Machine Reading Comprehension on SQUAD V2

Marius Seritan
LinkedIn
Mountain View, CA 94043
mseritan@stanford.edu

Yang Chen
LinkedIn
Mountain View, CA 94043
yangc27@stanford.edu

Abstract

We will build a question answering model in this project and evaluate on SQuAD 2.0. In particular, we are interested in how pre-trained context embedding and attention mechanism affect the performance of the question answering models. We found that using Pre-training of Deep Bidirectional Transformers (BERT), the performance will be limited by hardware resources. In addition, we also explored using character embedding to better handle unknown words and carefully designing attention mechanisms that incorporate self-attention. The latter two experiments are still in progress.

1 Approach

1.1 Baseline model

Our baseline model is the BiDAF model [1] with the following five layers from the bottom to the top:

1. Embedding layer, which contains an fixed Glove embedding look up table and a two-layer highway network
2. Encoder layer, which consists of a bidirectional LSTM
3. Bidirectional attention flow layer
4. Modeling layer, which is a bidirectional LSTM to refine the question-aware context representation
5. Output layer, which applies softmax on the transformed the concatenated output of the attention layer and the modeling layer

Specifically in the attention layer, we first calculate a similarity matrix S:

$$S_{ij} = w_{sim}^T[c_i; q_j; c_i \circ q_j], \tag{1}$$

where c_i is the context hidden states from the encoder layer and q_j is the question hidden states. We then get Context-to-question (C2Q) and Question-to-context (Q2C) attention from S: the C2Q output a_i is calculated as the weighted sum of question hidden states

$$\bar{S}_{i,:} = softmax(S_{i,:}) \tag{2}$$

$$a_i = \sum_j \bar{S}_{i,j} q_j \tag{3}$$

The Q2C output b_i is calculated as the weighted sum of context hidden states:

$$\bar{\bar{S}}_{:,j} = softmax(S_{:,j}) \tag{4}$$

$$S' = \bar{S}\bar{\bar{S}}^T \tag{5}$$

$$b_i = \sum_j S'_{i,j} c_j \tag{6}$$

1.2 Character embedding

We first explored using character embedding [2] in addition to word embedding to better handle out-of-vocabulary words. The character embedding layer contains embedding lookup, a convolutional network, and highway network with dropout. In the convolutional network, we use 1-dimensional convolutions, and learn a weight matrix W and a bias vector b. Assume x is the reshaped character embedding lookup result, the output of the convolutional network is calculated as

$$x_{conv} = Conv1D(x) \tag{7}$$

$$x_{conv_out} = MaxPool(ReLU(x_{conv}) \tag{8}$$

The character level embedding for out-of-vocabulary words is then combined with word embedding in the previously described embedding layer.

1.3 Self attention

We are exploring an attention mechanism [3] that leverages self-attention. Specifically, assume that the original context/passage representation and question representation is P and Q, respectively. The aligned representation for context/passage is $\tilde{P} = \{b_j\}, j = 1, ..., N$ and for question is $\tilde{Q} = \{a_i\}, i = 1, ..., N$. We first fuse the original and aligned representation as follows:

$$P' = g(P, \tilde{Q}) \cdot m(P, \tilde{Q}) + (1 - g(P, \tilde{Q})) \cdot P \tag{9}$$

where g is a gate function and m is defined as

$$m(P, \tilde{Q}) = tanh(W_f[P; \tilde{Q}; P \circ \tilde{Q}; P - \tilde{Q}] + b_r) \tag{10}$$

Then we calculate the self-attention part, which further refine the obtained information from the context-question-attention layer. In this layer, we will explore adding manual features to the question-aware passage representation:

$$D = BiLSTM([P'; feat_{man}]) \tag{11}$$

and then calculate bilinear self-attention:

$$L = softmax(D \cdot W_1 D^T) \tag{12}$$

Therefore the self-aligned representation is

$$\tilde{D} = L \cdot D \tag{13}$$

Finally we apply the fusion function again to get

$$D' = g(D, \tilde{D}) \cdot m(D, \tilde{D}) + (1 - g(D, \tilde{D})) \cdot D \tag{14}$$

2 Experiments

2.1 Data

We train and tune models on the provided SQuAD 2.0 dataset. We follow the default setting on splitting train, dev and test set.

2.2 Evaluation method

We use the EM and F1 metrics as defined by SQuAD to evaluate models. We will compare our results with the public leaderboard SQuAD and the class leaderboards.

2.3 Experimental details and results

2.3.1 Unknown words ratio

In order to judge the usefulness of integrating character encoding we evaluated the number of unknown words. For this we analyzed the word indexes generated with GloVe and looking for the index of the unknown tag. We obtained the following

Set	OOV Words	Total Words	Ratio
Train	136,228	51,840,172	0.26%
Dev	22,230	235,8170	0.94%

Ratio of Out Of Vocabulary words

2.3.2 Data loss during pre-processing

In the pre-processing logic input records are dropped under some conditions. We computed the following statistics in order to evaluate the impact of this data loss:

Set	Processed examples	Total examples	Loss %
Train	129941	130319	0.29%
Dev	5951	6078	2.1%

Example loss during pre-processing

The code for the above 2 sections is here:

https://github.com/winding-lines/cs224n-squad/blob/master/input_stats.py

Note: this is private repo as per the instructions, let us know which TA credentials to enable.

2.3.3 BERT feasibility

BERT and related PCE techniques are exciting development in the last year or so. In order to evaluate the feasibility of using in our project we used the off-the-shelf code to pre-train BERT. We get out-of-memory errors in the CUDA layer for most combinations of parameters. See table below for more details

max_seq_length	batch_size	training status
384	32	OOM
256	14	OOM
128	32	OOM
64	64	success

BERT memory usage

Given the difficulty of fine-tuning BERT it may be more beneficial for our learning process to focus and master other techniques first.

2.3.4 Baseline pre-train

We have pre-trained a baseline of the BiDAF code as provided by the TAs (many thanks!) and we have obtained the following scores

EM	58.125
$F1$	61.526

The link to the submission is here:
https://www.gradescope.com/courses/34514/assignments/161228/submissions/14983685

2.3.5 Dropout hyper parameter tuning

We have run some experiments with different values for the dropout probability parameter. See below for the different values of the dropout we have tried.

Dropout probability	EM	F1
0.0	58.03	61.4
0.2	57.44	60.74
0.5	52.13	55.53

These results are a bit surprising, using dropout probability of 0.0 gives the best result for this network architecture. It may be interesting to explore this further.

3 Future work

We will finish experimenting with character embedding based model and the self-attention approach. Additionally we plan to improve the pre-processing pipeline and see if we can reduce the loss of examp;es.

We will consider applying pre-trained context embedding as a stretch goal.

References

[1] Seo, M., Kembhavi, A., Farhadi, A., Hajishirzi, H. (2016). Bidirectional attention flow for machine comprehension. arXiv preprint arXiv:1611.01603.

[2] Kim, Y., Jernite, Y., Sontag, D., Rush, A. M. (2016, March). Character-aware neural language models. In Thirtieth AAAI Conference on Artificial Intelligence.

[3] Wang, W., Yan, M., Wu, C. (2018). Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. arXiv preprint arXiv:1811.11934.