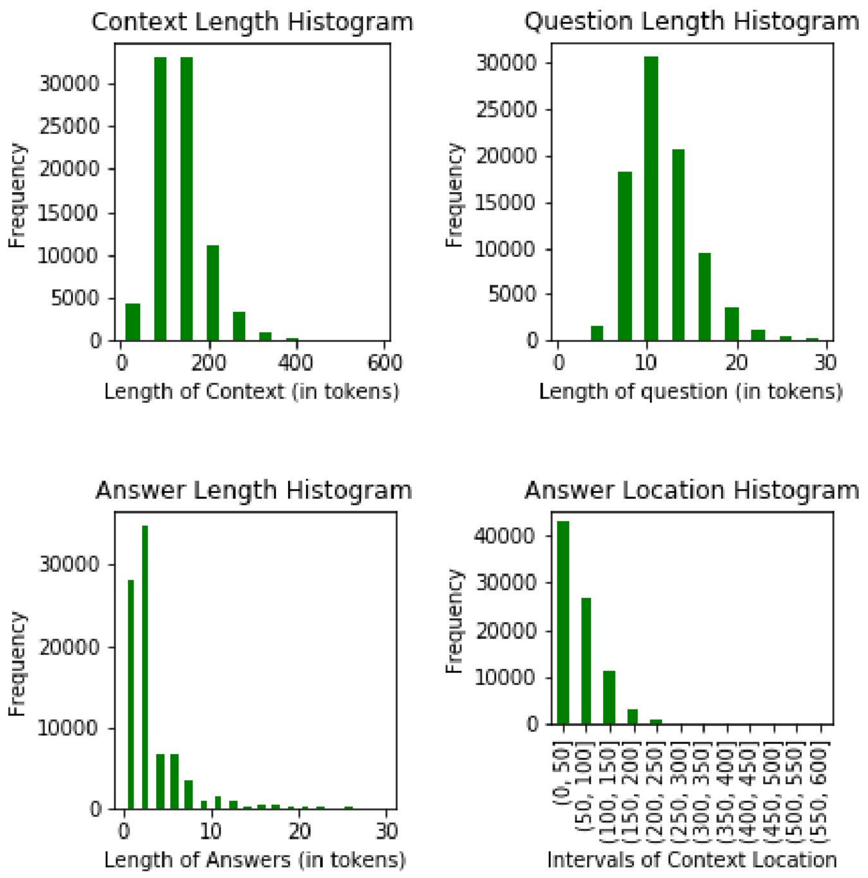


Figure 1: Histogram Pictures

length. Each passage in SQuAD has accompanying reading comprehension questions. These questions are based on the content of the passage and can be answered by reading through the passage. Finally, for each question, we have one or more answers. One defining characteristic of SQuAD is that the answers to all of the questions are segments of text, or spans, in the passage. These can be single or multiple words, and are not limited to entities any span is fair game. The dataset contains more than 100,000 question-answer pairs on 500+ articles. The histograms of the question, context, and answer length of the training set can be seen in Figure 1.

From the histogram plots we observed that more than 99% of the contexts are of length less than 350. Also answer location in majority of the anwers is in begining of the context.

4 Models and different Approaches

We took incremental development approach on the baseline model provided in assignment4 of CS224N.

- BiDAF We tried with BiDAF with word embedding by adding context to question and question to context attention layer to baseline model.
- Coattention We implemented the coattention layer and dynamic decoder using LSTM.

- RNET On baseline model, we implemented self-attention model with word embedding followed by adding gating strategy and character level embedding to that.
- FUNE TUNNING We later applied additional fine-tuning strategies.

We will describe our final model which is customized version of R-net model implemented in the paper.

4.1 BiDAF

Another successful model for reading comprehension is the Bidirectional Attention Flow model. This model introduces a bidirectional attention ow (BiDAF) mechanism to obtain query-aware context and context-aware query without early summarization.

4.2 Coattention

Dynamic Coattention Networks [5] model consists of document and question encoder, coattention encoder, and dynamic pointer decoder. The coattention encoder is the attention mechanism for this model. It encodes the interaction between the encoded question and the encoded document. The dynamic pointing decoder uses highway maxout network to compute the probability of the start index and the end index. The process is repeated a few times, using the information of the previous prediction to improve the next prediction. This iterative process allows the model to recover from from a local maxima.

The model consists of document and question encoder, gated matching layer to match question and context, self attention over document itself and pointer network. For gated matching layer we use the output of Basic Attention model and perform self attention using additive or tanh attention.

4.3 R-NET

4.3.1 Basic Understanding and Motivation

R-Net model is based on assumption that most of questions are paraphrase of sentences from the context. A gated matching layer matches the question and context and self matching layer on context to fetch more information from context itself and pointer based answer boundary prediction. The gated mechanism assigns different level of importance to context parts depending on question taking out irrelevant information and just focusing on the one dependent on question. Since the recurrent networks can only memorize limited context, the possible answer will have no information for other parts of context. To solve this we use self attention to make refined context representation by using gated attention on context itself.Thus we obtain context representation from question (first part) and context itself(second part). This better representation help us predict answers accurately. Using this as motivation we implemented our R-NET model as illustraded in Figure 2.

4.3.2 Embedding Layer

The embedding layer includes both word and character level inputs and are generated similar to the one in R-net paper. Character level embedding are taken from the final hidden state of bi-directional RNN based on characters in the tokens. The concatenation of word embeddings and character embeddings used to generate new representation of context and question. We used Gated Recurrent Unit (GRU) as they are computationally efficient and similiar in result than LSTM.

Consider context with words level embeddings as $words^{Context}$ and character level embeddings as $chars^{Context}$ and question with word level embeddings as $words^{Question}$ and character level embeddings as $chars^{Question}$

$$x_t = BiRNN(x_{t-1}, [words_t^{Context}, chars_t^{Context}]) \tag{1}$$

$$q_t = BiRNN(q_{t-1}, [words_t^{Question}, chars_t^{Question}]) \tag{2}$$

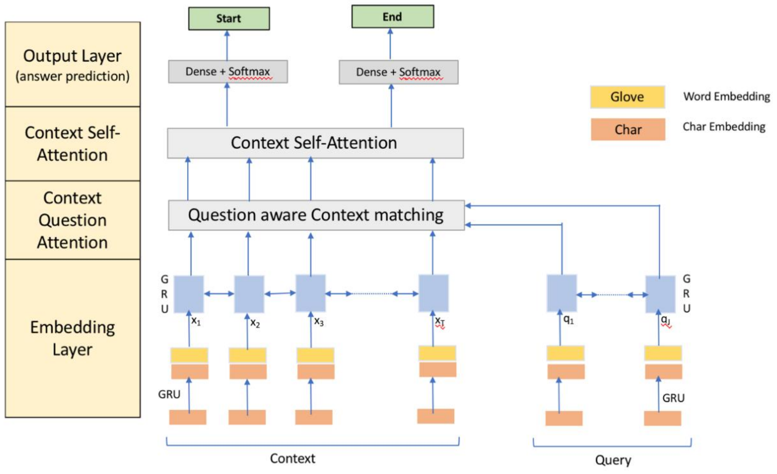


Figure 2: R-NET Architecture

4.3.3 Context to Question attention Layer

We apply basic dot-product attention with context hidden states c_i attending question hidden states q_j as in our baseline model, the attention score e_i was calculated as :

$$e^i = [x_i^T q_1 .....x_i^T q_M] \tag{3}$$

, where M is length of question. Attention distribution :

$$\alpha_i = softmax(e^i) \tag{4}$$

Attention Output :

$$c_t = \sum_{j=1}^{M} \alpha_j^t q_j \tag{5}$$

To determine importance of context parts relevant to the question, we add another gate to the input $[c_t, keys]$.

$$g_t = sigmoid(W_g[x_t, c_t]) \tag{6}$$

$$[x_t, c_t] = g(t) \odot [x_t, c_t] \tag{7}$$

$$v_t = RNN(v_{t-1}, [x_t, c_t]) \tag{8}$$

4.3.4 Self-attention Layer

The context to question representation generated in previous part layer lacks in full representation of context as it focuses only on specific parts relevant to question. To get full representation of context we directly match context to question representation to itself. In this layer we use additive attention to generate attention score as : Here v is context to question output we obtained in previous layer

$$e_j^i = V^T tanh(W_1 v_j + W_2 v_i) \tag{9}$$

Attention distribution :

$$\alpha^i = softmax(e^i) \tag{10}$$

Attention Output :

$$c_t = \sum_{j=1}^{N} \alpha_j^i v_j \tag{11}$$

, where M is length of question. We also applied additional gate to these representations same as in previous layer

$$g_t = sigmoid(W_g'[v_t, c_t]) \tag{12}$$

$$[v_t, c_t] = g(t) \odot [v_t, c_t] \tag{13}$$

$$h_t = BiRNN(h_{t-1}, [v_t, c_t]) \tag{14}$$

4.3.5 Output Layer

The attention outputs (self attention and context to question attention) are concatenated to obtain the blended reprentations b_i. Each of the blended representations b_i are fed through a fully connected layer followed by ReLU non-linearity.

$$b_i = ReLU(W_{FC}h_t + v_{FC}) \,\forall \in 1, ..., N \tag{15}$$

where W_{FC} and v_{FC} are weights matrix and bias vector. Each context location i is assigned a score(logits) which is calculated as :

$$logits_i^{start} = w_{start}^T b_i + u_{start} \tag{16}$$

We apply softmax to logits to obtain probability distribution p^{start} as :

$$p^{start} = softmax(logits^{start}) \in R^N \tag{17}$$

Similarly p^{end} is computed.

5 Experiments and Evaluation

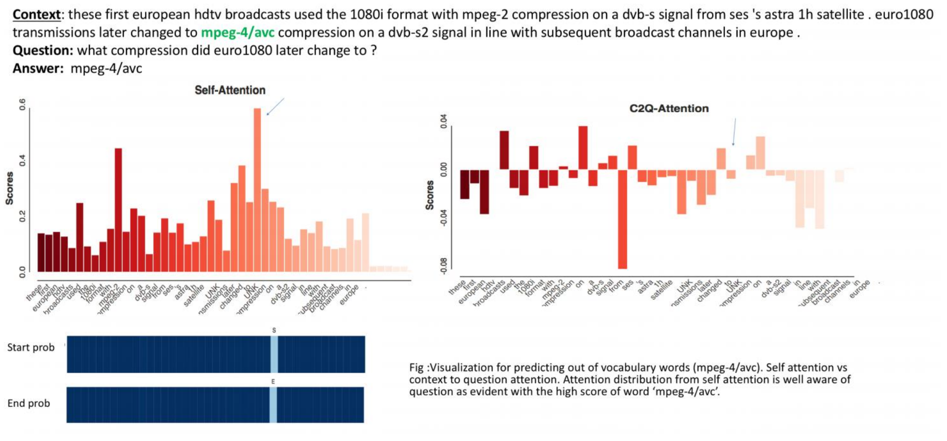
In this section, we present our experimental methodology, evaluation metrics and the results of the experiments(Table 1) . In all of the experiments, we used cross validation to pick the model with the best performance. Namely, we discuss how the data was split into train, dev and test sets, but the results are reported only on train and test sets.

5.1 Analysis

Error analysis: Our model predicts start and end probability independently. This causes issue like above where end probability index is less than start probability with errored output. Accuracy issues due to these issues can be improved by our model considering start probability distribution location into end probability distribution calculation.

Model	Dev F1/EM	Train F1/EM
BasicAttn(Baseline)	43.2/33.9	63.7/52.2
BiDAF (q2c attention on Baseline)	48.7/38.9	67.5/56.6
Coattention(with dynamic decoder)	57.2/48.2	69.3/54.6
R-NET(our model with self-attention on Baseline)	65.1/53.0	69.1/53.3
R-NET(our model with self-attention, Char emb, Gating)	71.3/60.6	76.2/61.6

Table 1: F1/EM Results for different model implementations


Context: these first european hdtv broadcasts used the 1080i format with mpeg-2 compression on a dvb-s signal from ses 's astra 1h satellite . euro1080 transmissions later changed to **mpeg-4/avc** compression on a dvb-s2 signal in line with subsequent broadcast channels in europe .

Question: what compression did euro1080 later change to ?

Answer: mpeg-4/avc

Fig :Visualization for predicting out of vocabulary words (mpeg-4/avc). Self attention vs context to question attention. Attention distribution from self attention is well aware of question as evident with the high score of word 'mpeg-4/avc'.

Figure 3: Attention Visualization

5.2 Hyperparameter Tuning

We tried experimenting various combinations of hyperparameters.
-We tested with different learning rate of 0.01, 0.002, 0.005 and 0.001. We noticed our model gave best result with 0.001
-We also tried including epsilon but did not see any impact on the model performance.
-We tried various combinations of dropout(zero dropout, 0.2 drop out variable dropout across layers). Of all combinations, we saw best accuracy with single dropout of 0.2 across all models.
- From input data analysis, we could see most of the context examples are of max lenght 350 or less. So we reduced the context lenght to 350 instead of 600. This improved the total time taken for each iterations by atleast 10% without any loss to accuracy.

5.3 Conditioning Start and End Predictions

We tried implementing output layer for Coattention and R-net using pointer networks but were not able to get signficant gain. We will continue our future work on implementing the pointer network.

5.4 Regularization

Droupout Regularization
When we ran the model with zero dropout and we found that model is overfitting after 4K iterations and loss started increasing for dev data. we noticed the gap of around 25% between train and dev F1 scores after 10k iterations. We then tried with single dropout (0.2) for complete model and also variable dropouts [1] at different layers (hidden and self attention) of the model. When we gave 20% dropout for embedding layer and increased dropout for hidden states in later layers (with 30% in question to context matching and 40% for self-attention layer) we got the best results. Drop out regularization helps to generalize the results

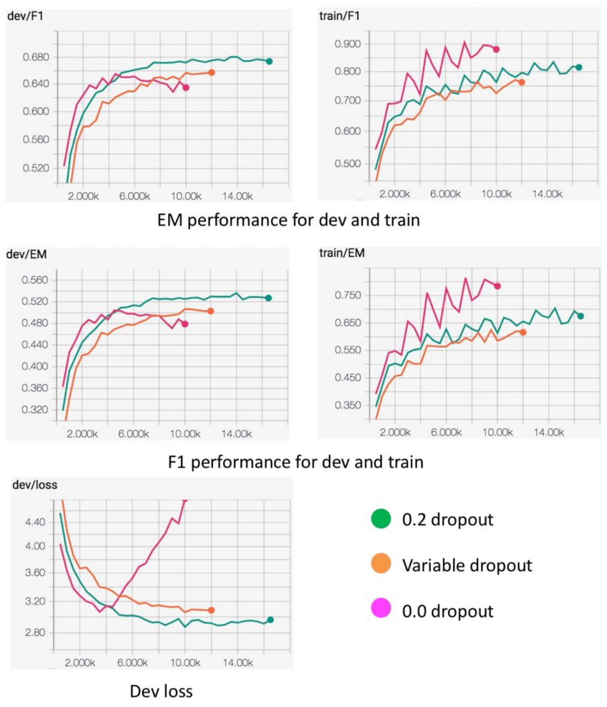


Figure 4: EM, F1 and dev loss

Experiment	Dev F1/EM	Train F1/EM
R-NET(no dropout at 10k iteration)	63.5/47.8	88.0/78.4
R-NET(0.2 dropout at 10k iteration)	62.6/46.3	76.9/62.0
R-NET(variable dropout; 0.2 for embed , 0.4 for hidden states , 0.5 in self-attention)	65.7/50.7	72.5/58.5
R-NET(variable dropout; 0.2 for embed , 0.3 for hidden states, 0.4 in self-attention)	66.2/51.1	75.7/60.6

Table 2: F1/EM for different Dropouts on R-NET (our model with self-attention, Char emb, Gating)

and prevents overfitting. It is important for model to have maximum information of inputs while dropping more of hidden states helps to generalize the model. Results (Table 2) also supports this with model giving better results when variable dropouts given across different layers, keep lower dropout at embedding layer and increasing in later layers.

6 Conclusion

For this assignment, we tried implementing three different models (BiDAF, Coattention, R-NET) with minor modifcations to the approach defined in the paper. We saw self attention improved the acuracy of the model significantly. The visualization of attention demonstrates and our final model (R-NET) with char embeddings gave F1 / EM score of 71.3%/ 60.6% on Test set. Our possible future work is to implement the same model with char CNN and also extend to include pointer networks.

References

[1] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[2] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016.

[3] M. J. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2016.

[4] S. Wang and J. Jiang. Machine comprehension using match-lstm and answer pointer. *CoRR*, abs/1608.07905, 2016.

[5] C. Xiong, V. Zhong, and R. Socher. Dynamic coattention networks for question answering. *CoRR*, abs/1611.01604, 2016.