Evaluating Different Techniques on SQuAD

Zhangyuan Wang, Xinyu Xu

March 22, 2018

Abstract

SQuAD is a natural language question-answer dataset, which requires NLP algorithms to not only understand each word, but also model the relationship between words in sentences. In this project, we implement several standard techniques, including Character CNN, BiDAF and Co-Attention, and apply them on SQuAD dataset. Through careful experiments and comparison study, we find the best hyperparameters and model given the computation resources. In-depth discussion and visualization is carried out to gain a better understanding in the model we implemented. The best test result reaches F1=0.706.

1 Introduction

Question answer of machine comprehension is an important task of natural language processing. Given a paragraph and a question, the machine should únderstandt́he context and give the right answer. Some large labeled question-answering datasets have been proposed recently, which can be trained by some neural network methods so that the machine is able to give the right answer to the question. Stanford Question Answering Dataset (SQuAD) is a significant reading comprehension system which is consisting of around 100k questing asking and answering datasets of wikipedia articles. The answer of each question is in the given paragraph and it is suitable for deep neural network learning. Samples and description of the dataset are presented in fig. 4.

In this paper, we aim to explore different models to learn SQuAD datasets and compare different hyperparameters while implement these models. We use different attention mechanisms including character-level CNN for additional embedding information, bidirectional attention and co-attention at the attention layer to improve the performance. Also, other techniques including more layers of RNN encoder, regularization and dropout are evaluated as well..

2 Models

A block diagram of our model is shown in fig. 1. Each input sample is composed of context and question. The two corpus are fed into character CNN to obtain word features from character level embedding, which is then concatenated with pretrained Glove vector to form the embedding of the sentence. Then the two embedding vectors are encoded by two RNN Encoder which may or not not share weight. Given the encoded sentences, attention layer (either BiDAF or Co-Attention) takes in the two features and with another fully connected layer and softmax layer, produces output start and end probability over context. Each individual module will be addressed in the following subsections.

2.1 Character embedding layer

The algorithm for character embedding CNN is shown in fig. 2. Similar to the word embedding layer, we embedded the character of context and question using an alphabet list separately. So that we can get character-level vectors e_1, e_L. Each word in input sentence is concatenated from embedding vectors of each characters and padded to word_len, the maximum number of character in word. The convolutional filter is $w \in \mathbb{R}^{k*dc,1}$, which will slide across the matrix and strides across different character embeddings. A feature map was calculated and we were able to get the max c which is the one with the most important feature. Then, we apply ReLU for non-linearity. Finally, we concatenated it

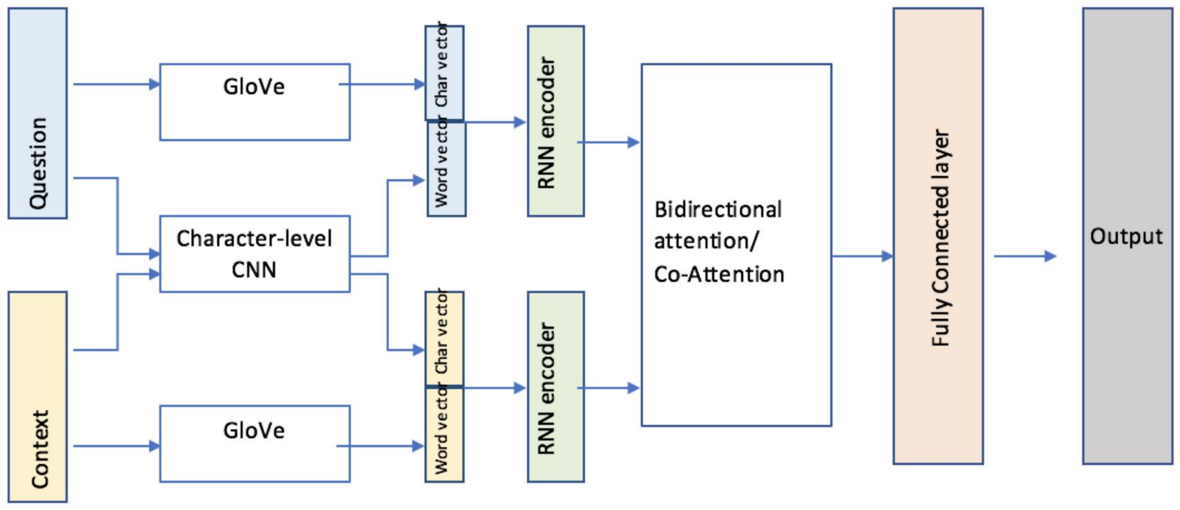


Figure 1: Model of this project.

with the embedded context and question word as the input of the next layer. This let our model has better F1 score by 3 4 percent.

2.2 RNN Encoder

In the baseline model, a bi-directional GRU is used. In our model, we compare the long-short term memory cell as Bidirectional RNN (BiLSTM). We also carried out experiment using stacked-BiLSTM, i.e. the output state from the first BiLSTM is used as input for the second layer of BiLSTM. Stacking more layers is also possible.

2.3 Attention layer

2.3.1 Bidirectional Attention Flow

Bidirectional Attention Flow (BiDAF) perform on both context to question and question to context directions, so that the attention can flow both ways. It categorizes the importance of the attention to each word vector and employs a similarity matrix based on the context vector and the question vector and the combination of both context and question vectors.

2.3.2 Co-attention layer

Similar to BiDAF, Co-Attention also has flow between both directions. After a non-linear tanh function and concatenating a sentinel vector, an affinity matrix is first calculated. Then C2Q and Q2C attention distributions are computed, and taken weighted sum of Q2C attention output as second-level attention. Afterwards, another biLSTM model is built on the concatenation of second-level attention and weighted C2Q attention output, which enables the model to give more accurate attention behavior.

2.4 Output layer

The output from attention layer is fed into the fully connected layers (either one or two FC layers, with ReLU activation and dropout). Then two separate softmax are applied to obtain the final distribution over start and end distribution.

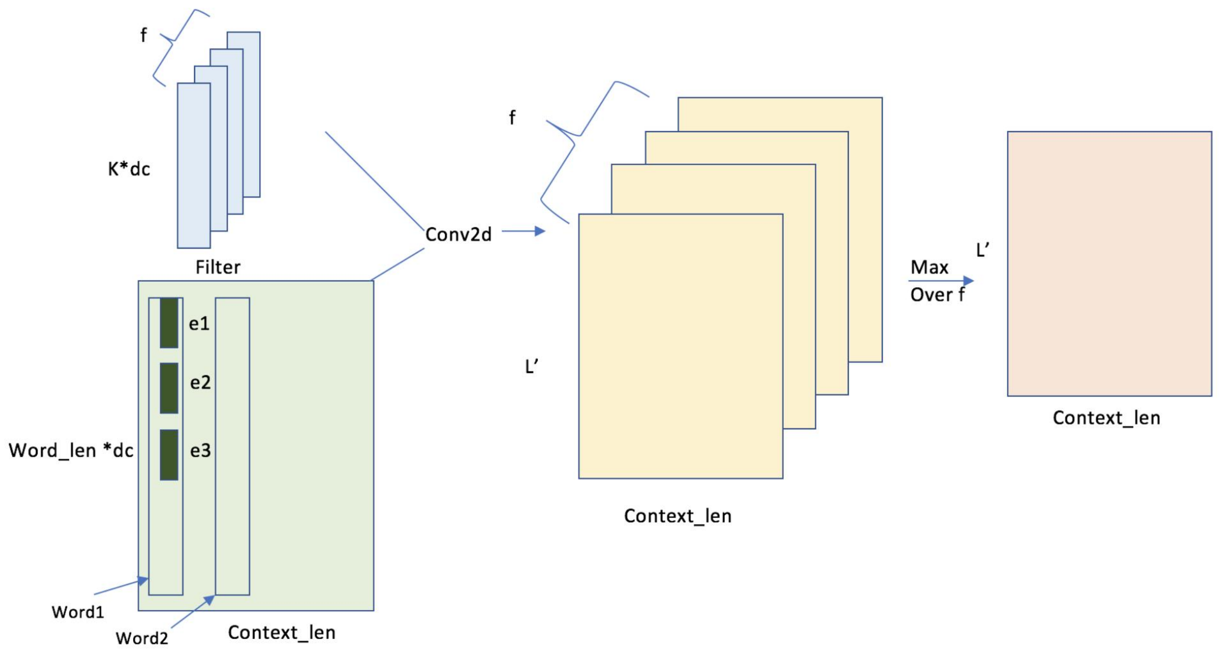


Figure 2: Algorithm of character CNN with conv2d. word_len refers to the length of characters in each word. dc is the dimension of character embedding. K and f specifies windows size and number of filters for the convolution kernels.

2.4.1 Smart Span

To obtain the predicted start and end position, one naive implementation takes maximum of start and end distribution. However, since start position must be present before end position, we can adopt a s̈mart spans̈trategy introduced in DrQA paper Danqi, et al(2017). The end position (p_end) is related with the start position(p_start) such that $p\_start <= p\_end <= p\_start + 15$.

2.4.2 Weighted loss

In the experiment, we found that if the start position is wrong, the whole prediction will be wrong with large probability. Therefore, we tried to assign different weight to the start and end position when calculating the loss. Then the loss becomes $\frac{4}{3}loss_{start} + \frac{2}{3}loss_{end}$.

3 Experiments and Result

3.1 Experiment Setup

In this experiment, we run our models on Azure, which is a cloud platform supported by Microsoft. Regarding to the optimization algorithms, our loss is calculated based on the Adam optimizer. The learning rate of this experiment is 0.001 as default. For context length, after we plotted the context length histogram , we found that most of the context length range from 0 to 300. Therefore, we always use context length value 300 as our input flag.

3.2 Evaluation Metric

We used two metrics of evaluations provided by SQuAD - EM and F1 scores. EM stands for exact match and it is a binary measure which is the percent of the machine answers the question exactly the same as the given answer. F1 score allows some variances between the predicted answer and the real answer, it evaluate the harmonic mean of precision and recall. Since it is very hard for machine to answer a question totally correct, the combination of both EM and F1 score will give us a better understanding of how our model performs.

3.3 Hyperparameter details

- **Different RNN:** We have tried Gated Recurrent Unit(GRU) and Long-short Term Memory(LSTM) at the encoder layer. After some experiments on both co-attention and bidirectional attention model, we found that LSTM outperform GRU by the average of 2%. The result is shown in figure 3(a).
- **Separate Encoder:** Whether separate the RNN encoder of context and question or not won't change the performance very much. The result is shown in figure 3(a).
- **Different number of layers of RNN** Figure 3(b) shows there is no obvious difference between 1 and 2 layers.
- **Dropout rate:** Compared with the default dropout value of 0.15, we found dropout rate equaled 0.30 was able to improve our model by 1%. The result is shown in figure 3(c)
- **Type of regularization**: We compared L1 and L2 regularization and found that L2 regularization outperform L1 by around 5% as shown in figure 3(d).
- **Embedding size:** We tried the embedding size of 100, 200 and 300 and found that as the size increases, the F1 and EM score would increase.
- **Hidden size:** By comparing the different hidden size of 100, 400, 500, 600 and 1200, we found that as the hidden size gets bigger, we can get better performance on the training set. However, when the hidden size became larger than 600, the model would suffer overfitting.

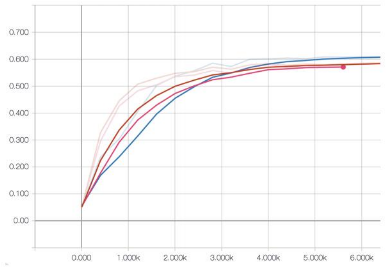
3.4 Final Results

With the Co-attention and character CNN (Coattn+charCNN) model, we have achieved F1 score 70.567 and EM score 59.027 on the 224n test leader board, with Coattn model[1] . Table 1 summarizes the local dev score of different models. Compared with other similar work on Bidaf, our bidaf model has lower score. But for co-attention, we achieved reasonable results as discussed in the previous paper.

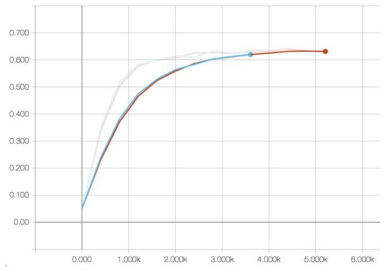
	F1	EM
Baseline	39.87	28.57
Bidaf	44.48	29.55
Bidaf + charCNN	49.90	35.32
Coattn	65.14	48.60
Coattn + charCNN	66.61	51.14

Table 1: Result on **local dev** of different models

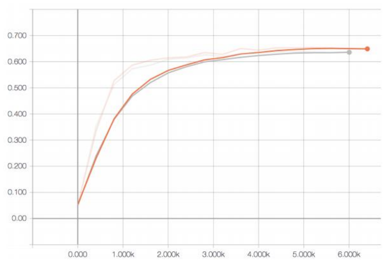
[1]The new best from Coattn+charCNN is not submitted to dev board, due to limited time



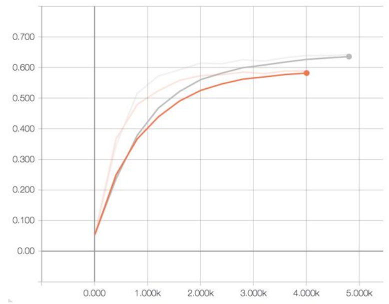
(a) F1 score of LSTM (blue), GRU (pink), GRU with separate encoder (red)



(b) F1 score of One layer (blue), Two layers of RNN (pink)



(c) F1 score of Dropout = 0.15 (gray), Dropout=0.3 (orange)



(d) F1 score of score of L1 orange, L2 Regularization (gray)

Figure 3: Dev/F1 of Co-attention using different values of hyperparameters.

4 Discussion

4.1 Evaluation on hyper-parameter selection

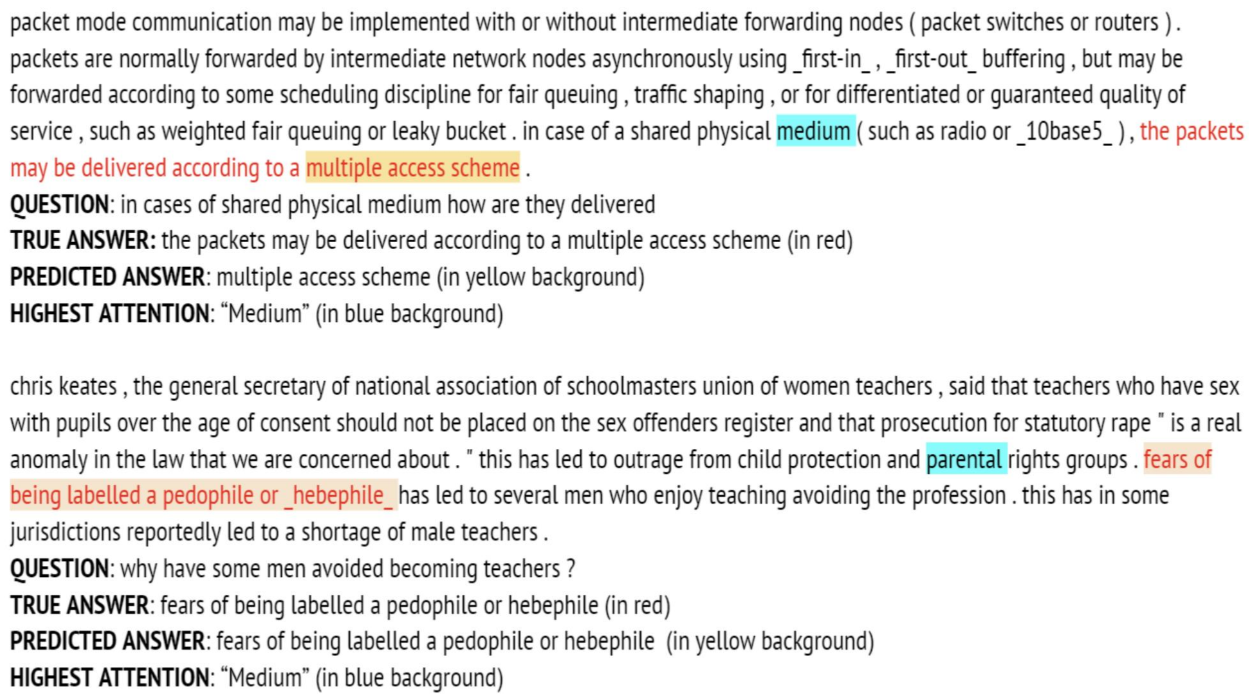
The results from previous section match well with our intuition on choices of hyper-parameters. For example, higher embedding size will bring more information into the model and can result in 1-3% improvement. Switching to LSTM cell from GRU cell enables the model to better express the temporal relationship. On the other hand, Model complexity is a hard trade-off between high expressiveness and overfitting. Choosing a proper hidden size is obviously important. Yet it is harder to tell whether we should have two separate RNN encoder/character CNN for context and question, or whether we should use stack-biLSTM. Results show minor improvement, if any, with these more complex models. The reason could be that context and questions share similar syntax and semantic structures, which can be better represented by one encoder and one character CNN. Following the rule of Occam's Razor, we always choose the simplest model wherever the difference is small.

Regularizing the model is an art. In our project, we have tried to use l_1 and l_2 regularization with different parameters, as well as dropout rate. Theoretically, l_1 is more robust and will encourage sparsity, while l_2 is more stable for smaller weights. In practice, weight regularization has limited effect: values larger than 1e-3 will most likely stops the training F1 at around 0.6. By comparison, l_2 performs better, which combined with dropout=0.3, gives the best result.

4.2 Evaluation of attention layer

To visualize the behavior of attention layer, we analyze where the highest attention score occurs in co-attention layer. Formally, let $a_i \in R^l$ denote attention-weighted question vectors over the ith context word (same as the handout). Take $argmax_{i\in[1,context\_len]} max_{j\in[1,l]} a_i j$ to be the position where a_i has the largest response. Ideally this should correspond to the position of the answer span, since the answer start/end word should take the largest effect over the model. The following two examples show what our model attends to (fig. 4).

The first example shows correct attention, since the question was asking for "shared physical medium" and the attention was pointed to "medium". However, the answer is only partially correct. From a human point of view, the predicted answer is also reasonable, since it comes after "delivered according to", which usually indicates the answer. The second example shows exactly matching answer. The attention spot is not as explainable as the previous one. However, since we are only visualizing the first-level attention, this might be an output from a left-to-right read, and in this sense the attention on the leading word "parental" is reasonable.


packet mode communication may be implemented with or without intermediate forwarding nodes (packet switches or routers) . packets are normally forwarded by intermediate network nodes asynchronously using _first-in_ , _first-out_ buffering , but may be forwarded according to some scheduling discipline for fair queuing , traffic shaping , or for differentiated or guaranteed quality of service , such as weighted fair queuing or leaky bucket . in case of a shared physical medium (such as radio or _10base5_) , the packets may be delivered according to a multiple access scheme .
QUESTION: in cases of shared physical medium how are they delivered
TRUE ANSWER: the packets may be delivered according to a multiple access scheme (in red)
PREDICTED ANSWER: multiple access scheme (in yellow background)
HIGHEST ATTENTION: "Medium" (in blue background)

chris keates , the general secretary of national association of schoolmasters union of women teachers , said that teachers who have sex with pupils over the age of consent should not be placed on the sex offenders register and that prosecution for statutory rape " is a real anomaly in the law that we are concerned about . " this has led to outrage from child protection and parental rights groups . fears of being labelled a pedophile or _hebephile_ has led to several men who enjoy teaching avoiding the profession . this has in some jurisdictions reportedly led to a shortage of male teachers .
QUESTION: why have some men avoided becoming teachers ?
TRUE ANSWER: fears of being labelled a pedophile or hebephile (in red)
PREDICTED ANSWER: fears of being labelled a pedophile or hebephile (in yellow background)
HIGHEST ATTENTION: "Medium" (in blue background)

Figure 4: Two examples with highlight on context to show the true answer, predicted answer, and position of highest attention score.

4.3 Visualization of character embedding

In fig. 5, a T-SNE visualization is shown about the weight of character embedding. It can be observed that letters (a,b,...,z) are clustered in one group, and numbers (0,1,2,...,9) are clustered in another group. However, the pattern is not as regular as we would expect, having a lot of other symbols mixing together. We might be able to learn better character embeddings with some other techniques, such as stacked character CNN.

5 Conclusion

In this project we implement Character CNN, BiDAF, Co-Attention as well as other useful functions for improving the F1/EM score on SQuAD dataset. By thorough quantitative comparison study and visualization of the model, we provide reasons for each design choice. It is shown that proper model complexity and regularization is import. As future work, we could try to use bidaf and coattention together, or take ensemble over multiple models.

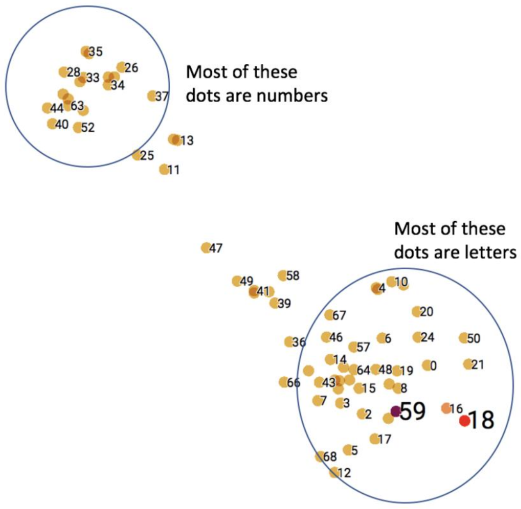


Figure 5: T-SNE visualization of embedding weights of character. Number shows index in the alphabet, where 0-25 index are corresponding to letters, and 26-35 correspond to numbers. Other index contains a variety of symbols.

Acknowledgments

We would like to thank the CS224N staff for all the efforts they made. Without your great starter code and detailed handout, it would be impossible for us to make such progress. Thank you for the fantastic semester.

References

[1] Alexander, J.A. & Mozer, M.C. (1995) Template-based algorithms for connectionist rule extraction. In G. Tesauro, D. S. Touretzky and T.K. Leen (eds.), *Advances in Neural Information Processing Systems 7*, pp. 609-616. Cambridge, MA: MIT Press.

[2] Bower, J.M. & Beeman, D. (1995) *The Book of GENESIS: Exploring Realistic Neural Models with the GEneral NEural SImulation System.* New York: TELOS/Springer-Verlag.

[3] Hasselmo, M.E., Schnell, E. & Barkai, E. (1995) Dynamics of learning and recall at excitatory recurrent synapses and cholinergic modulation in rat hippocampal region CA3. *Journal of Neuroscience* **15**(7):5249-5262.