Reading Comprehension for SQuAD 2.0 using U-Net

Team:
Qingyun Wan
`qywan@stanford.edu`

Abstract

In this paper, I implement and exlore a primitive version of an end-to-end model based on U-Net Sun et al. (2018) designed for SQUAD 2.0 with certain modifications and evaluate its effectiveness compared to the baseline model Seo et al. (2016) through evolving it gradually. Eventually after experiments of different configurations, this single model achieves 65.418% EM, 69.488% F1 on the dev set and 62.925% EM, 66.931% F1 on the test set and the ensemble model achieves 66.258% EM, 69.815% F1 on the dev set and 63.838% EM, 67.372% F1 on the test set as shown on the leaderboard.

1 Introduction

Machine Reading Comprehension and Question Answering are challenging tasks in natural language processing as they require a system to understand a given context (or passage) and parse various kinds of questions to select the correct text span from the context as their answers. SQuAD-the Stanford Question Answering Dataset is designed for evaluating such reading comprehension systems by extracting contexts and building questions from Wikipedia and so far, neural methods have achieved near-human performance on this dataset. However, such dataset suffers from allowing random guesses out of the context when the questions are virtually unanswerable. SQuAD 2.0 Rajpurkar et al. (2018) is therefore constructed to overcome this weakness by including unanswerable questions so the systems that predict plausible answers for unanwerable questions will be punished.

Rather than using Pre-trained Contextual Embeddings like Bert Devlin et al. (2018) which have achieved leading performance all over the SQuAD 2.0's leaderboard, this project is focused on implementing a non-PCE method that improvees the baseline which is a BiDAF-based model and explore more on how different parts in this method contribute to the overall performance.

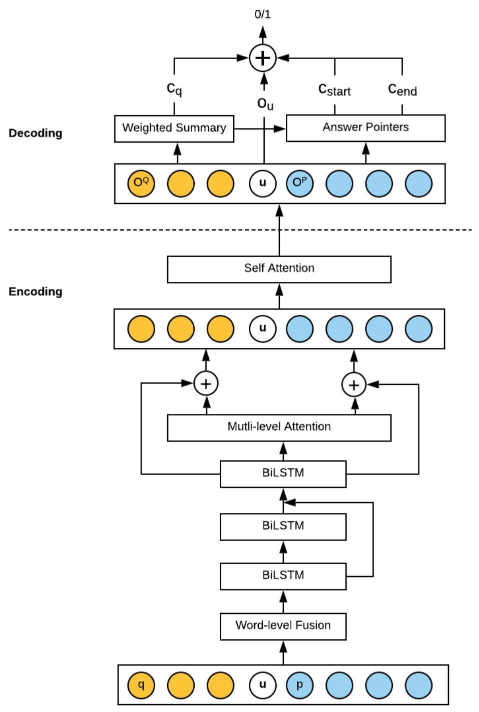
2 Related Work

The model implemented in this paper is from U-Net Sun et al. (2018) which is an end-to-end model designed specifically for SQuAD 2.0 as it trains answer pointers to obtain answer start and end positions and answer verifier to determine whether a question has an answer. The reason to choose this method is that first it has a competitive performance in the leaderboard among non-PCE models as its single model obtains 69.262% EM and 72.642% F1, second unlike other models that have a pipeline structure for answer verifier, this approach learns in a multi-task fashion that anwers verifier is portable in a unified model and third it introduces a universal node to capture information from both question and context to improve the accuracy of answer verifier.

3 Approach

The U-Net model follows the typical structure that has a encoding layer and a decoding layer. Though not clearly pointed out in the original paper, its encoding layer is following the structure of FusionNet

Huang et al. (2017) after carefully comparing them. The answer pointer and answer verifier lives in the deconding layer. The universal node is involved in both encoding and decoding layers. So the overall architecture is as below:



3.1 Encoding Layer

3.1.1 Unified Encoding

Like normal models, the question and the passage are represented with GloVe Embedding and additional features as $Q = q_{i=1}^{m}$ and $P = p_{i=1}^{n}$. In addition to these representations, a universal node **u** initialized randomly is added into the whole representation which is:

$$V = [q_1, q_2, \ldots, q_m, \mathbf{u}, p_1, p_2, \ldots, p_n]$$

Before composing V, a word-level fusion is applied to inform the passage about the words in the question like:

$$\hat{g_i}^{P} = \sum_j \alpha_{ij} g_j^Q, \alpha_{ij} \propto \exp(S(g_i^P, g_j^Q)), S(x,y) = ReLU(Wx)^T Relu(Wy)$$

where g represents the GloVe word embedding.

V is then sent to two-layer bidirectional LSTM to obtain low-level and high-level semantic information both of which are then forwarded to another bidirectional to construct the final representation as below:

$$H_{low-level} = BiLSTM(V), H_{high-level} = BiLSTM(H_{low-level})$$

$$H_{understanding} = [H_{low-level}; H_{high-level}; BiLSTM([H_{low-level}; H_{high-level}])]$$

which is for reading and question understanding. BiLSTM stands for bidirectional LSTM here.

3.1.2 Three-level Attention

Then the model uses bidirectional attention between the question and passage on these three levels: low-level, high-level and understanding level. On each level, the hidden representations of the question and passage are both attached with the hidden representation of the universal node so that this node carries both question and passage information. The goal for this three-level fusion is to capture different levels of information.

According to the FusionNet, first, history-of-word information is built as

$$HoW = [g; H_{low-level}, H_{high-level}]$$

where g represents the GloVe word embedding. Then the low-level, high-level and understanding level fusion are computed following the same formula:

$$\hat{H}_i^P = \sum_j \alpha_{ij} H_j^Q, \alpha_{ij} \propto \exp(S(HoW_i^P, HoW_j^Q))$$

$$S = ReLU(U(HoW_i^P))^T D ReLU(U(HoW_i^Q))$$

The output of this block is $\hat{H}_{low-level}$, $\hat{H}_{high-level}$ and $\hat{H}_{understanding}$ after applying the aforementioned bidirectional multiplicative attention where different attention weights S are learned for different levels. During this phase, the universal node representation is attached to both the encoded passage and question representations. Finally, to fully fuse the information, we have

$$H^A = BiLSTM([H_{low-level}; H_{high-level}; H_{understanding}; \hat{H}_{low-level}; \hat{H}_{high-level}; \hat{H}_{understanding}])$$

3.1.3 Final Fusion

This phase continues fusing the information by concatenating the original embedded representation to follow the DenseNet convention:

$$A = [V; H^A]$$

And the final fused information is

$$O = BiLSTM[H^A; \text{self-attention}(A)]$$

which is splited into O^Q and O^P to represent the question and the passage respectively. Specifically, the universal node is only attached to O^P as it is pointed when there is no answer.

3.2 Decoding Layer

By summarizing the fused information O^P and O^Q, this layer tackles the aforementioned three sub-tasks which are trained with the total loss function

$$L = L_{\text{answer pointer}} + L_{\text{plausible answer pointer}} + L_{\text{answer verifier}}$$

, which has three parts corresponding to the three sub-tasks:

- $L_{\text{answer pointer}} = -\log(p_{start}) - \log(p_{end})$ where p_{start} and p_{end} indicate the probablity of the ground-truth start and end of the answer span.
- $L_{\text{plausible answer pointer}} = -\log(p'_{start}) - \log(p'_{end})$ where p'_{start} and p'_{end} indicate the probablity of the ground-truth start and end of the plausible answer span.
- $L_{\text{answer verifier}} = -(\delta \log p^c + (1-\delta)\log(1-p^c))$ where δ is 1 if the question has an answer otherwise 0. p^c is the probablity of answerability of the question which is summarized by fixed-dim representations generated from O^P and O^Q.

p_{start} and p_{end} are computed by a fixed-dimensional representation c_q of question information:

$$c_q = \sum_i \frac{\exp(W^T) o_i^Q}{\sum_j \exp(W^T) o_i^Q} o_i^Q$$

$$p_{start_i} \propto \exp(c_q W_{start} o_i^P)$$

$$p_{end_i} \propto \exp(c_q W_{end} o_i^P)$$

p^c is computed from a fixed vector F representing question information, context information and universal information o_u as below:

$$F = [c_q; o_u; c_{start}; c_{end}], c_{start} = \sum_i p_{start} o_i^P, c_{end} = \sum_i p_{end} o_i^P$$

3.3 Modification

In the original U-net model, the answerability p^c of a question is computed using:

$$p = \sigma(W_f F)$$

However during training, I observed that the probablities of has answer and no answer were always around certain values and the no-answer probablity was always larger, leading the model to predict no answer all the time. To address this issue, a bias term is added to balance the probablities of has answer and no answer as below during the training:

$$p = \sigma(W_f F + b)$$

The original final loss function is:

$$L = \delta L_{\text{answer pointer}} + (1 - \delta) L_{\text{no answer pointer}} + L_{\text{answer verifier}}$$

where δ is 1 if the question has an answer otherwise 0 and $L_{\text{no answer pointer}}$ includes plausible answer loss stated in the original paper, leaving no opportunity to train on plausible answers to predict answer boundary. Therefore, the final loss function is modified to:

$$L = L_{\text{answer pointer}} + L_{\text{plausible answer pointer}} + L_{\text{answer verifier}}$$

so that it trains plausible answer pointer and answer pointer simultaneously. During evaluation, if answer verifier indicates a question has an answer, use either answer pointer or plausible answer pointer to determine the answer boundary depending on model configurations.

4 Experiment

4.1 Data and Evaluation Method

The training and evaluation set in the experiments is same as used the baseline model. As mentioned in the U-Net paper, their experiment ignores passages with more than 400 words and questions with more than 50 words and leverages Elmo which is pre-trained contextual embeddings, which implies that our experiments will absolutely achieve lower performance than their reported scores with the same hyperparameters. All the experiments are evaluated against EM and F1 scores.

4.2 Hyperparameters

To match with the original paper, the hidden layer dimension is set as 125 and attention layer dimension is set as 250. The dimension is 12 for embed POS tags and 8 for NER tags. The dimension of word embedding generated from character embedding is 256. Over all the modeling layers, there is a dropout layer with a dropout rate of 0.3.

4.3 Experiments & Results

The baseline model is based on Seo et al. (2016) without a character-level embedding layer. With the modified U-Net model, there are four major experiments under different configurations shown in table 1 carried out. The first experiment and the second one only involves GloVe word embeddings as input, while the second one has a plausible answer pointer that leverage the plausible answers of no answer questions. The third experiment has the same answer pointers as the first two, but involves more features including NER, POS tags and 3 binary features: exact match, lower-case and lemma match mentioned in the U-Net paper. The forth experiment contains all the configuration of the first three but introduces CNN-based character embeddings to augment the GloVe embeddings. The configurations of each model are listed in Table 1.

The results are demonstrated in Table 2 for all the models listed in Table 1. We can see that compared to the baseline model, there is a small lift of 2% and 1.1% in F1 and EM. But when plausible answers are introduced, the model gains more training power and its performance is further improved by 1.8% and 2.2% in F1 and EM. Since both results are not comparable to the result of full-blown U-Net in the original paper, I included additional features mentioned in the paper excluding Elmo embeddings

Table 1: Model Configurations

Model	Glove	Elmo	Plausible Answer	POS & NER tags	Binary Features	Character Embeddings
Baseline	✓					
Modified U-Net 1	✓					
Modified U-Net 2	✓		✓			
Modified U-Net 3	✓		✓	✓	✓	
Modified U-Net 4	✓		✓	✓	✓	✓
U-Net reported in the papaer	✓	✓	✓	✓	✓	

Table 2: Results on the dev set

Model	EM in %	F1 in %
Baseline	56.38	59.75
Modified U-Net 1	**57.42**	**61.76**
Modified U-Net 2	**59.60**	**63.52**
Modified U-Net 3	**64.51**	**68.79**
Modified U-Net 4	**65.41**	**69.48**
Modified U-Net 4 ensembled with baseline	**66.25**	**69.81**
U-Net reported in the paper	70.3	74

and it turns out feature engineering to represent sentences and passages plays a significant role in model precision as F1 and EM are further boosted up by 5.2% and 4.9% in F1 and EM. Finally, by introducing character embedding to better represent OOV words, the performance continues to improve by 0.7% and 0.9% in F1 and EM. Finally, by ensembling with the baseline model when the weights of U-Net versus baseline is 3:1, the performance is boosted by 0.840% in EM, 0.327% in F1. The details of ensembling will be described in the following section.

5 Analysis

5.1 Ablation Study

5.1.1 Feature Ablation

Most of the feature ablation study is already discussed in the previous section where we compare the same models with different additional features. It turns out the POS, NER tags and 3 binary features empowers the model most. The original U-Net paper also uses tf-idf feature but with my experiment, it couldn't boost the performance of the final model with additional features and character embeddings. By introducing character embeddings, the model is enhanced in predicting dollar numbers for example:

Context: ...or the 2012Ž01313 school year annual tuition was $ 38,000, with a total cost of attendance of $ 57,000. Beginning 2007, families with incomes below $ 60,000 pay nothing for their children to attend, including room and board. Families with incomes between $ 60,000 to $ 80,000 pay only a few thousand dollars per year, and families earning between $ 120,000 and $ 180,000 pay no more than 10% of their annual incomes...
Question: What is the total cost of attendance in 2012-13
Answer:"$ 57,000"
U-Net model w/ character embedding prediction:"$ 57,000"
U-Net model w/o character embedding prediction: ""

5.1.2 Model Ablation

In addition to the ablation study in the original paper, I carried more experiments of the model structure for ablation purpose shown in Table 3.

First I removed the higher-level and the understanding-level representations in the multi-level fusion and the result is a non-trivial drop in both EM (-2.22%) and F1 (-2.2%). which indicates higher-level fusion is helpful to further capture the information from the question and the passage.

Then in the self-attention phase of the model, I replaced with multi-head self-attention using the module in https://github.com/jadore801120/attention-is-all-you-need-pytorch but it didn't benefit the performance.

The original paper pointed out that it was worth allocating different weights to the losses of answer verifier and answer pointers in the final function. It turns out that it is best to stay the same and with more weight on the answer pointers, the performance is more deteriorated compared with more weight on the answer verifier.

Table 3: Model Ablation Study

Model Change	EM change in %	F1 change in %
No high-level hidden states	-2.22	-2.2
Use multi-head self-attention	-0.58	-0.66
More weight on answer pointer's loss	-1.95	-0.76
More weight on answer verifier's loss	-0.28	-0.14

5.2 Error Analysis

By comparing the incorrect predictions of U-Net when the baseline can predictly correctly, I found in many cases, U-Net is unable to narrow down the span accurately by simply returning a verbose span of passage as the prediction which contains the actual answer, below is an example:

Context: Closely related fields in theoretical computer science are analysis of algorithms and computability theory. A key distinction between analysis of algorithms and computational complexity theory is that the former is devoted to analyzing the amount of resources needed by a particular algorithm to solve a problem, whereas the latter asks a more general question about all possible algorithms that could be used to solve the same problem...
Question: What field of computer science analyzes all possible algorithms in aggregate to determine the resource requirements needed to solve to a given problem
Answer:"computational complexity theory"
Baseline model prediction:"computational complexity theory"
U-Net model prediction: "theoretical computer science are analysis of algorithms and computability theory. A key distinction between analysis of algorithms and computational complexity theory"

On the contrary, the baseline model often predicts correctly in most of these cases so that their predictions are complementary in most time.

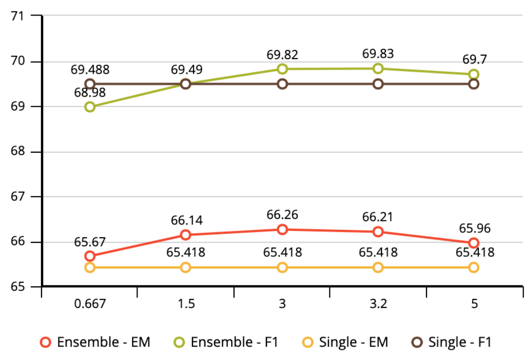
5.2.1 Ensembling

Based on the above error, by ensembling the U-Net and the baseline model, we can potentially leverage the baseline model's ability to precisely locate the answer span in the context. To value the unanswerability prediction of U-Net, the ensembling is only applied the cases when the answer verifier predicts the question has an answer, and the probability distributions of the start and end positions of the answer are determined as below:

$$p_{start} = \sum_{i=1}^{N} w_i p_{start_i}, p_{end} = \sum_{i=1}^{N} w_i p_{end_i},$$

for N models where w_i is weight for ith model. I carried out experiments imposing different weights on two models and the results on the dev set are shown as below:

The x-axis represents the ratio of weights of U-Net and baseline and the figure shows the best result is at 3:1.It turns out that even ensembled with a very primitive BiDAF model, we can achieve +0.913% EM, +0.441% F1 on the test set and +0.840% EM, +0.327% F1 on the dev set if more weights are put to U-Net model, which implies by ensembling with BiDAF, the location of answer span is located



more precisely. There is more room for performance boost if we ensemble with an advanced version BiDAF model.

6 Conclusion

In this paper, by thoroughly experimenting the U-Net model and comparing with the baseline model, we conclude that the U-Net model achieves small performance gain compared to the baseline without additional features which further largely improves this model. The U-Net model has some fundamental problems in narrowing down the correct answer span compared to the baseline so if we can improve the model structure (basically FusionNet) of U-Net learning from the BiDAF, it might perform much better.

References

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Huang, H.-Y., Zhu, C., Shen, Y., and Chen, W. (2017). Fusionnet: Fusing via fully-aware attention with application to machine comprehension. *arXiv preprint arXiv:1711.07341*.

Rajpurkar, P., Jia, R., and Liang, P. (2018). Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*.

Seo, M., Kembhavi, A., Farhadi, A., and Hajishirzi, H. (2016). Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*.

Sun, F., Li, L., Qiu, X., and Liu, Y. (2018). U-net: Machine reading comprehension with unanswerable questions. *arXiv preprint arXiv:1810.06638*.