Question Answering Using Bi-Directional Attention Flow with Position Encoder

Denis Stepanov*
JetBrains, Co. Ltd.
St-Petersburg, Russia
denis.stepanov@jetbrains.com

Abstract

Automated reading comprehension with Stanford Question Answering Dataset is a challenge bringing new breakthroughs in Natural Language Processing every month. The purpose of this project was to try different attention algorithms in the question answering model and possibly stack them. The final model achieved 67.26/55.94 F1/EM score on the test dataset.

1 Introduction

The question answering became very popular field in the Natural language processing especially after the release of the Stanford Question Answering Dataset (SQuAD) [5].

In this project the author aimed to use different types of attention algorithms and possibly stack them to achieve a better performance. The algorithms of choosing the start and end position are playing an important role in model accuracy too, so some efforts were dedicated to the selection of algorithm for start-end position prediction.

Considerable time was spent on the overfitting problem, which stayed the prevalent issues during the whole project.

Observed problems and possible solution are discussed after the model analysis.

2 Dataset

The SQuAD is a text comprehension dataset, which consists of human posed questions for a set of Wikipedia articles. The answers for these questions are the segments of reading passages for which the questions were posed. The dataset has more than 100 000 question-answer pairs on more than 500 articles.

For each part of some Wikipedia article, called context, there were written around 5 questions by humans. The dataset is split into train, development and test sets. The train dataset was used to train the model, the development set was accessible for model validation and the test set was used for final validation of the model.

3 Model

We were provided with the Baseline Model code and description [6]. Describing the model we will follow the notations used in the handout.

*Use footnote for providing further information about author (webpage, alternative address)—*not* for acknowledging funding agencies.

3.1 Encoding Layer

Each context passage is represented by a sequence $x_1, \ldots, x_N$ of pretrained GloVe embeddings $x_i \in \mathbb{R}^h$, where h was taken equal to 100 (out of choice 50/100/200/300). Smaller values of h significantly slowed down the model training, the larger values are not acceptable due to limitations on the resulting model size. The same embedding size was used for the question sequence $y_1, \ldots, y_M \in \mathbb{R}^h$.

Like in Baseline model, the context and question embeddings are the inputs for the 1-layer biGRU:

$$\{\overrightarrow{c_1}, \overleftarrow{c_1}, \ldots, \overrightarrow{c_N}, \overleftarrow{c_N}\} = \text{biGRU}(\{x_1, \ldots, x_N\})$$
$$\{\overrightarrow{q_1}, \overleftarrow{q_1}, \ldots, \overrightarrow{q_M}, \overleftarrow{q_M}\} = \text{biGRU}(\{y_1, \ldots, y_M\})$$

The author also tried to use biLSTM cells, but the history of observations advocates that it doesn't worth to change this part of Baseline model: the increased model complexity does not provide increase in model accuracy and slowers down the training process. The obtained forward $\overrightarrow{c_i}, \overrightarrow{q_j} \in \mathbb{R}^h$ and backward $\overleftarrow{c_i}, \overleftarrow{q_j} \in \mathbb{R}^h$ hidden states are stacked in pairs to produce context and question hidden states:

$$c_i = [\overrightarrow{c_i}; \overleftarrow{c_i}] \in \mathbb{R}^{2h}, i \in \{1, \ldots, N\}; \quad q_j = [\overrightarrow{q_j}; \overleftarrow{q_j}] \in \mathbb{R}^{2h}, i \in \{1, \ldots, M\}$$

3.2 Attention Layer

The attention layer was implemented in accordance to BiDAF model [4]. Having context and question hidden states $c_i, q_j \in \mathbb{R}^{2h}$ we calculate the similarity matrix $S \in \mathbb{R}^{N \times M}$ with elements:

$$S_{ij} = w_{sim}^T [c_i; q_j; c_i \circ q_j] \in \mathbb{R}$$

To avoid the problem with allocating a huge tensors for the concatenated hidden states the similarity matrix computation was split into three parts

$$w_{1,sim}^T c_i; \quad w_{2,sim}^T q_j; \quad w_{3,sim}^T [c_i \circ q_j],$$

where $w_{sim} \in \mathbb{R}^{6h}$ and $w_{k,sim} \in \mathbb{R}^{2h}$, $k = 1, 2, 3$. Each separate computation can be seen as the fully connected (FC) layer with identity activation and number of outputs equal to 1.

Using this analogy one can make use of different options of FC layers. On different stages of the research, the author tried to use different activation functions (to get more flexible similarity measure) and batch normalization (to prevent the growth of weights), but finally none of these changes get into the final version of the model.

The Context-to-Question Attention was implemented in accordance to the BiDAF model description:

$$\alpha^i = \text{softmax}(S_{i,:}) \in \mathbb{R}^M, \; a_i = \sum_{j=1}^{M} \alpha_j^i q_j \in \mathbb{R}^{2h}, i \in \{1, \ldots, N\}$$

Next the Question-to-Context Attention was implemented:

$$m_i = \max_i S_{ij} \in \mathbb{R}, \; \beta = \text{softmax}(m) \in \mathbb{R}^N, \; c' = \sum_{i=1}^{N} \beta_i c_i \in \mathbb{R}^{2h}, \; i \in \{1, \ldots, N\}$$

The attention layer output was defined as

$$b_i = [c_i; a_i; c_i \circ a_i; c_i \circ c'] \in \mathbb{R}^{8h}, \; i \in \{1, \ldots, N\}$$

For the purpose of decreasing the size of the model the alternative variant of the output was also used:

$$b_i = \text{maxpool}(c_i; a_i; c_i \circ a_i; c_i \circ c') \in \mathbb{R}^{2h}, \; i \in \{1, \ldots, N\}$$

3.3 Modeling Layer

Following the BiDAF model [4] description, the Modeling Layer was added also. To decrease the model size and to speedup the training process the author tried both 2-layer biLSTM and biGRU.

Actually no difference in the final model accuracy was observed, so a more light-weight biGRU cell was used to construct this layer:

$$\{\overrightarrow{v_1}, \overleftarrow{v_1}, \ldots, \overrightarrow{v_N}, \overleftarrow{v_N}\} = \mathrm{biGRU}_2(\{b_1, \ldots, b_N\})$$

where biGRU_2 denotes the 2-layer biGRU cell. The author tried both biGRU_2 and biGRU cells in this layer and found no actual difference in the overall model accuracies in both cases. The output of the modeling layer is the concatenated forward and backward states vectors: $v_i = [\overrightarrow{v_i}; \overleftarrow{v_i}]$, where the dimension of v_i depends on the dimension of the blended output of the attention layer.

3.4 Self Attention Layer

As was shown in [2] the self-attention is a very powerful instrument in question answering models. It takes some sequence of question-aware representations $v_1, \ldots, v_N \mathbb{R}^\ell$ and makes an attention layer, where each each representation attends to all another representations:

$$e^i_j = \mathrm{score}(v_i, v_j) \in \mathbb{R};\ \alpha^i = \mathrm{softmax}(e^i) \in \mathbb{R}^N;\ a^i = \sum_{j=1}^{N} \alpha^i_j v_j \in \mathbb{R}^\ell$$

In original article [2] the Bahdanau additive style score was offered. In order to decrease the size of the model, the author tried dot product and Luong scores [7]. In order to decrease the model size the context length was set to 300 and batch size was decreased to 50 and the attention layer output was taken in the max pooling variant. These changes made model extremely slow to train and led to performance degradation. This layer was excluded from the final model variant.

3.5 Output Layer with Position Encoder

The modeling layer outputs v_i are fed to the output layer, which has to predict the start and end positions of the answer inside the context. After some experimentation the author choose the next model (similar to one used in [4]), which gave the performance boosting due to more accurate answer span prediction. The probability distribution of the start index is defined as in [6]:

$$p^{start} = \mathrm{softmax}\left(w^T_{start}\mathrm{ReLU}\left(W_{FC}v_i + b_{FC}\right) + b_{start}\right)$$

Then we pass $v_1, \ldots, v_N$ to biGRU encoder and obtain $v'_1, \ldots, v'_N$, which then is used to compute the probability of the end index:

$$p^{end} = \mathrm{softmax}\left(w^T_{end}\mathrm{ReLU}\left(W_{FC}v'_i + b_{FC}\right) + b_{end}\right)$$

3.6 Loss Function and Optimizer

Loss function was defined as the sum of the cross-entropies for start and end positions. To control the span of the predicted answer, author also tried to introduce an additional term with the square loss computed for the difference of the actual span and the expected value of the predicted span, computed according to p^{start} and p^{end} distributions. Unfortunately, the author didn't manage to tune parameters under which such model could train with the reasonable speed.

The author tried different optimizers: Adam, AdaDelta, AdaMax [3] with different learning rate exponential decays, but found no considerable improvement compared to Baseline Adam optimizer with learning rate equal to 0.001. The implementation of the AdaMax optimizer was borrowed from the open.ai library [1].

3.7 Prediction

At the evaluation time for a given context and question in the Baseline model the argmax over p^{start} and p^{end} distributions is obtained and the predicted span $(\ell^{start}, \ell^{end})$ is returned.

There are two problems with this approach:

- Distributions p^{start} and p^{end} are supposed to be independent, but they are not (actually we introduced the position encoder in order to lessen the independence assumption);

- We can obtain $\ell^{start} > \ell^{end}$ and predictor will return the empty string as an answer.

To mitigate the aforementioned problems the distribution of the true answers spans was constructed for the examples from the train set. As was advised in the [6], the prediction was changed to return the span within the range 15 with the highest probability. This solves the problem of empty and long predicted answer spans.

3.8 Regularization

The main problem with model training was that model tends to overfit after approximately 5000 steps. To solve the problem author tried to tune dropout rate (and implement dropout on different layers of the model). High dropout rates (over 0.5) hurt the model performance and give just a moderate effect on overfitting.

The author also implemented the L_1 and L_2 regularization for the model. Even low L_1 regularization makes model hardly trainable (like in case of high dropout rates). Small L_2 regularization (around 0.0001) improves accuracy on the development set. Higher L_2 regularization coefficients also decrease model accuracy on train and development sets.

The size of the training batch can also be used as a remedy against overfitting. It was checked, that large batch sizes make model train faster and overfit quickly. The small batch size make the training much slower and the performance decreases slightly.

The author also tried different sizes of embedding vectors, but smaller embedding size lead to performance degradation on the train set and large embedding size causes OOM problems for models with attention.

4 Experiments and Discussion

Starting from the Baseline model the author first examined the influence of the different types of regularization (dropout, L_1, L_2) to mitigate the problem of overfitting observed on the Baseline model. The parameter search led to the choice of small L_2 regularization.

Then the input data analysis showed, that the input vectors lengths can be decreased due to the fact that the most part of the distribution of the context length lays below the value 300. This helped to implement bigger models and avoid OOM issues.

The first thing to implement in the model were different types of attention, because attention happens to be very powerful tool in question answering models. The most part of experimentation was carried over bi-attention[4] and self-attention [2].

The first implemented attention was the bi-attention [4]. It gave the performance boost over the baseline. Implementation of the self-attention (with Luong score) led to the necessity of decreasing the model parameters, defining context length, embedding size and batch size. The overall impact of self-attention on the model performance was lower than in case of bi-attention, so the basic attention mechanism was chosen to be bi-attention.

The author also tried to stack bi-attention with self attention, but this led to necessity of decreasing model parameters (embedding size, batch size) and problems with model training.

The analysis of the predicted answers spans led to the idea, that the important role in every model plays the method of choosing start and end tokens of the answer. The default approach with independent argmax calculation of span boundaries from two distributions leads to several problems, including empty predicted answers (when start position goes after the end position).

To mitigate this problem two things were done: Position encoder implemented and the choice of the start and end positions changed.

The Position encoder is analogous to the combination of modeling and output layers of the BiDAF model [4]. The main difference is that in the modeling layer the 1-layer biRNN is used instead of 2-layer biLSTM. This change of the model deceased the number of model parameters and made the training process faster. This simplification didn't led to the model accuracy degradation compared to

the full implementation of the modeling and output layers, described in [4], so the simplified model was used in the final version of the model.

The choice of the predicted start and end position was performed by choosing the span with length less than 15 and the highest probability of combination of start and end positions.

The best model was run on the development and test sets and submitted to the leaderboards.

4.1 Results

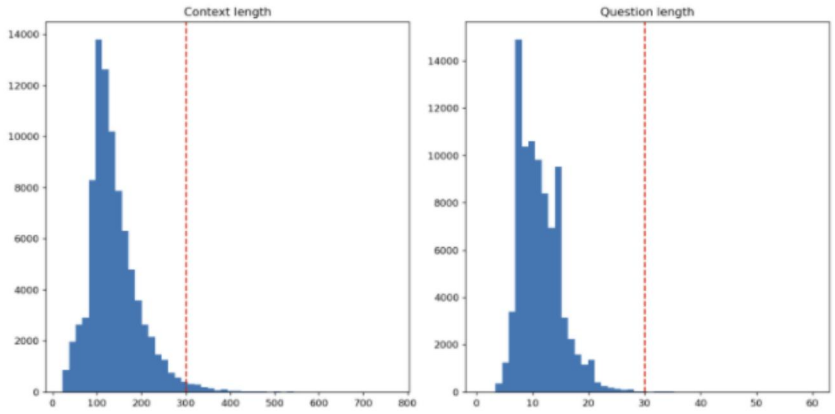
The best model submitted to the leaderboard was the combination of bi-attention [4] with modeling layer from [4] replaced with the Position encoder. The prediction was improved with the help of restriction on the span length (the predicted span under 15 length with the highest span probability was taken). The results for the best model are presented in Table 1.

Model	F1	EM
BaseLine Dev	39.94	33.95
Best Model Dev	65.88	54.57
Best Model Test	67.26	55.94

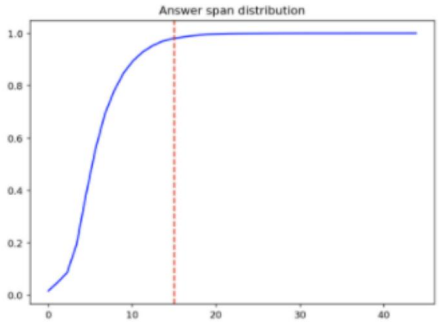
Table 1: Results of experiments

4.2 Data Preprocessing

In order to be able to painlessly decrease the context length and get the faster model training, the distributions of the context and question lengths were constructed. The default context value 600 can be decreased to the value 300 without serious damage to the model performance. The question length can be also decreased to the value 20 against default value 30, but this plays no role on the model training and was set to default. When needed, the context length was set to 300. This limit was used in the final model also.



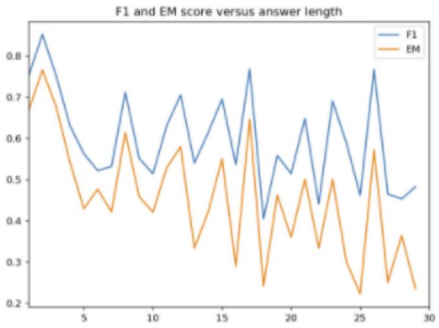
As was noticed in [6], one can also decrease the length of the answer span in the context to solve the problem of very large predicted answer spans. In order to check, that the 15 is the valid value for this limitation, the distribution of answers lengths was calculated, which shows, that we can actually take 15 as the maximal answer span without damage for the model performance.



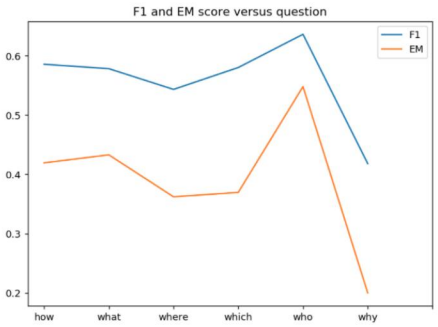
4.3 Result Analysis

As far as the obtained model accuracy is far from excellent, we can search possible reasons for this result. First of all, consider the F1 and EM scores versus the length of the answer. As was expected, the length of the answer plays crucial role in the EM accuracy. The F1 score also is lower for lengthy answers. This observation shows that the part of the model responsible for the selection of start and end position plays very important role in the model quality.

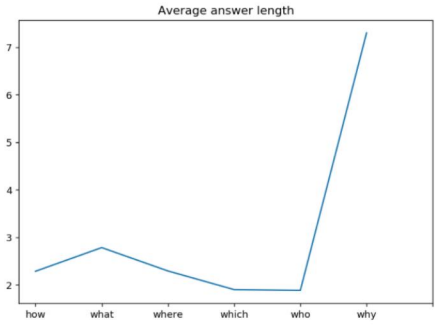
The obvious conclusion is that to achieve better performance author had to implement some more powerful algorithm of the start-end position choice.



We can also consider the main questions: ”how”, ”what”, ”where”, ”when”, ”who”, ”why” and observe, that the question ”why” gets the lowest score among all other.



The partial explanation for this observation is that questions ”why?” lead to more lengthy answers, which, as we already know tend to have lower scores. Short questions ”who” has the highest score, as we can see from the plot.



Let's consider a couple of textual examples of the model work.

Example 1. CONTEXT: abc also owns the times square studios at 1500 broadway on land in times square owned by a development fund for the 42nd street project ; opened in 1999 , good morning america and nightline are broadcast from this particular facility . abc news has premises a little further on west 66th street , in a six-story building occupying a 196 feet (60 m) 379 feet (116 m) plot at 121135 west end avenue . the block of west end avenue housing the abc news building was renamed peter jennings way in 2006 in honor of the recently deceased longtime abc news chief anchor and anchor of world news tonight .

QUESTION: a block of west end avenue that houses an abc news building was renamed for what abc anchor ?

TRUE ANSWER: peter jennings

PREDICTED ANSWER: peter jennings way

F1 SCORE ANSWER: 0.800

EM SCORE: False

Here we can see that the word "way" was added to the right answer, which is the name of the person. In this case additional token features like named entities can help to increase the EM score.

Example 2. CONTEXT: to remedy the causes of the fire , changes were made in the block ii spacecraft and operational procedures , the most important of which were use of a nitrogen/oxygen mixture instead of pure oxygen before and during launch , and removal of flammable cabin and space suit materials . the block ii design already called for replacement of the block i plug-type hatch cover with a quick-release , outward opening door . nasa discontinued the manned block i program , using the block i spacecraft only for unmanned saturn v flights . crew members would also exclusively wear modified , fire-resistant block ii space suits , and would be designated by the block ii titles , regardless of whether a lm was present on the flight or not .

QUESTION: what type of materials inside the cabin were removed to help prevent more fire hazards in the future ?

TRUE ANSWER: flammable cabin and space suit materials

PREDICTED ANSWER: space suit materials

F1 SCORE ANSWER: 0.667

EM SCORE: False

This example shows why the lengthy answers have usually lower scores. Actually in this case looks like the predicted answers was closer to the right answer, then the true one (provided by a human).

5 Conclusions and Future Work

The final shows much better results then the Baseline model due to a more effective attention algorithm and corrections in the start and end positions.

The analysis of the model results show, that the directions for the model improvements are related to the algorithms of choosing start and end positions and introducing token features.

The main problem while training the model was the overfitting. Having enough time to investigate the problem of overfitting could provide a much better solution of the question answering task by the model implemented in this work.

Acknowledgments

The author would like to thank the CS224N course staff for the lectures, homework tasks and the challenging Default final project.

For experiments the author had a machine with 1080 Ti GPU generously provided for use in this project by his employer JetBrains, Co. Ltd.

References

[1] Open AI. Adamax optimizer. URL `https://github.com/openai/iaf/blob/master/tf_utils/adamax.py`.

[2] Microsoft Research Asia. R-net: Machine reading comprehension with self-matching networks. URL `https://www.microsoft.com/en-us/research/wp-content/uploads/2017/05/r-net.pdf`.

[3] J. Ba D. P. Kingma. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL `http://arxiv.org/abs/1412.6980`.

[4] A. Farhadi M. Seo, A. Kembhavi and H. Hajishirzi. Bidirectional attention flow for machine comprehension. URL `arXiv:1611.01603`.

[5] K. Lopyrev P. Rajpurkar, J. Zhang and P. Liang. Squad: 100,000+ ques- tions for machine comprehension of text. URL `arXiv:1606.05250`.

[6] A. See et al. R. Socher. Cs224n: Default final project. URL `http://web.stanford.edu/class/cs224n/default_project/index.html`.

[7] TensorFlow Tutorials. Neural machine translation (seq2seq) tutorial. URL `https://www.tensorflow.org/tutorials/seq2seq`.