Answering Question with Bidirectional Attention Flow

Alan Ansell

June 25, 2018

Abstract

Reading Comprehension (RC) is a difficult task in NLP. Since its release in 2016, the Stanford Question Answering Dataset (SQuAD) has been the focus of a lot of RC research and has seen rapid improvement. Here we reimplement a method called Bidirectional Attention Flow (BiDAF) and achieve an F1 score of 76.6% on the development set and an exact match score of 66.0%. An original modification to BiDAF boosted the performance of our model by 0.7%.

1 Introduction

The SQuAD dataset [5] consists of more than 100,000 reading comprehension problems. Each example consists of a "context," which is a paragraph extracted from an English Wikipedia article, a "question" written by crowdsourced workers, and an answer, which must be a contiguous substring of the context. The constraint that the answer must be a substring of the context allows the use of simple performance measures, namely the F1 score and Exact Match (EM) score, the percentage of predictions which perfectly match the correct answer. Since there are often several reasonable spans which answer the question, each question has three (not necessarily different) anwers provided by independent human judges.

2 Model

Bidirectional Attention Flow [6] is a method which achieved state of the art performance on SQuAD in late 2016. "Attention" mechanisms in NLP are used to determine which words in a text are relevant for a certain task. In Bidirectional Attention Flow, attention techniques are used to determine which words in the question are relevant to each word in the context, and which words in the context are relevant to the question overall. We will now explain in detail each component of our implementation of the Bidirectional Attention Flow model.

2.1 Word Embedding Layer

The input to the model is a word vector for each token in the context and question. We use two methods to derive this word vector. The first is to use pretrained GloVe vectors [4]. These are available for 400,000 of the most frequent tokens in English. The second method is an encoding yielded by a character-level convolutional neural network (CNN) [3], which we will describe now.

We assign each unique character in the dataset an integer label from 1 to N, the number of characters. Let $\boldsymbol{e}^{(1)}, \boldsymbol{e}^{(2)}, ..., \boldsymbol{e}^{(N)} \in \mathbb{R}^{d_c}$ be trainable embeddings for each character. For some word w consisting of characters $c_1, c_2, ..., c_L$, first apply a 1D convolution over the embeddings $\boldsymbol{e}^{(c_1)}, \boldsymbol{e}^{(c_2)}, ..., \boldsymbol{e}^{(c_L)}$, yielding vectors $\boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}, ..., \boldsymbol{h}^{(L)} \in \mathbb{R}^f$, where f is the number of filters used in the convolution. We then obtain the character-level encoding $\boldsymbol{v}^{char}(w)$ for w by max-pooling over the character dimension: $v_i^{char}(w) = \max_{j=1,...,L} h_i^{(j)}(w)$.

In the original Bidirectional Attention Flow paper, the GloVe vectors $\boldsymbol{v}^{glove}(w)$ and character-level embeddings $\boldsymbol{v}^{char}(w)$ are fed into a highway network, but for simplicity we just concatenate them to form the final word embeddings $\boldsymbol{v}(w) = [\boldsymbol{v}^{glove}(w);\ \boldsymbol{v}^{char}(w)]$. Let us call the final embeddings for the context and query $\boldsymbol{c}_1, ..., \boldsymbol{c}_{L_c}$ and $\boldsymbol{q}_1, ..., \boldsymbol{q}_{L_q} \in \mathbb{R}^{d_g+f}$ respectively, where L_c and L_q are the number of tokens in the context and query, and d_g is the dimensionality of the GloVe vectors.

2.2 Contextual Embedding Layer

In this layer we try to produce an embedding which captures the significance of each word in the context of the surrounding words. We do this by feeding the outputs from the word embedding layer into a Bidirectional Recurrent Neural Network using Gated Recurrent Unit cells (BiGRU) [1]. There are two BiGRUs, one for the context and one for the query, but we share weights between them so that the outputs have similar structure (this is important in the attention flow layer). The original paper uses slightly more complex Long Short-Term Memory (LSTM) units [2] in place of GRUs, but in our experiments this did not make a significant difference.

The BiGRUs output hidden representations in both the forwards and backwards directions for each token in the context and question. We call these $(\overrightarrow{\boldsymbol{c}_1}, \overleftarrow{\boldsymbol{c}_1}), ..., (\overrightarrow{\boldsymbol{c}_{L_c}}, \overleftarrow{\boldsymbol{c}_{L_c}})$ and $(\overrightarrow{\boldsymbol{q}_1}, \overleftarrow{\boldsymbol{q}_1}), ..., (\overrightarrow{\boldsymbol{q}_{L_q}}, \overleftarrow{\boldsymbol{q}_{L_q}})$. The final output of the contextual embedding layer is the concatention of the hidden representations in the two directions, $[\overrightarrow{\boldsymbol{c}_1};\ \overleftarrow{\boldsymbol{c}_1}], ..., [\overrightarrow{\boldsymbol{c}_{L_c}};\ \overleftarrow{\boldsymbol{c}_{L_c}}]$ and $[\overrightarrow{\boldsymbol{q}_1};\ \overleftarrow{\boldsymbol{q}_1}], ..., [\overrightarrow{\boldsymbol{q}_{L_q}};\ \overleftarrow{\boldsymbol{q}_{L_q}}] \in \mathbb{R}^{2h}$, where h is the dimensionality of the hidden layer of the RNN.

In the original BiDAF paper, the GloVe vectors are held constant. However it seems that it might be useful to allow some of these vectors to be changed, especially for frequent "function" words, whose meanings are not necessarily well captured by co-occurence based techniques such as GloVe. Therefore we make the embeddings for the V most frequently occurring tokens variable, allowing the optimiser to backpropagate into them.

2.3 Attention Flow Layer

In this layer we attempt to determine the relationship between words in the context and the question. The involves first computing a "similarity score" S_{ij} between each pair of words in the context and question. From now on we will refer to the contextual embeddings of the context and question as $\tilde{\boldsymbol{c}}_1, ..., \tilde{\boldsymbol{c}}_{L_c}$ and $\tilde{\boldsymbol{q}}_1, ..., \tilde{\boldsymbol{q}}_{L_q}$ respectively. S is calculated as

$$S_{ij} = \boldsymbol{w}^\top [\tilde{\boldsymbol{c}}_i; \tilde{\boldsymbol{q}}_j; \tilde{\boldsymbol{c}}_i \circ \tilde{\boldsymbol{q}}_j] \quad \forall i = 1, ..., L_c \ \forall j = 1, ..., L_q, \tag{1}$$

where $\circ$ denotes element-wise product and $\boldsymbol{w} \in \mathbb{R}^{6h}$ is a trainable weight vector. This method of calculating the similarity matrix relies on the intuition from word vectors that embeddings with a larger dot product tend to be more similar in meaning.

2.3.1 Context-to-query Attention

The similarity matrix is used to calculate an attention distribution over the question words for each context word. This distribution is then used as coefficients for a linear combination of the question words to produce an "attended question vector" $\boldsymbol{u}_i$:

$$\boldsymbol{a}_i = \mathrm{softmax}(S_{i,:}) \in \mathbb{R}^{L_q} \qquad \forall i = 1, ..., L_c \tag{2}$$

$$\boldsymbol{u}_i = \sum_{j=1}^{L_q} a_{ij} \tilde{\boldsymbol{q}}_j \in \mathbb{R}^{2h} \qquad \forall i = 1, ..., L_c. \tag{3}$$

2.3.2 Query-to-Context Attention

Query-to-Context attention allows us to determine which context words are relevant to the question overall. We define a context word's relevance to the question to be its maximum similarity to any question word. We use these relevances to form a distribution and linear combination over the context words.

$$r_i = \max_{j=1,...,L_q} S_{ji} \quad \forall i = 1, ..., L_c \tag{4}$$

$$\boldsymbol{b} = \text{softmax}(\boldsymbol{r}) \tag{5}$$

$$\boldsymbol{c}' = \sum_{j=1}^{L_c} b_j \tilde{\boldsymbol{c}}_j \tag{6}$$

The final output from the Attention Flow layer for each token i in the context is the concatenation $\boldsymbol{g}_i = [\tilde{\boldsymbol{c}}_i; \boldsymbol{u}_i; \tilde{\boldsymbol{c}}_i \circ \boldsymbol{u}_i; \tilde{\boldsymbol{c}}_i \circ \boldsymbol{c}'] \in \mathbb{R}^{8h}$.

2.4 Modeling Layer

The modeling layer allows the query-aware representations of the context words to interact. We can think of this being where reasoning about the context in relation to the question takes place. We use two layers of BiGRU with hidden size h with inputs $\boldsymbol{g}_1, ..., \boldsymbol{g}_{L_c}$. We again concatenate the outputs from the forward and backward directions, yielding outputs $\boldsymbol{m}_1, ..., \boldsymbol{m}_{L_c} \in \mathbb{R}^{2h}$.

2.5 Output Layer

Recall that answers in SQuAD always consist of a contiguous span of the context tokens. Therefore the model must output a predicted start and end index for the answer span. The output layer outputs two probability distributions $\boldsymbol{p}^{(s)}$ and $\boldsymbol{p}^{(e)} \in \mathbb{R}^{L_c}$ over the context tokens for the start index and end index. We found the following setup to work slightly better than the output layer in the original BiDAF paper.

First a fully connected neural network layer with ReLU activation is applied to the output of the modeling layer:

$$\tilde{\boldsymbol{m}}_i = ReLU(X\boldsymbol{m}_i) \quad \forall i = 1, ..., L_c, \tag{7}$$

where $X \in \mathbb{R}^{h \times 2h}$ is a trainable weight matrix.

We then calculate

$$l_i^{(s)} = \boldsymbol{w}_s^\top \tilde{\boldsymbol{m}}_i \qquad \forall i = 1, ..., L_c \tag{8}$$

$$l_i^{(e)} = \boldsymbol{w}_e^\top \tilde{\boldsymbol{m}}_i \qquad \forall i = 1, ..., L_c \tag{9}$$

and hence obtain the start and index probability distributions:

$$\boldsymbol{p}^{(s)} = \text{softmax}(\boldsymbol{l}^{(s)}) \tag{10}$$

$$\boldsymbol{p}^{(e)} = \text{softmax}(\boldsymbol{l}^{(e)}). \tag{11}$$

The training objective is to minimize the mean cross-entropy loss of the predicted start and end indices:

$$L(\boldsymbol{\theta}) = -\frac{1}{N} \sum_{j=1}^{N} \Big(\log(p^{(s)}_{y_i^{(s)}}) + \log(p^{(e)}_{y_i^{(e)}}) \Big), \tag{12}$$

where $\boldsymbol{y}^{(s)}$ and $\boldsymbol{y}^{(e)}$ are the true start and end indices for the answer spans of the N training examples and $\boldsymbol{\theta}$ contains all trainable parameters of the model.

At test time, the starter code chooses the pair of start and end indices (i, j) for which the individual probabilities $p_i^{(s)}$ and $p_j^{(e)}$ are maximised. However we use the slightly more effective policy of choosing the pair (i, j) whose probabilities have maximum product $p_i^{(s)} p_j^{(e)}$.

3 Experiments

3.0.1 Configuration

We implemented the model in Tensorflow based on the starter code provided. Efficient batch execution on a GPU required us to fix all context and question sequences to be the same length, padding all shorter inputs with a special PAD token. We used a context length of 600 tokens and a question length of 30 tokens. We used 100 dimensional GloVe vectors trained on a 6 billion word corpus. GloVe vectors were available for the 400,000 most frequent tokens, and tokens with no corresponding GloVe vector were replaced with a special UNK token and an associated vector. For all RNNs we used a hidden layer size $h = 200$ and applied dropout [7] with probability 0.15 to the output. Neither increasing nor decreasing the dropout probability seemed to improve the performance. The character-level CNN used $f = 100$ filters of width 5 and was applied to words which were padded to fixed length 20. When character-level embeddings were not used, the model seemed to perform just as well with $h = 100$ as it did with $h = 200$. We allowed backpropagation into the $V = 1000$ most frequent embeddings, including the embeddings for PAD and UNK. We used the ADAM optimiser with a learning rate of 0.001 and clipped gradients to a maximum of 5. Each model was trained until the performance on the development set plateaued.

3.0.2 Ablations

In order to determine the value of each component of the model, we have done an experiment with each one removed. The results are shown in Table 1.

Configuration	**Dev F1 score**	**Dev EM score**
Original BiDAF implementation	77.3%	67.7%
Our final model	76.6%	66.0%
No variable embeddings, i.e. $V = 0$	75.9%	65.5%
No character-level CNN	75.2%	64.5%
Predict on individual start and end probabilities	75.6%	65.4%
Starter code	43.7%	34.9%

Table 1: Experimental results: dev set performance

We see that the use of variable word embeddings is worth 0.7%, the character-level CNN is worth 1.4%, and predicting based on the product of start and end index probabilities is worth 1.0%. In the ablation analysis in the original BiDAF paper, the use of a chararacter-level CNN was found to be worth 1.9% which seems significantly higher than our gain, and accounts for the overall difference in performance. This may be because we lowercased all tokens before feeding them into the CNN, which might be suboptimal because upper-case letters carry some valuable information such as indicating proper nouns and acronyms.

4 Error Analysis

We will now analyse the errors made by the model in its predictions on the development set. We find that the model almost always gives an answer that bears some relation to the question. However the model often fails to appreciate the subtleties of the question and hence returns the wrong information about the right subject. This is a typical example:

Context: *There are some hills (mostly artificial) located within the confines of the city – e.g. Warsaw Uprising Hill (121 metres (397.0 ft)), Szczęśliwice hill (138 metres (452.8 ft) – the highest point of Warsaw in general).*

Question: How high is the highest point in Warsaw?

Correct answer: 115.7 metres
Prediction: Szczęśliwice hill

The model has correctly identified the location in question, but has provided its name rather than its height.

Context: *Oxygen is a chemical element with symbol O and atomic number 8. It is a member of the chalcogen group on the periodic table and is a highly reactive nonmetal and oxidizing agent that readily* *forms compounds (notably oxides) with most elements. By mass, oxygen is the third-most abundant element in the universe, after hydrogen and helium.*

Question: What are the three most abundent elements of the universe by mass?

Correct answer: oxygen is the third-most abundant element in the universe, after hydrogen and helium
Prediction: hydrogen and helium

This is another instance of the model failing to fully understand the question. This is a particularly difficult example because it requires the model to be able to count, a difficult ability to learn given how few questions there are of this type.

A different type of error which the model makes is characterised by excessively long predictions where the correct answer occurs at one edge of the predicted span.

Context: *The Broncos' defense ranked first in the NFL yards allowed (4,530) for the first time in franchise history, and fourth in points allowed (296). Defensive ends Derek Wolfe and Malik Jackson each had 5½ sacks. Pro Bowl linebacker Von Miller led the team with 11 sacks, forced four fumbles, and recovered three.*

Question: What two Denver players ranked at 5 percent for sacks?

Correct answer: Derek Wolfe and Malik Jackson
Prediction: Derek Wolfe and Malik Jackson each had 5½ sacks. Pro Bowl linebacker Von Miller

Context: *The official record high temperature for Fresno is 115 °F (46.1 °C), set on July 8, 1905, while the official record low is 17 °F (-8 °C), set on January 6, 1913. The average windows for 100 °F (37.8 °C)+, 90 °F (32.2 °C)+, and freezing temperatures are June 1 thru September 13, April 26 thru October 9, and December 10 thru January 28, respectively, and no freeze occurred between in the 1983/1984 season. 1962.*

Question: On what date was the record low temperature in Fresno?

Correct answer: January 6, 1913
Prediction: July 8, 1905, while the official record low is 17 °F (-8 °C), set on January 6, 1913

At first these appear to simply be wild errors, but there is a pattern. The substrings occurring at each edge of the predicted spans ("Derek Wolfe and Malik Jackson" and "Von Miller," and "July 8, 1905" and "January 6, 1913") both bear some relation to the question, as they are both "players who achieved some number of sacks" and "days on which temperature records were set" respectively. Since the model only considers the probabilities of each token being the start and end of the answer span, nothing prevents it from including a large number of clearly irrelevant tokens in the middle of its prediction if the start and end tokens appear relevant.

The model makes a variety of other less severe "imprecise boundary" type errors. This is an example of a "prediction too short" error:

Context: *Unlike conventional cilia and flagella, which has a filament structure arranged in a 9 + 2 pattern, these cilia are arranged in a 9 + 3 pattern, where the extra compact filament is suspected to have a supporting function. These normally beat so that the propulsion stroke is away from the mouth, although they can also reverse direction. Hence ctenophores usually swim in the direction in which the mouth is pointing, unlike jellyfish. When trying to escape predators, one species can accelerate to six times its normal speed; some other species reverse direction as part of their escape behavior, by reversing the power stroke of the comb plate cilia.*

Question: What direction do ctenophore swim?

Correct answers: in the direction in which the mouth is pointing, the direction in which the mouth is pointing, direction in which the mouth is pointing
Prediction: the mouth is pointing

Here the human judges do not agree perfectly on where the answer should begin, but the model's prediction is clearly inappropriate, since although it contains the correct information, it does not sound like a coherent answer to the question. On the other hand the model sometimes provides more information than is relevant:

Context: *Genghis Khan ... is also given credit for the introduction of the traditional Mongolian script and the creation of the Ikh Zasag (Great Administration), the first written Mongolian law. "Ikh Zasag law* *adopted during Genghis Khan's time in Mongolia had points to punish illegal matters related to corruption and bribery very heavily," Mongolian President Tsakhiagiin Elbegdorj noted.*

Question: What did Mongolian President Tsakhiagian Elbegdorj note was significantly punished by Genghis Khan's laws?

Correct answer: corruption and bribery
Prediction: illegal matters related to corruption and bribery very heavily

These errors have something in common: we could be reasonably common that the predictions are incorrect without even needing to read the context paragraph. They are either not coherent answers to any question, or not possible answers to the question asked. This suggests a possible means of improving the performance of the model by adding another component which takes as input only the question and a potential answer, and predicts how plausible the answer is for the question. At test time, the top K most likely spans generated by the main model would be fed into the plausibility model, whose output would be used to inform the final prediction. Training such a model would require us to generate a number of negative training examples which are (a) unrelated to the question and (b) related but incoherent. We could generate examples of type (a) by extracting spans of the context which are distant from the correct spans and type (b) by generating spans which overlap but do not match the correct spans.

5 Summary

We implemented the Bidirectional Attention Flow reading comprehension model and were able to reproduce similar results to those reported in the original paper on the SQuAD dataset. By allowing the 1000 most frequent GloVe word vectors to vary, we were able to gain an 0.7% improvement over our unmodified BiDAF implementation. Our error analysis suggested that the majority of errors made by the model could be detected without needing to read the context paragraph. Therefore we hypothesised that a fruitful avenue of improvement might be to train a separate model which would check whether a prediction of the main model is a plausible answer to the question or not.

References

[1] K. Cho et al. *Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation.* 2014. URL: `https://arxiv.org/abs/1406.1078`.

[2] S. Hochreiter and J Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: `10.1162/neco.1997.9.8.1735`. URL: `http://dx.doi.org/10.1162/neco.1997.9.8.1735`.

[3] Y. Kim. *Convolutional Neural Networks for Sentence Classification.* 2014. URL: `https://arxiv.org/abs/1408.5882`.

[4] J. Pennington, R. Socher, and C. D. Manning. *GloVe: Global Vectors for Word Representation.* 2014. URL: `https://nlp.stanford.edu/pubs/glove.pdf`.

[5] P. Rajpurkar et al. *SQuAD: 100,000+ Questions for Machine Comprehension of Text.* 2016. URL: `https://arxiv.org/abs/1606.05250`.

[6] M. Seo et al. *Bidirectional Attention Flow for Machine Comprehension.* 2016. URL: `https://arxiv.org/abs/1611.01603`.

[7] N. Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15 (2014), pp. 1929–1958. URL: `http://jmlr.org/papers/v15/srivastava14a.html`.