SQuAD Reading Comprehension Task - CS224n Final Project

Adva Wolf
Stanford University
codelab: advaw
advaw@stanford.edu

Abstract

The Stanford Question Answering Dataset (SQuAD) contains more than 100000 questions and paragraphs with highlighted answers, taken from Wikipedia. Our model uses Bidirectional attention flow and predicts the location of the highlighted answer with an accuracy of 67.6% F1 and 52.2% EM (exact match). We also investigate the effect of different span selections.

1 Introduction

The SQuAD challenge allows us to measure how well machine learning systems can understand text: given inputs (paragraphs from Wikipedia articles) and questions about them, can our model extract the answers from the inputs? Our output is the "span" of text in the paragraph that answers the question. We compare our model span to the ground truth, which is based on crowdsourced answers. We compare by using two measures: The exact match, which is the percentage of predictions that match any one of the ground truth answers exactly, and the F1 score metric, a looser metric that measures the harmonic average between precision (what percentage of our span also appears in the ground truth?) and recall (what percentage of the ground truth appears in our span?).

Our approach is mostly based on a prior work by Seo et al. [1] - also known as BiDAF. Our model includes word embedding, BiLSTM encoder, bi-directional attention flow layer, a BiLSTM modeling layer and an output layer. We investigate different models for predicting the span: we can predict the start position of the span, the end position and / or the length of the span. We try different combinations of the above and analyze their performance.

2 Related Work

The SQuAD databased allows many researchers to investigate different deep learning and NLP techniques, as it can be seen in the project website: https://rajpurkar.github.io/SQuAD-explorer/. We will survey the BiDAF model approach and other models working with more complex span predictions.

2.1 BiDAF

The BiDAF model contains word and character embedding, BiLSTM encoder, bi-directional attention flow layer, double BiLSTM modeling layer and an output layer. The output layer obtains the probability distribution of the start index by applying a weight matrix on the output of the previous layers and a softmaxing the result. In order to get the probability distribution of the end index, the model applies an additional LSTM layer to the output of the modeling layer, and applies the same method as for the start index (with different weights). We notice that in this model, the start position and the end position are predicted independently. The core part of the BiDAF model is its attention layer, which we implemented in our model. Attentions are a key ingredient in many NLP models, since they allow us to focus on a particular part of the input that is relevant to the current state. Here, the attention is computed in two directions: from context to question and from question to context. Both of these attentions are calculated from a shared similarity matrix between the embeddings of the context and the question, where the i, j entry of this matrix indicates the similarity between the i-th context word and j-th question word.

2.2 More complex predictions

One of the problems with predicting the start position and the end position independently is the possibility that the most probable option has the end position before the start position. Different models deals with this problem. For example, the DrQA model [3] chooses the best span from word i to word i' such that $i \leq i' \leq i + 15$ and $P_{start}(i)P_{end}(i')$ is maximize. Other models, such as RaSOR [2] or DCR [4] define a probability distribution over all possible answer spans.

3 Approach

The layout of our model can be seen in the following figure. This figure is based on the BiDAF [1] model architecture figure, and the layers that we omitted are blackened.

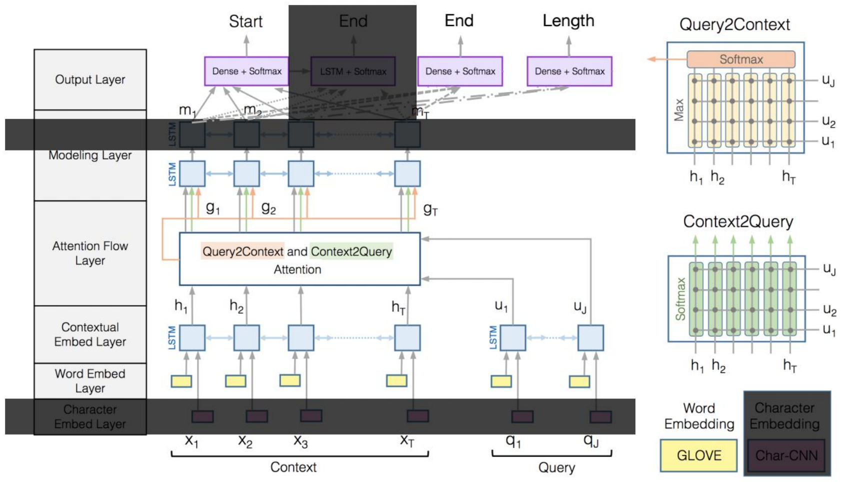

Figure 1: Histogram of the ground truth width

Word Embedding layer: maps each word in the context and question to a fixed size vector, using pre-trained word vectors, GloVe. The output of this layer is two sequences of d-dimensional vectors: $\mathbf{x_1}, \ldots, \mathbf{x_T}$ representing the context, and $\mathbf{q_1}, \ldots, \mathbf{q_J}$ representing the question.

biLSTM Encoder layer: 1-layer bidirectional LSTM (which share weights between the context and the question):

$$\{\mathbf{h_1}, \ldots, \mathbf{h_T}\} = \text{BiLSTM}(\{\mathbf{x_1}, \ldots, \mathbf{x_T}\}), \quad \mathbf{h_i} = [\overrightarrow{\mathbf{h_i}}; \overleftarrow{\mathbf{h_i}}] \in \mathbb{R}^{2h}$$

$$\{\mathbf{u_1}, \ldots, \mathbf{u_J}\} = \text{BiLSTM}(\{\mathbf{q_1}, \ldots, \mathbf{q_J}\}), \quad \mathbf{u_i} = [\overrightarrow{\mathbf{u_i}}; \overleftarrow{\mathbf{u_i}}] \in \mathbb{R}^{2h}$$

when $[\overrightarrow{\mathbf{h_i}}; \overleftarrow{\mathbf{h_i}}]$ is the concatenation of the forward and backward hidden state for the context, and similarly for the question.

Attention Flow Layer - BiDAF Attention: We implemented the attention layer described in Seo et al. [1]. This layer output is a question-aware representation of the context words, together with the output of the previous layer. We start with calculating the similarity matrix:

$$\mathbf{S_{tj}} = \mathbf{w_1^T h_t} + \mathbf{w_2^T u_j} + (\mathbf{w_3} \circ \mathbf{h_t})^{\mathbf{T}} \mathbf{u_j} \in \mathbb{R}$$

When $\mathbf{w_1}, \mathbf{w_2}, \mathbf{w_3} \in \mathbb{R}^{2h}$ are weight vectors. Notice that our formula is equivalent to the description in [1], and represents our implementation. Implementing the above formula for $\mathbf{S_{tj}}$ is more memory efficient than direct implementation of the original formula. Next, we use this matrix to calculate attention-aware vectors in both directions:

Context-to-question Attention. First we calculate for each context word attention distributions over the question words $\alpha^t = \text{softmax}(\mathbf{S_{t:}}) \in \mathbb{R}^J$, and we use them to calculate a weighted sum of the question hidden states, for each context word: $\tilde{\mathbf{u_t}} = \sum_{j=1}^{J} \alpha_j^t \mathbf{u_j} \in \mathbb{R}^{2h}$.

Quetion-to-context Attention. First we calculate the maximal element in the each column in the similarity matrix, and softmax the result: $\mathbf{b} = \text{softmax}(\max_{col} \mathbf{S}) \in \mathbb{R}^T$. This vector represents the weights on the context word, when the the most important words with respect to the question will have larger weight. The attended context is $\tilde{\mathbf{h}} = \sum_{t=1}^{T} b_t \mathbf{h_t} \in \mathbb{R}^{2h}$.

The output of this layer the following combination of the above:

$$\mathbf{g_t} = [\mathbf{h_t}; \tilde{\mathbf{u_t}}; \mathbf{h_t} \circ \tilde{\mathbf{u_t}}; \mathbf{h_t} \circ \tilde{\mathbf{h}}] \in \mathbb{R}^{8h}$$

Modeling layer: This layer contains 1-biLSTM layer applied to the sequence $\mathbf{g_t}$:

$$\{\mathbf{m_1}, \dots, \mathbf{m_T}\} = \text{BiLSTM}(\{\mathbf{g_1}, \dots, \mathbf{g_T}\}), \quad \mathbf{m_i} = [\overrightarrow{\mathbf{m_i}}; \overleftarrow{\mathbf{m_i}}] \in \mathbb{R}^{2h}$$

it captures the interaction between the question-aware context words. It is different than the encoder layer, since the encoder layer captured the interaction between the context words and question words independently. The original BiDAF implementations includes double biLSTM layer, and we decided to include only one since every biLSTM layer increases the running time of the model by almost twice.

Output layer: We experimented with four different models, sharing the above layers and differ in the output layer. All outputs layer include a fully-connected layer followed by a ReLU non-linearity, and we obtain the vectors $\mathbf{m_i}' \in \mathbb{R}^h$ after this layer.

The "naive" model: This model output layer predicted the start and end position independently, by applying the following to the vectors $\{\mathbf{m'_1}, \dots, \mathbf{m'_T}\}$:

$$p^{start} = \text{softmax}_i(w_{start}^T \mathbf{m'_i} + b_{start}) \quad p^{end} = \text{softmax}_i(w_{end}^T \mathbf{m'_i} + b_{end})$$

This model used the following loss function: $\text{loss} = -\log p^{start}(i_{start}) - \log p^{end}(i_{end})$, and the following predictions: $l^{start} = \text{argmax}_i p_i^{start}$, $l^{end} = \text{argmax}_i p_i^{end}$.

The conditioning model: This model differs from the above only in the choice of l^{start} and l^{end}. Here we choose the best span from word i to word i' such that $i \leq i' \leq i + 15$ and $p^{start}(i)p^{end}(i')$ is maximal, following DrQA [3].

The start-and-length model: This model output layer predicted the start position and the length of the span independently, by applying the following to the vectors $\{\mathbf{m'_1}, \dots, \mathbf{m'_T}\}$:

$$p^{start} = \text{softmax}_i(w_{start}^T \mathbf{m'_i} + b_{start}) \quad p^{len} = \text{softmax}_i(w_{len}^T \mathbf{m'_i} + b_{len})$$

This model used the following loss function: $\text{loss} = -\log p^{start}(i_{start}) - \log p^{len}(i_{len})$, and the following predictions: $l^{start} = \text{argmax}_i p_i^{start}$, $l^{end} = \text{argmax}_i p_i^{len} + l^{start}$. See the remark in the conclusion regarding this implementation for the length prediction.

The “triple-power” model: This model output layer predicted the start, end positions and the length of the span independently, by applying the following to the vectors $\{\mathbf{m'_1}, \dots, \mathbf{m'_T}\}$:

$$p^{start} = \text{softmax}_i(w_{start}^T \mathbf{m'_i} + b_{start}) \quad p^{len} = \text{softmax}_i(w_{len}^T \mathbf{m'_i} + b_{len})$$
$$p^{end} = \text{softmax}_i(w_{end}^T \mathbf{m'_i} + b_{end})$$

This model used the following loss function:

$$\text{loss} = -\log p^{start}(i_{start}) - \log p^{len}(i_{len}) - \log p^{end}(i_{end})$$

Here we choose the best span from word i to word $i+k$ such that $0 \leq k \leq 15$ and $p^{start}(i)p^{len}(k)p^{end}(i+k)$ is maximal. See the remark in the conclusion regarding this implementation for the length prediction

4 Experiments

In this section we evaluate and analyze our models performance. We start with the F1 and EM scores on the official SquAD DEV set:

Table 1: F1/EM results on DEV set

Model	EM	F1
Naive	50.9	65.5
Conditioning	**52.2**	**67.6**
Start-And-Length	35.3	58.0
Triple-Power	52.0	66.6
BiDAF (Seo et al. implementation, single mode)	**67.7**	**77.3**

4.1 Error analysis

We analyze the errors in our models quantitively based on the answer lengths, start, end positions and the question type. In addition, we explore specific examples.

4.1.1 Performance on specific question type

Table 2: Performance on specific question type

question containing:	’what’		’why’		’where’		’who’	
Total number of such questions:	5981		153		478		1227	
	EM	F1	EM	F1	EM	F1	EM	F1
Naive	51.2	65.7	53.6	64.7	48.7	62.0	50.0	65.3
Conditioning	**52.3**	**67.4**	**60.1**	**74.1**	**51.5**	**66.0**	**50.0**	**66.3**
Start-And-Length	35.3	58.0	45.7	66.4	36.0	57.4	34.9	56.6
Triple-Power	51.9	66.8	58.2	68.1	51.5	64.0	50.4	65.3

question containing:	’how’		other	
Total number of such questions:	1173		1379	
	EM	F1	EM	F1
Naive	50.0	64.4	51.5	66.8
Conditioning	**53.2**	**67.7**	52.4	**69.2**
Start-And-Length	34.7	57.7	34.9	58.9
Triple-Power	52.1	67.3	**52.9**	67.5

In Table 2 we summarized the performance of each model on each question type. The results are quite similar to the total EM/F1 evaluations - the conditioning model has the best performance. We also see that all the models performed exceptionally well on questions of type “why”.

4.1.2 Performance on specific location of the answer

We measure the accuracy of our models based on the starting and ending positions: For intervals of 10 consecutive positions, we average the F1 and EM scores for all examples in the DEV set with ground truth answer starting / ending at this interval. If our interval included less than 25 examples, we set the average to be zero. The results are summarized in the graphs below:

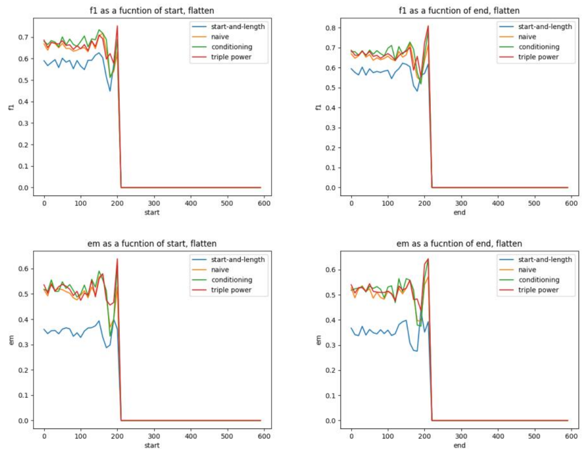

Figure 2: Performance on start and ending position location

In order to understand this results better, it would be also useful to look at the histogram of the ground truth start and ending positions:

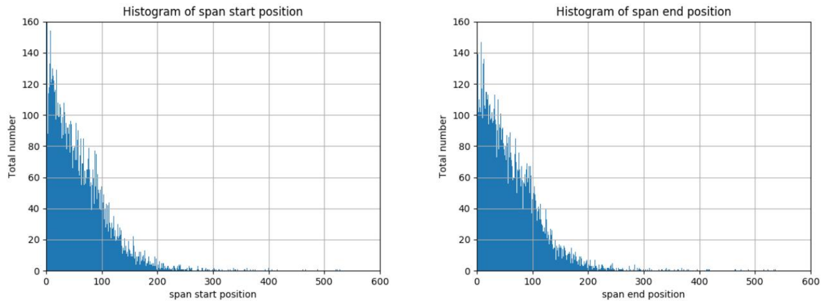

Figure 3: Histogram of the ground truth location

Overall, the performance of the different models looks pretty uniform, with the conditioning model with the best results. We also see that starting from the ~ 150 location, the number of examples that we have is quite small, and the jumps that we have in the average scores are probably due to noise. We do see a slight advantage for the "triple power" model for answers that are located at the first sentence in the paragraph, which happens frequently. However the conditioning model performs better in all the other locations, and has a better score overall.

4.1.3 Performance on specific width of the answer

We measure the accuracy of our models based on the length of the ground truth answer: For each length, we average the F1 and EM scores for all examples in the DEV set with ground truth with that length. If the number of examples in that length included less than 25 examples, we set the average to be zero. The results are summarized in the graphs below:

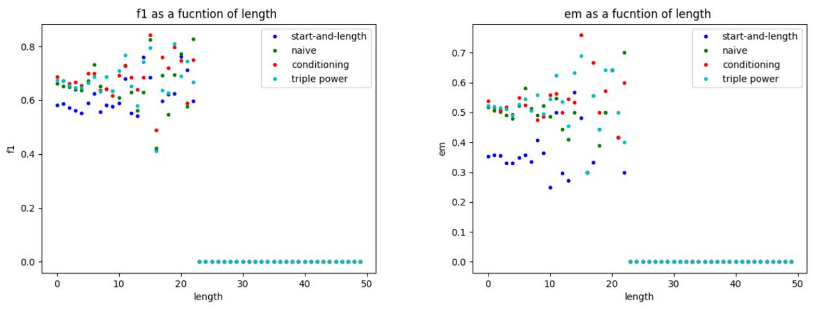

Figure 4: Performance on start and ending position location

In order to understand this results better, it would be also useful to look at the histogram of the ground truth answers length:

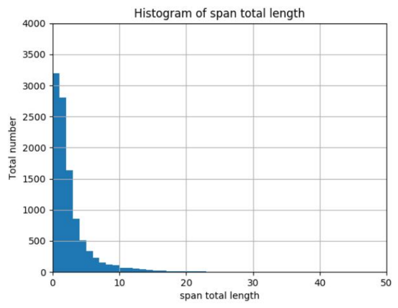

Figure 5: Histogram of the ground truth width

Also here we see the effect of noise starting from length of size ~ 15. We notice that the most frequent option (length size equal to 0, i.e. the golden truth answer is a single word) represent the total performance that we get on the total DEV set.

4.1.4 Exploring examples

We are interested in examples in which some of models were correct and other failed. We will try to explain the specific reasons for it in each example.

Example 1 - predicting the length helps in short answers:

- context:as well as being added , forces can also be resolved into independent components at right angles to each other . a horizontal force pointing northeast can therefore be split into two forces , one pointing north , and one pointing east . summing these component forces using vector addition yields the original force . resolving force vectors into components of a set of basis vectors is often a more mathematically clean way to describe forces than using magnitudes and directions . this is because , for orthogonal components , the components of the vector sum are uniquely determined by the scalar addition of the components of the individual vectors . orthogonal components are independent of each other because forces acting at ninety degrees to each other have no effect on the magnitude or direction of the other . choosing a set of orthogonal basis vectors is often done by considering what set of basis vectors will make the mathematics most convenient . choosing a basis vector that is in the same direction as one of the forces is desirable , since that force would then have only one non-zero component . orthogonal force vectors can be three-dimensional with the third component being at right-angles to the other two .
- question: what can orthogonal forces be when there are three components with two at right angles to each other ?

- predictions: naive and conditioning: "no effect on the magnitude or direction", start and length: "no", triple power: "three-dimensional".
- ground truth:"three-dimensional"

Here we see that the "triple power" model has an advantage in small sized answers. However, when depending completely on the start and the length prediction, we get the same wrong answer, just must shorter. Hence our chance for good $F1$ score drop with the "start and length" model.

Example 2 - length prediction ruins the final answer:

- context:a resurgence came in the late 19th century , with the scramble for africa and major additions in asia and the middle east . the british spirit of imperialism was expressed by joseph chamberlain and lord _rosebury_ , and implemented in africa by cecil rhodes . the _pseudo-sciences_ of social darwinism and theories of race formed an ideological underpinning during this time . other influential spokesmen included lord cromer , lord curzon , general _kitchner_ , lord milner , and the writer rudyard kipling . the british empire was the largest empire that the world has ever seen both in terms of landmass and population . its power , both military and economic , remained unmatched .
- question: in which continent besides asia were major gains made by the british empire in the late 19th century ?
- predictions: naive: EMPTY, conditioning: "asia and the middle east", start and length and triple power: "africa".
- ground truth: "middle east"

Here we see that the naive model predict the start position after the end position, resulting in an empty string. In addition, it is not suprizing (based on the length histogram) that the models that use the length prediction would tend to predict short answer. Apparently here, the attention-aware context gave larger weight to possible locations, and together with a condition for a single word answer, "africa" was the most possible answer for the latter two models. We also see here that predicting long answers can be better than predicting short ones in terms of the F1 score: even if the start / end position were wrong, limiting ourselves to short answers decreases the chance for a positive F1. We remark here that we can also see this phenomenon in Table 1: the difference between the conditioning model (our best model) and the "triple power" model is larger in the F1 score.

4.2 Model details:

In all the models, we used a learning rate of 0.001, dropout of 0.15, batch size of 100 and our hidden size d is 200. Our models trained on $\sim 15k$ examples from the training set.

5 Conclusion

In total, we have seen that predicting the length of the answer didn't improve our model, and even worsen it when relying on it too much (start-and-len model). In retrospect, even for a human reader, predicting the length of the answer in a reading comprehension task can be a complex task: we can probably say if the answer would be short / long, but in order to predict an exact number we would most likely first find the answer, and then count it length. The most significant improvements to our model baseline came from the modeling layer. We've also see that studying our models performance based on the location of the answer mimicked the performance on the whole DEV set, which shows that our models are flexible. As future improvements, adding one more BiLSTM layer to the modeling layer would most likely boost our models performances (it was omitted in our implementation due to running time considerations).

remark: we treated the prediction of the length distribution the same as the start and end location prediction: however, conceptually they differ and predicting that the length is i doesn't necessarily rely on the i-th context word. Our current architecture forces a connection between the two, which might explain the poor results we obtained. A better architecture would produce a length prediction that depends on all the outputs of the model layer together - e.g. by adding a NN layer that takes as an input the sum of these vectors, and output a single vector. Unfortunately, understanding this flaw in our models was very close to the due date on the project, and training the better architecture couldn't have been completed by the due date for this project.

Acknowledgments

I would like to thank the CS-224n staff for their guidance and help though the course and this project.

6 References

[1] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

[2] Kenton Lee, Shimi Salant, Tom Kwiatkowski, Ankur Parikh, Dipanjan Das, Jonathan Berant.Learning Recurrent Span Representations for Extractive Question Answering. *arXiv preprint arXiv:1611.01436,* 2017.

[3] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*, 2017.

[4] Yang Yu, Wei Zhang, Kazi Hasan, Mo Yu, Bing Xiang, and Bowen Zhou. End-to-end answer chunk extraction and ranking for reading comprehension. *arXiv preprint arXiv:1610.09996*, 2016.