Answer Validation for SQuAD 2.0

Ruben Contesti
Stanford University
contesti@stanford.edu

Demetrios Fassois
Stanford University
dimifass@stanford.edu

Kade Keith
Stanford University
kade@stanford.edu

Abstract

We explore approaches to extractive reading comprehension with unanswerable questions, as defined in the SQuAD 2.0 dataset. In particular, we explore answer verification, the task of determining whether a predicted answer is legitimate given the question, context paragraph, and answer. We attempt enhancing the baseline BiDAF with better loss function to handle unanswerable questions, as well as enhancing it with additional sub-layers and character embeddings. Beyond that, we build and evaluate a separate "Answer Verifier" component that is used to double check the model's predicted output as a post-processing step. We see improved performance both from our enhancements and from the new answer verifier. We then perform some basic analysis on where the two models (the question answerer and the answer verifier) make mistakes, and how they compliment each other. We finally made an attempt to build a simple verifier on top of BERT base lowercased, and report on how it does when combined with a BERT question answerer.

1 Introduction

Machine comprehension and question answering is a standard problem in NLP, with many deep learning models proposed in the recent years for solving it. State of the art systems developed on the golden standard SQuAD reading comprehension dataset have surpassed human performance. The recently introduced SQuAD 2.0 dataset adds another challenge, as roughly half of its questions don't contain an answer. New model architectures and modifications to existing question answering systems are needed in order to handle this additional complexity. In our work we explore different ways to tackle this task. Our methodology can be grouped in two areas. First, starting from a baseline BiDAF model introduced in Seo et Al (2016), we enhance it with additional sub-layers in order to increase model capacity. We also modified and extended its loss function in order to produce no-answer probabilities as described in Hu et Al (2018). For the second part we also took inspiration from the same body of work. We developed a separate verifier model that validates the answer, by deciding whether it is supported by the context or not. This verifier is then added to the pipeline as a post-processing step to the original model's output. Different extensions to the BiDAF baseline and different architectures for the verifier model are then tested and compared in order to pick the combination with the best performance. We find that the addition of an answer verifier component provides additional performance improvements, against our best end-to-end model. We also tried to build a verifier with BERT and use it in conjuction with it. For this purpose we used a second BERT model whose objective is to predict whether an answer is imposible or not. We averaged the probabilities of the two BERT models and if that probability is above 0.5 we output no answer. Unfortunately this basic verifier produced the same results as BERT.

2 Related work

Different approaches have been proposed to handle the case of unanswerable questions. In Clark et Al (2017) a special score that represents the probability of "no-answer" is calculated. The loss function then applies a softmax function over the "no-answer" score and the span scores from the pointers for answer extraction. Hu et Al (2018) contend that the shared normalization softmax layer confounds the "no-answer" and span scores. For this reason they introduce two independent losses, to concentrate on answer extraction and no-answer detection separately. Another argument the make is that previous systems are not trained to find candidate answers for unanswerable questions. For this reason they develop their reader model with the ability to extract plausible answers for every question, through the independent span loss. The last step is to apply the verifier model on the plausible answer to check whether it is legitimate and output the final prediction.

For extending the BiDAF model and increasing its learning capacity, we leveraged recent model implementations such as the OpenAI Transformer from Radford et Al (2018), and the QANet model introduced in Yu et Al (2018). These models rely on the mechanism of using self-attention as described in Vaswani et Al (2017).

3 Approach

The enhancements to the BiDAF baseline that were written by the project group are described here. We used the open source implementation of the transformer model from Vaswani et Al (2017) to implement a decoder stack. Modifications to the open source implementation included separating the decoder stack from the transformer model, modifying its inputs to match the BiDAF implementation, and creating a classification output for the verifier model. We also created a hidden layer output instead of the classification head, in order to extend the baseline BiDAF implementation with another module. An additional modification to the decoder stack included adding another sub-layer that uses depth-wise separable convolutions as described in Yu et At (2018). This augmented block was used in the verifier model as well as a hidden layer to extend the baseline BiDAF model.

A couple more enhancements to the baseline BiDAF were entirely our contributions. First we augmented the embedding layer to include character-level embeddings for every word as as described in Kim et Al (2015). We also implemented the three custom loss functions described in Hu et Al (2018).

For the answer verifier, a considerable amount of modifications for the data pre-processing pipeline was needed in order to extract plausible answers as the required inputs for the verifier models. Once these were done, we independently train the verifier based on the real answers and the plausible answers in the dataset. That verifier then learns to take question, context, and answer as input and output whether it is valid. We describe this in more detail in a later section.

To take advantage of the verifiers in an complete workflow, we take the predicted model output, feed it into the verifier, and get as output both the verifier's prediction, and it's confidence in that prediction. If the verifier predicts no-answer with high enough confidence, we change that output.

3.1 Character-level Encodings

In order to enhance the baseline BiDAF model, we computed character-level embeddings for every word as described in Kim et Al (2015). Characters are first embedded into vectors, which pass as 1D inputs to a CNN layer. The outputs of that layer are max-pooled over the entire length of the word in order to obtain a single size character embedding representation of each word. We then concatenate the character embedding with the pre-trained word embedding for every word to pass that through the highway network.

3.2 Auxiliary Losses

Following Clark et Al (2017), and adapting the last layers of the baseline output, we allow the model to predict a special no answer option. In particular we have the model compute another score, z, to represent the weight given to a "no-answer" possibility. The revised objective function becomes:

$$\mathscr{L}_{joint} - log\left(\frac{(1-\delta)e^{z} + \delta e^{s_a g_b}}{e^{z} + \sum_{i=1}^{n}\sum_{j=1}^{n} e^{s_i g_j}}\right)$$

where δ is 1 if an answer exists and 0 otherwise, s_j and g_j are the scores for the start and end bounds produced by the model for token j, and a and b are the correct bounds. If there are multiple answer spans we use the same objective, except the numerator includes the summation over all answer start and end tokens.

z is calculated by adding an extra layer to the model. A soft attention over the span start scores is computed, $p_i = \frac{e^{s_i}}{\sum_{j=1}^{n} e^{s_j}}$ and then use the hidden layers that were used to generate those scores, $\mathbf{h}_i$, giving $\mathbf{v1} = \sum_{i=1}^{n} \mathbf{h}_i$. Similarly, we obtain $\mathbf{v2}$ with the end scores. Finally a step of learned attention is performed on the output of the self-attention layers, computing $\mathbf{v3}$. $\mathbf{v1}$, $\mathbf{v2}$, $\mathbf{v3}$ are concatenated and used as an input in a two layer network with a ReLU activation that produces z as its only output.

With the previous loss function the system is not incentivized to produce answers for unanswerable questions but in the approach we are following, we intend to verify the "answerability" of the question at a later stage. In order to solve this issue Hu et al (2018) introduce two auxiliary losses.

The independent span loss:

$$\mathscr{L}_{indep-I} = -log\left(\frac{e^{\tilde{\alpha_{a}}\tilde{\beta_{b}}}}{\sum_{i=1}^{lp}\sum_{j=1}^{lp} e^{\tilde{\alpha_i}\tilde{\beta_j}}}\right)$$

where $\tilde{a}$ and $\tilde{b}$ are the augmented ground truth answer boundaries and $\tilde{alpha}$ and $\tilde{beta}$ are another pair of span scores calculated independently.

The independent no answer loss:

$$\mathscr{L}_{indep-II} = -(1-\delta)log\sigma(z) - \delta log(1-\sigma(z))$$

Finally the three losses are combined:

$$\mathscr{L} = \mathscr{L}_{joint} + \gamma\mathscr{L}_{indep-I} + \lambda\mathscr{L}_{indep-II}$$

Where both γ and λ are two hyper-parameters that control the weight of two auxiliary losses.

3.3 Answer Verifier and BiDAF Extensions

We explored a number of architectures for the answer verifier. First, we modify the baseline BiDAF model to exclusively predict whether a given answer is valid given the context, question, and predicted answer. This is roughly based on the Sequential Architecture Model in Hu et al (2018). We concatenate the context and question into a single string, and use that as one of inputs to BiDAF, with the answer span as the other input. In cases where the question is unanswerable, we use the "plausible_answers" given in the SQuAD data. Instead of using just start and end indices, we build arrays of the word embeddings of the answers so they are the same format as the context and question. We create a simplified output layer that takes the output of the modeling layer, selects the modeling layer output corresponding to the last token in combined context + question (m_i where i is the length of the context + the length of the question). We then apply a linear projection layer to predict and softmax to choose between our two classes, answer and no-answer.

$$\text{Softmax}(m_i W_p), m_i \in \mathbb{R}^H, W_p \in \mathbb{R}^H$$

We originally used the whole context, but found better performance and a much faster (2x) training if we just used the sentence from the context that contained the answer. Confirming that the entailment we are trying to capture happens at the sentence-level.

In an attempt to follow the work of Hu et al (2018) more closely, we next tried a multi-layer Transformer decoder, modified from the Generative Pre-trained Transformer (OpenAI GPT) (Radford et al. 2018). It's input was the answer sentence concatenated with the question, then a delimiter token, then the answer: $[S; Q; \$; A]$. The concatenated word and character-level embeddings are added to the positional embeddings. A decoder stack is then applied to encode the sequence and produce its hidden representation. The subsequent mask is used in the self-attention layer in order to prevent the the model from attending future inputs. The final hidden state from the encoded sequence is then

Table 1: Dev results

Model	DEV EM	DEV F1	AvNA
BiDAF + Auxiliary Losses	32.5	36.8	47.9
BiDAF Baseline	56.8	60.2	67.0
BiDAF + Transformer	57.3	60.8	67.4
BiDAF + QANet	57.8	61.1	67.5
BiDAF + Char Embeddings	58.6	62.0	68.4
BiDAF + QANet + Char Embeddings	58.6	61.9	**68.7**
BiDAF + QANet + Char Embeddings + Verifier	**59.4**	**62.1**	67.8

used as input to a projection layer and a softmax activation in order to produce the two-class answer or no-answer probabilities.

We also extended the Transformer decoder block to include an additional sub-layer that includes depthwise separable convolutions as proposed in Yu et al (2018).

We use a binary cross-entropy loss to optimize the verifiers:

$$loss = y \cdot log(\hat{y}) + (1 - y) \cdot log(1 - \hat{y})$$

Combined, this gives us a binary classifier that can be added as a post-processing step to an end-to-end question answering model.

4 Experiments

4.1 Data

We use the SQuAD 2.0 data subset provided by the starter code for training and evaluating our models. The train set contains 129,941 examples, the dev set 6078 examples, and the secret test set 5915. Additionally, sampled custom subset ranging from 500-2000 training examples were used to allow training on local machines and verifying that our models worked.

4.2 Evaluation method

We use the standard evaluation metrics of EM and F1 score for question answering, using the dev dataset. We also independently evaluate the answer verifier based on a simple accuracy metric of percentage of correct classifications of answers as legitimate or not. We also produce confusion matrices for the models and verifiers as a means to compare them.

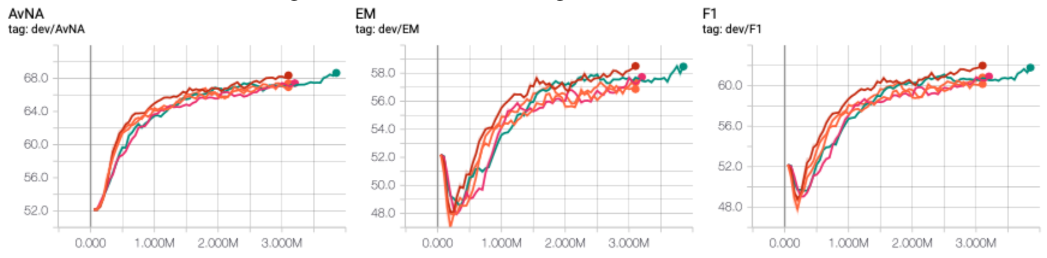
4.3 Experimental details

Full model training was done on Standard NV6 Azure VM. We ran the question-answerer models for 24-30 epochs, and the verifier models for 30. For the Transformer decoder model we used 8 number of heads throughout all the layers, and we used two decoder blocks in a stack for all models it was used in. The decoder stack layer for extending the BiDAF model was placed after the embedding level for both BiDAF models with and without character embeddings. The output of that layer is the same size as the hidden size (100). For the QANet layer the kernel size is 5, and the number of convolutional layers within a block is 4. The output size of this layer is also 100 and for extending the BiDAF model it was also placed after the embedding lyaer. For the BiDAF + Auxiliary Losses model, we set γ and λ both equal to 1. Finally, we set the threshold probability for no answer to 0.5; we will only output no answer in the case z_{prob} exceeds 0.5. For the combined model and answer verifier, we us a confidence threshold of 0.55. We did hyperparameter tuning on the dev set to find the best performing confidence threshold.

Table 2: Verifier dev results

Model	Dev Accuracy
Transformer Verifier	57.9 %
QANet Verifier	61.7 %
BiDAF Verifier	**62.8** %
BERT Verifier	79.7%

Figure 1: Results of training various QA models



4.4 Results

4.4.1 Question Answerer

See Table 1 and Figure 1.

4.4.2 Answer Verifier

See Table 2 and Figure 2.

4.4.3 Combined Model

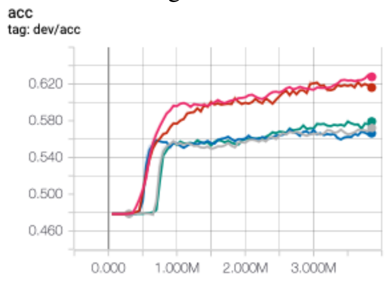
We choose our best non-PCE question-answering model, BiDAF + QANet + Char Embeddings, and our best non-PCE verifier, BiDAF. Their confusion matrices offer insight into how they differed.

Model confusion:

		Predicted	
		Answer	No-answer
Actual	Answer	37.5 %	22.8 %
	No-answer	10.4 %	29.3 %

Verifier confusion:

Figure 2: Results of training various answer verifier models



		Predicted	
		Answer	No-answer
Actual	Answer	40.8 %	30.2 %
	No-answer	07.1 %	21.9 %

Combined confusion:

		Predicted	
		Answer	No-answer
Actual	Answer	33.3 %	26.4 %
	No-answer	05.8 %	33.9 %

The combined model saw improvements and had F1: 62.1, EM: 59.4, AvNA: 67.8.

5 Analysis

5.0.1 Question Answerer

Our BiDAF baseline gave expected results. Adding the character embeddings also improved performance as expected, based on that model's position on the current SQuAD leaderboard. Adding Transformer and QANet layers offered small improvements, but it is difficult to know if that is because the model architecture was better, or if it is just because the overall model size increased.

We chose to combine BiDAF + QANet with character embeddings since they were our best non-PCE performers. Surprisingly, even though adding a QANet layer helped by itself, and adding character embeddings helped by itself, the combination of the two was only marginally better than just adding character embeddings. This confirms our theory that the gains from QANet were probably just from the increased model size.

The BiDAF with auxiliary losses performed poorly, with 20% less predictive power than the other models. The main reason is that the model failed to predict "no answers".

5.0.2 Answer Verifier

We were surprised that the BiDAF verifier performed the best. Even with increasing number of layers, the transformer verifier under-performed. We suspect that part of the reason why it didn't outperform the BiDAF verifier model was that we didn't pre-train the Transformer decoder as a language model. The QANet verifier improved on the Transformer decoder verifier as expected, but was slightly worse than BiDAF. Likewise pretraining it as a language model would have lead to gains in performance. Finally, we experimented with adding a version of the answer verifier in top of BERT. This model performed the best by far, which is to be expected given the power of BERT.

5.0.3 Combined Model

As the confusion matrices show, the improvement came from the answer verifier correctly flagging a number of the model's false positives as invalid. In the dev set specifically, we saw the verifier catch 271 bad answers and incorrectly remove 211 valid answers. Thus it did more good than harm.

The most interesting thing about the combination of the verifier and the question answerer is that even though it had worse AvNA accuracy than the model it was being combined with (62% vs 68%), it still improved the model's overall performance. Our intuition is that this must be because when the model outputs an invalid answer, that answer must be especially bad. It must be more obviously bad than even the "plausible answers" that the verifier had been trained to decipher. This makes sense when you look at the verifiers confusion matrix and see that it is much more likely to predict "answer" than "no answer". So when it sees something that it thinks is "no answer", it must be quite bad.

Regarding the combination of BERT baseline and BERT verifier, the verifier produced the same performance as BERT baseline. More detailed analysis would be necessary, but this confirms an initial suspicion we had that for the current state of the art (BERT), the verifier does not provide value.

6 Conclusion / Future work

In conclusion, sometimes you try a lot of stuff, and most of it doesn't work. We were pleased though to show a performance gain from combining the question answerer with a separate answer verifier, and in particular showing that they differ in what kinds of predictions (and errors) they make. This confirms our intuition that for non-PCE models at least, the two models are each learning different things, and have different strengths and weaknesses. This opens up a promising area of exploration: doing more combinations of different types of models and different types of verifiers to see if certain combinations compliment each other more. This could be paired with tuning the question answerers to be more aggressive about predicting an answer, because we know the verifier is there to catch bad predictions. And while our initial BERT exploration found that the verifier made little difference, maybe combining a BERT baseline question answerer with a verifier of a different architecture would help. One final positive comment about our system, is that practically speaking, training a verifier takes about half the time. The ability to tweak and re-train parts of the pipeline without having to completely retrain the whole system allows for quicker iterative development.

7 Additional information

Our submission of BiDAF + QANet + Character Embeddings + Verifier is "KDR_Model+Verifier" on the Non-PCE Dev Leaderboard, and " KDR+Model+Verifier" on the Non-PCE Test Leaderboard.

References

[1] Kim, Y. & Jernite, Y. & Sontag, D. & Rush, A. M. (2015) Character-Aware Neural Language Models. arXiv:1508.06615

[2] Clark, C. & Gardner, M. (2017) Simple and Effective Multi-Paragraph Reading Comprehension. arXiv:1710.10723

[3] Minghao Hu, Furu Wei, Yuxing Peng, Zhen Huang, Nan Yang, Dongsheng Li (2018) Read + Verify: Machine Reading Comprehension with Unanswerable Questions. arXiv:1808.05759

[4] Vaswani, A & Shazeer, N & Parmar, N & Uszkoreit, J & Jones, L & Gomez, A.N. & Kaiser, L & Polosukhin, I (2017) Attention Is All You Need. arXiv:1706.03762

[5] A. W. Yu, D. Dohan, M.-T. Luong, R. Zhao, K. Chen, M. Norouzi, and Q. V. Le. (2018) Qanet: Combining local convolution with global self-attention for reading compre- hension. In ICLR

[6] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. (2018) Improving language understanding by generative pre-training. [Online]. Available: https://blog.openai.com/language-unsupervised/