Building a QA system (IID SQuAD track)

Stanford CS224N {Default} Project

Youssef Aitousarrah & Naoufal Layad
Stanford University (Department of Energy & ICME)
aitous@stanford.edu & nlayad@stanford.edu

Abstract

The ability of reading text and then answering questions about it, is a very challenging task for machines, because it is requiring both understanding of natural language and knowledge about the world as well as the context of the question. In this paper, we will build a QA system for the SQuAD 2.0 dataset, which extends the original dataset with unanswerable questions. We have already a fully-functional neural baseline, our goal is to improve this baseline, using new techniques and models used in high performing SQuAD models. We are also adding some changes to our models, to see if these changes increase the accuracy of our model. First, we will extend the baseline model to match the original BiDAF model by including a character-level embedding layer. Furthermore, we will replace the basic Attention Layer of our baseline by different dynamic attention layers such as a Dynamic Co-attention Layer, Self-Matching-Attention Layer as well as adapting transformers ideas to question answering. We also add another BiDAF layer performing a self attention process similar to the one between the context and the query. A correct implementation of these changes improves our model, such as the character-level embedding layer which allows our model to handle out-of-vocabulary words and extract information from words' morphology.

1 Approach

Our baseline model is a hierarchical multi-stage process, it is based on **Bidirectional Attention Flow (BiDAF)** [3], but using only word-level embeddings. As a first step to improve our system, we include **character-level word embeddings**. Our QA model will consist of six layers, and we will try different approaches for the 4^{th} layer (corresponding to attention) to increase the accuracy of our model. We will add a **Co-attention layer**, **Self-attention layer (R-Net)** and a **QANet** layer. We will try also adding another BiDAF layer performing a self attention process similar to the one between the context and the query.

1.1 Character Embedding Layer

This layer maps each word to a vector space using character-level embedding convoluted and max-pooled using CNNs. We obtain the character-level embedding of each word using Convolutional Neural Networks (CNN), characters are embedded into vectors and constitute the inputs of the CNN, and the outputs of this CNN are max-pooled over the entire width to obtain a fixed-size vector for each word. The resulting representation is concatenated with the word embeddings we obtain via GloVe.

1.2 Word Embedding Layer

This layer maps each word to a vector space using a pre-trained word embedding model GloVe, to obtain the fixed word embedding of each word. We convert the words of the context and the questions into words embeddings, $c_1, ...c_N \in \mathbf{R}^D$ for the context and $q_1, ...q_M \in \mathbf{R}^D$ for the question, where N and M are the size of the context and the question respectively.

We concatenate the character and word embedding vectors. Then, we refine the concatenation by projecting an embedding vector v_i using a learnable matrix W_{proj}, so we obtain $h_i = W_{proj} v_i \in \mathbf{R}^H$ and by applying a two-layer highway network[5], which is consisting on applying a one-layer highway network twice[5].

$$g = \sigma(W_g h_i + b_q) \in \mathbf{R}^H$$

$$t = ReLU(W_t h_i + b_t) \in \mathbf{R}^H$$

$$h_i' = g \odot t + (1 - g) \odot h_i \in \mathbf{R}^H$$

where W_g, W_t, b_g, b_t are learnable parameters.

1.3 Encoder Embedding Layer

In this layer, we use an (LSTM) on top of the embeddings provided by the previous layers to model the temporal interactions and dependencies between words.

1.4 Description of different approaches used as a 4th layer.

1.4.1 Co-Attention Flow Layer

In the Attention layer, we construct query and context vectors and produces a set of query-aware feature vectors for each word in the context. Instead of this basic attention layer, we involve a second-level attention computation, attending over representations that are themselves attention outputs. Concretely, we first start by applying a non-linearity to a query representations $q_1, q_2, ..., q_M \in \mathbb{R}^{2H}$ after applying a linear layer, the equation for this first transformation is:

$$\boldsymbol{q}_j^{'} = \texttt{tanh}(\boldsymbol{W}\boldsymbol{q}_j + \boldsymbol{b}) \in \mathbb{R}^{2H}.$$

Next, we compute an affinity matrix $\boldsymbol{L}$ where each element (i, j) contains the dot product of the context and query vectors $\boldsymbol{L}_{ij} = \boldsymbol{c}_i^T \boldsymbol{q}_j^{'}$. We add also a sentinel context and query vectors to make it possible to attend none of the provided hidden states.
For the Context-to-Question (C2Q) Attention, we compute the coefficients as follows:

$$\alpha^i = \texttt{softmax}(\boldsymbol{L}_{i,:}) \in \mathbb{R}^{M+1},$$

$$\boldsymbol{a}_i = \sum_{j=1}^{M+1} \alpha_j^i \boldsymbol{q}_j^{'} \in \mathbb{R}^{2H}$$

For Question-to-Context Attention:

$$\beta^j = \texttt{softmax}(\boldsymbol{L}_{:,j}) \in \mathbb{R}^{N+1},$$

$$\boldsymbol{b}_j = \sum_{i=1}^{N+1} \beta_i^j \boldsymbol{c}_i \in \mathbb{R}^{2H}$$

We use these and the alpha coefficients we computed earlier to compute a second-level attention output:

$$\boldsymbol{s}_i = \sum_{j=1}^{M+1} \alpha_j^i \boldsymbol{b}_j$$

We concatenate the outputs with the a_i we obtained earlier and feed the resulting vectors to a bidirectional LSTM to get what we call coattention encoding:

$$\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_N\} = \texttt{biLSTM}(\{[\boldsymbol{s}_1; \boldsymbol{a}_1], \ldots, [\boldsymbol{s}_N; \boldsymbol{a}_N]\})$$

1.4.2 Self-Matching-Attention (R-Net)

R-Net is a high-performing SQuAD model that has both a Context-to-Question attention layer (similar to our baseline), and a self-attention layer (which they call Self-Matching Attention) [6].

In [6], they proposed a **gated attention-based** recurrent network to include information from the question into text representation. It is another variant of attention-based recurrent networks, with an additional gate to quantify the importance of information in the passage regarding a question. They propose generating ¨ sentence-pair representation $(v_t^{context}), t = 1, ...N$:

$$v_t^{context} = \texttt{RNN}(v_{t-1}^{context}, a_t),$$

where $a_t = \texttt{att}(q, [c_t, v_{t-1}^{context}])$ is an attention-pooling vector of the whole question q.

$$s_j^t = v^T \texttt{tanh}(W_q q_j + W_c c_j + W_v v_{t-1}^{context}),$$

$$a_i^t = \frac{\texttt{exp}(s_i^t)}{\sum_{j=1}^{M} \texttt{exp}(s_j^t)},$$

$$a_t = \sum_{j=1}^{M} a_i^t q_i$$

Each passage representation $v_t^{context}$ dynamically incorporates aggregated matching information from the whole question.

Other researchers (Wang & Jiang 2016) has also introduced a match-LSTM, which takes c_t as an additional input into the recurrent network.

$$v_t^{context} = \texttt{RNN}(v_{t-1}^{context}, [c_t, a_t]),$$

After the computation of the question-aware passage representation $v_t^{context} t = 1, ...N$. This representation has very limited knowledge of context, and also the representation my be poor if there is a lexical divergence between the question and context. To solve this issue, we propose directly matching the question-aware passage representation against itself, by introducing new vectors of self-attention $h_t^{context}$, defined in the same as before

$$h_t^{context} = \texttt{RNN}(h_{t-1}^{context}, [v_t, a_t]),$$

where $a_t = \texttt{att}(v, v_t)$ is an attention-pooling vector of the whole question $v^{context}$.

$$s_j^t = v^T \texttt{tanh}(W_v^{context} v_j^{context} + W_v^{context'} v_t^{context}),$$

$$a_i^t = \frac{\texttt{exp}(s_i^t)}{\sum_{j=1}^{M} \texttt{exp}(s_j^t)},$$

$$a_t = \sum_{j=1}^{M} a_i^t v_i^{context}$$

The additional gate used in the attention-based Recurrent Networks control the input of RNN.

Self-matching extracts evidence from the whole passage according to the current passage word and question information [6].

1.4.3 QANet

In order to mitigate the computational expensiveness of the RNNs models, We have also experimented with an implementation of the QANet model. THe model is based only on convulsions and self-attention layers. Using a convolutional structure allows to process the tokens in parallel, it allows us to use methods specific to the ConvNets like layer dropout.

We use the same attention layer in the BiDAF model. What is new here is the use of an encoding layer containing Convolutions. These encoding layers are structured as follows [convolution-layer x # + self-attention-layer + feed-forward-layer]. The number of conv layers within a block is 4. Each of the operations is placed inside a residual block. This encoder layer is repeated 3 times after the Context-Query Attention layer. These 3 layers share the same weights.

1.4.4 Double BiDAF

We have also experimented with an idea that we had when reading the papers about self attention. The attention layer in the BiDAF model only accounts for the Context-Query attention. We wanted somehow to account also for the Context-to-Context interaction, this is will provide valuable information about the co-dependence between different words in the context.

To put this idea into practice we have added another BiDAF layer performing a self attention process similar to the one between the context and the query. The input to this layer will be the representation we get from the first BiDAF attention layer and the words context representations we get from the first encoder. The output of this layer will successfully account not only for the interactions between the context and question and also for the ones within the context. This is the model that provided the highest score.

We have also being experimenting with additional gates and nonlinearities applied to the summary vector after the attention step. These gates and nonlinearities enable the model to focus on important parts of the attention vector for each word.

1.5 Modeling Layer

The output of the this layer captures the interaction among the context words conditioned on the query [3], which is different from the Encoder embedding layer, that catches the interaction among context words independent of the query.

1.6 Output Layer

The output layer produces a vector of probabilities corresponding to start and end position of the answer, in the context. It applies a bidirectional LSTM to the modeling layer's output and then concatenate the results with the co-attention layer's outputs before projecting them using W_{start} and W_{end} and taking their softmax to produce two vector of probabilities p_{start} and p_{end}.

When we concatenate the forward and backward hidden states from the bidirectional RNN, we are trying many types of concatenation such as averaging or max pooling them.

2 Experiments

In this section, we will describe how we train and evaluate our QA model:

2.1 Data

We will be using the original training set of SQuAD, however we will be using custom dev and test sets, which are obtained by splitting the official dev set in half into a dev and test sets, because the default test set is entirely secret.

2.2 Evaluation method

We evaluate on the dev or test sets, by taking the maximum F1 and EM scores across the three human-provided answers for that question, which makes our evaluation more forgiving. The EM and F1 scores are averaged across the entire evaluation dataset to get the final reported scores. The main metric here that we will try to maximize is the F1 score because the leaderboard ranking will be based on it.

2.3 Experimental details

We kept the same configuration from the baseline model. However, there are several new parameters that need to be optimized and chosen well. These parameters are related to the CNN layer for character-level embedding and to the Coattention layer. For the CNN layer, there are 2 main parameters, the number of channels and the channel width (second dimension of the kernel). Regarding the Coattention layer, it is the hidden dimension that need to be chosen. The convolutional layer have a kernel size and a number of convolutions in each block.

Hyperparameters used for the model:

parameter	value
hidden size	100
dropout	0.1
batch size	64
char channel size	16
char channel width	4
num attention heads	1
num conv layer	2
kernel size	3

Table 1: Hyperparameters for the different models with different attention layer.

We did not have time to fine-tune these hyperparameters as we have been experimenting with multiple models. The long training time made it very difficult to perform some kind of grid search.

2.4 Results

We have results for the character-level model and the Coattention + character-level embeddings model, and we compare these to the baseline model.

	AvNA	EM	F1
baseline	65	55	58
Only Character embedding	63	57.6	60.6
Coattention	62.9	51.4	54.5
Coattention + RNN	66.54	56.76	60.01
QANet	66.19	57.23	60.19
Double BiDAF	69.16	59.62	63.03

Table 2: Results for different models

We can see from the previous table that we were able to improve the F1 score by using a CNN layer to learn a character-level word representation. For the Coattention model, we were not able to enhance the results and we are suspecting an error in the implementation and working on debugging the code.

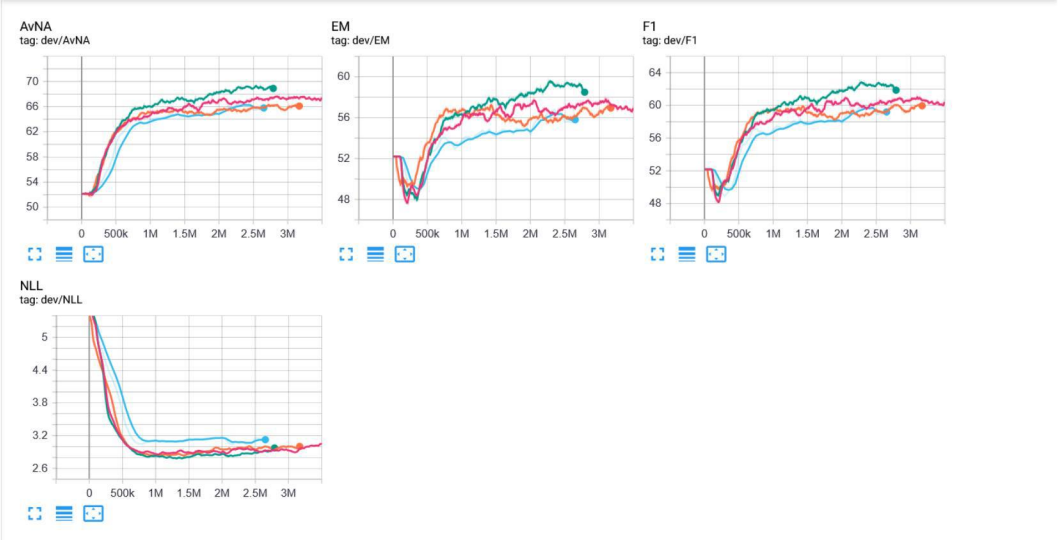

Figure 1: Dev set metrics (Green = Double BiDAF + gates + nonlinearities, Pink = Double BiDAF, Orange = QANet, Bleu = CoAttention + RNN)

3 Analysis & Discussion

The objective of our work was to examine the different ideas and compare them to each other to see which factors play an important role in improving the model predictions.

We have came to the conclusion that the CoAttention model was not involved enough to represent the interactions between the context and the question. The reason is that we are using the same matrix containing the dot product between a transformed form of the query and the context. This same matrix is used to produce the summary representation of the context word and the query word. We have also experimented with additive attention, but the improvement was not pronounced.

The R-Net has a promising idea of using the attention vector dynamically. At each step of the RNN layer, we compute an attention vector that is dynamically depending on the hidden state. Doing this allows us to represent better the relationship between the context and the question and also between context words too. The problem with this method is the computations get expensive as we have to handle each word at an RNN step separately (dependence). To improve this computational requirement, we tried a static version of this where we pass the attention output through a RNN layer. The model got better computationally but we have not seen huge accuracy improvement.

The QANet explores a different approach. We implemented this model to compare the performance of the ConvNets in comparison to RNN. The model gives decent results that would have been improved if we had increased the number of convolutional layers in each block. The ability of the model to process tokens in parallel makes it really attractive and provide a speedup that can be taken advantage of by training it on more data and training it for longer.

Double BiDAF is the model we have devised ourselves. We got inspired from the idea of accounting for the relationship between words in the same context. Thus, the prediction of the answer span takes into account the dependency between the context words and not only the relationship context-query. This idea improved the F1 score by 5 points. This is very promising because we haven't increase the model size by a lot, it is a simple tweak that provided a huge improvement.

There is also another idea we wanted to try but have not had the time to do it. The idea consist of adding a BiDAF layer before the Context-Query layer to account for the relationship between the query words. We will be trying this next.

4 References

[1] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. **Squad: 100, 000+ questions for machine comprehension of text**. *CoRR, abs/1606.05250, 2016*.

[2] Pranav Rajpurkar, Robin Jia, and Percy Liang. **Know what you don't know: Unanswerable-questions for SQuAD.** *In Association for Computational Linguistics (ACL), 2018*.

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

[4] Caiming Xiong, Victor Zhong, and Richard Socher. **Dynamic coattention networks for question answering**. *arXiv preprint arXiv:1611.01604, 2016*.

[5] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. **Highway networks**. *arXiv preprint arXiv:1505.00387, 2015*.

[6] Natural Language Computing Group, Microsoft Research Asia, Furu Wei and Ming Zhou *R-NET: MACHINE READING COMPREHENSION WITH SELF-MATCHING NETWORKS*

5 Appendix

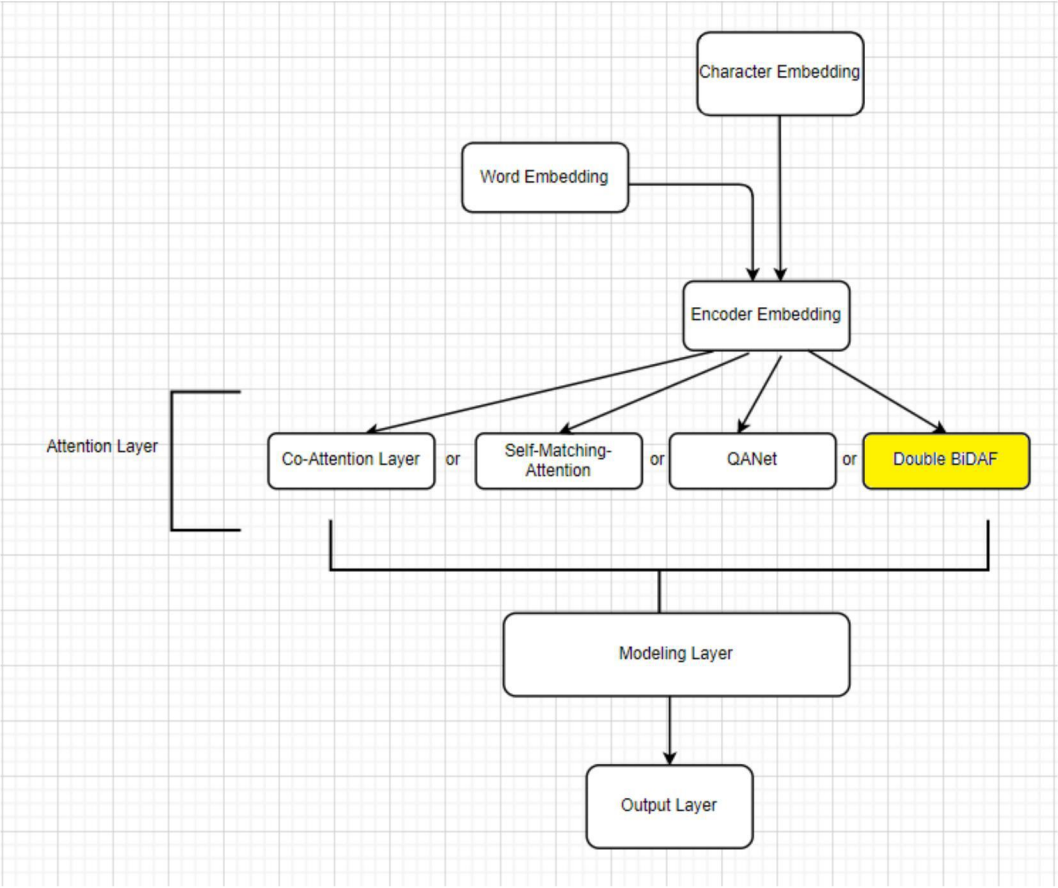

Figure 2: Diagram for the different layers in our QA model (Double BiDAF is our personal contribution)