Building a QA System (IID SQuAD track)

Stanford CS224N Default Project

Bo Yu
boyu19@stanford.edu

Abstract

The task of this project is to build a question answering system that works well on SQuAD 2.0. The goal is to investigate and implement models to improve the performance of a baseline model, called Bidirectional Attention Flow (BiDAF). Specifically, the improvements mainly focus on 1) extending the baseline embedding layer implementation to include character-level word embeddings, 2) enhancing the baseline attention layer, 3) improving the output layer by conditioning the prediction of end position on the prediction of start position of the answer span. Experiment results show significant performance improvement with the additional character-level word embeddings, deep residual coattention, and a pointer network model in the output layer.

1 Introduction

Question answering (QA) is a crucial and challenging task in natural language processing that aims to build systems that can automatically answer questions given a passage or document. It has attracted lots of attention in the past several years due to its numerous practical real-world applications. The current state-of-the-art QA systems are almost all built on top of end-to-end training and pre-trained language models since BERT [1] was released in 2018. Prior to BERT, there has been lots of research on designing end-to-end neural network architectures for QA, with most of them focusing on attention scheme design and output decoding scheme design [2, 3, 4, 5, 6].

In this project, we focus on the design, implementation and evaluation of techniques and models without using pre-trained models for QA on the Stanford Question Answering Dataset (SQuAD) 2.0 [7] and aim to build a system to achieve the highest performance as possible on this dataset. The new challenge in SQuAD 2.0 is that systems must not only answer questions when possible, but also determine when no answer is supported by the paragraph and abstain from answering. To achieve this goal, we explore different techniques in three major layers of the end-to-end architecture, namely, the encoding layer, the attention layer and the output layer. Experiment results show that better performance can be achieved with different enhancements on top of the baseline model. Especially, with extra character embedding and deep residual coattention, we can achieve EM of 61.17 and F1 of 64.97 in comparison to EM of 58.32 and F1 of 61.78 of the baseline model on the development data set. On the test set, an EM of 59.966 and F1 of 63.386 can be achieved.

2 Related Work

During the pre-BERT era, most of the QA systems employed the recurrent neural network architecture and focused on attention scheme design and output decoding/prediction scheme design.

The bidirectional attention flow (BiDAF) model in [2] proposed a multi-stage hierarchical process to represent the context at different levels of granularity (character, word, context) and use a bidirectional attention flow mechanism, namely, context-to-query (C2Q) and query-to-context (Q2C), to achieve a query-aware context representation.

The dynamic coattention networks (DCN) proposed in [3] consists of a coattention encoder that captures the dependence between the question and the document, and a dynamic pointing decoder

that can estimate the start and end points of the answer span multiple times by iterating over potential answer spans. Unlike the attention flow in BiDAF, the coattention in DCN involves a second-level attention computation by taking weighted sums of the Q2C attention outputs using the C2Q attention distributions. Then the second-level attention outputs are concatenated with the first-level C2Q attention outputs and the sequence are fed through a bidirectional LSTM encoding layer.

Later, the same authors of the DCN model proposed a new attention scheme called deep residual coattention in [6]. This is basically an extension to the original DCN attention scheme by stacking coattention layers and merging coattention outputs from each layer with residual connections aiming to improve the modeling of long-range dependencies in the input texts.

In contrast to the BiDAF model, where the start location and the end location are predicted independently, [8] proposed to use pointer network to predict the start and end locations such that the end location prediction is conditioned on the start location prediction.

All of the above models and techniques were evaluated on SQuAD 1.0. In this project, we explore these techniques and modes on SQuAD 2.0.

3 Approach

The high level hierarchical multi-stage process of the system model is shown in Fig. 1. For three major layers of the model (the encoding layer, the attention layer and the output layer), we investigate the performance of different techniques.

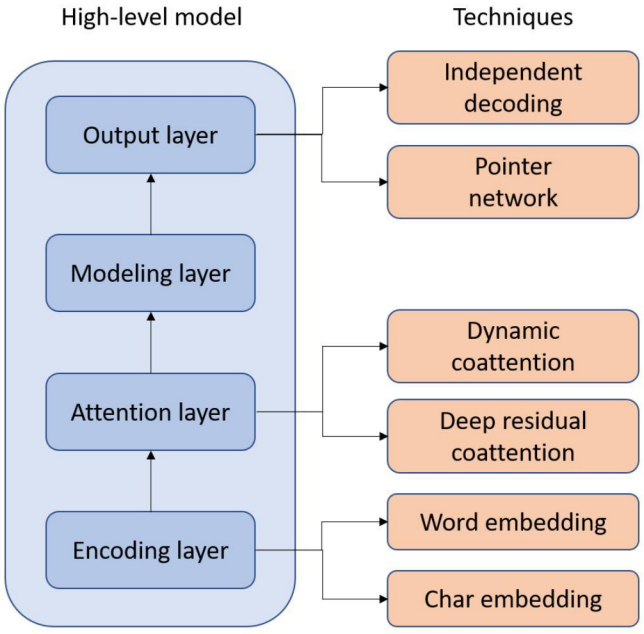

Figure 1: High level model architecture and techniques for each layer.

3.1 Baseline

The baseline is the BiDAF model proposed in [2]. The details of the BiDAF model can be found in the original paper or the project handout. Note that the attention layer implemented in the starter code is not the same as the scheme proposed in the original paper, instead it is based on the dynamic coattention scheme proposed in [3]. The details are described in Section 3.3.

3.2 Character-level word embeddings

The first enhancement implemented is integrating the character-level word embeddings into the embedding layer. This character-level embedding of each word is implemented using convolutional

neural networks (CNN) based on [9] as shown in Fig. 2. The embedding vectors of each character for each word is fed into the CNN. Then the CNN outputs a fixed-size vector for each word with max-pooling. Then the character-level embedding vector and the word-level embedding vector are concatenated and passed to a two-layer Highway Network.

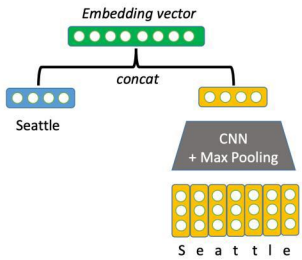

Figure 2: Concatenation of word embedding and character-level embedding.

3.3 Attention layer

The second core part of the model is the attention layer, which is responsible for finding query-aware representation for the context words.

The original attention scheme in BiDAF model is illustrated in Fig. 3, where a C2Q attention and a Q2C attention are calculated based on a shared similarity matrix $\boldsymbol{S}$ derived from the contextual embeddings of the context words $\boldsymbol{C}$ and the query words $\boldsymbol{Q}$. Both attention outputs are combined or simply concatenated with the contextual embeddings of the context words to form final attention output $\boldsymbol{G}$ and fed to the modeling layer.

A similar bidirectional attention scheme proposed in DCN model is illustrated in Fig. 4. As can be seen from the figure, the only difference is there is a second-level attention calculation after the Q2C attention output. This is the scheme that is implemented in the baseline BiDAF model in the starter code. The attention output is modeled as:

$$G = \textbf{concate}(C; A; C \odot A; C \odot B),$$

where $\odot$ is element-wise matrix multiplication.

As a second extension, we implemented the deep residual coattention (DRC) scheme as illustrated in Fig. 5 and compared the performance to the baseline attention scheme. As can be seen from Fig. 5, the main idea is to stack two coattention layers (each coattention layer computes the bidirectional attentions as shown in Fig. 4) and merge the coattention outputs from each layer with residual connections aiming to model long-range dependencies between the context words and the query words. The attention output is modeled as the following in the original paper:

$$G = \textbf{concate}(C; A; B; C2; A2; B2).$$

However, in our implementation, we choose to model the attention output as the following for better performance:

$$G = \textbf{concate}(C; A; C \odot A; C \odot B; C \odot A2; C \odot B2).$$

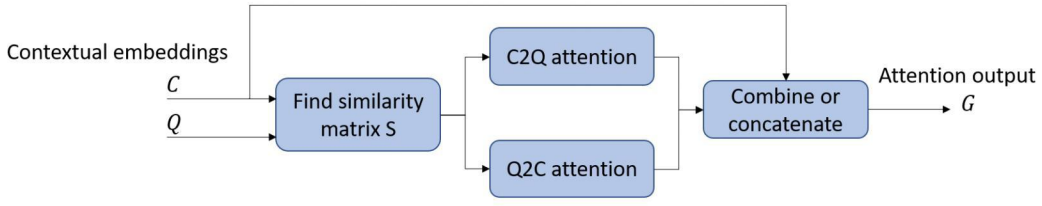

Figure 3: C2Q and Q2C bidirectional attention in BiDAF.

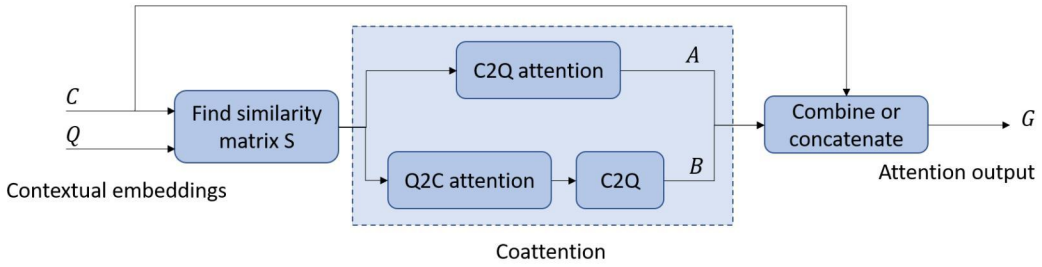

Figure 4: Coattention in DCN.

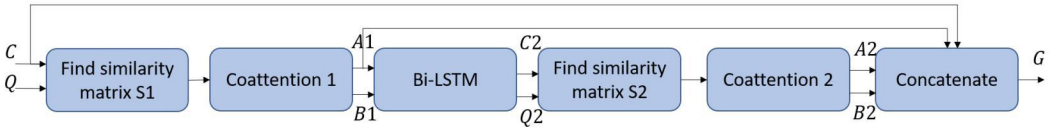

Figure 5: Deep residual coattention in DCN+.

3.4 Output layer enhancement with pointer network

The third enhancement implemented is based on the pointer network model proposed in [8]. In contrast to the baseline output layer where the start position and the end position are predicted independently, pointer network model predicts the probability distribution for the end position based on the start position probability distribution based on the outputs of the attention layer.

Specifically, for the baseline output layer, the probability distribution for the start position and the end position are produced as:

$$\boldsymbol{p}_{start} = \text{softmax}(\boldsymbol{W}_{start}[\boldsymbol{G}; \boldsymbol{M}]), \;\; \boldsymbol{p}_{end} = \text{softmax}(\boldsymbol{W}_{end}[\boldsymbol{G}; \boldsymbol{M}']),$$

where $\boldsymbol{G} \in \mathbb{R}^{8H \times N}$ and $\boldsymbol{M} \in \mathbb{R}^{2H \times N}$ are the outputs of the attention layer and the modeling layer, respectively, $\boldsymbol{M}' \in \mathbb{R}^{2H \times N}$ is the output of a bidirectional LSTM with $\boldsymbol{M}$ as input, $\boldsymbol{W}_{start}, \boldsymbol{W}_{end} \in \mathbb{R}^{1 \times 10H}$ are learnable parameters, H is the hidden layer size, and N is the context length.

The pointer network model is a RNN that is run for two steps, one for the start position, and one for the end position. Specifically, at time step k,

$$\boldsymbol{F}_k = \tanh(\boldsymbol{W}_1 \boldsymbol{M} + \boldsymbol{W}_2 \boldsymbol{h}_{k-1}), \;\; \boldsymbol{\beta}_k = \text{softmax}(\boldsymbol{W}_3 \boldsymbol{F}_k),$$

where $\boldsymbol{W}_1, \boldsymbol{W}_2, \boldsymbol{W}_3$ are learnable parameters, $\boldsymbol{\beta}_k \in \mathbb{R}^N$ is the probability distribution for the k-th position in the answer ($\boldsymbol{\beta}_1$ for start position and $\boldsymbol{\beta}_2$ for end position). $\boldsymbol{h}_{k-1}$ is the hidden vector at position $k-1$ modeled as:

$$\boldsymbol{h}_k = \text{LSTM}(\boldsymbol{M}\boldsymbol{\beta}_k^\top, \boldsymbol{h}_{k-1}).$$

This is essentially how $\boldsymbol{\beta}_2$ is conditioned on $\boldsymbol{\beta}_1$.

4 Experiments

4.1 Dataset

The official SQuAD 2.0 dataset is used. The dataset splits are summarized in Table 1. Note that more than 50% of the questions in the dev set are not answerable, while about only 33% of the questions in the train set are not answerable.

Splits	Size	No-answer ratio	Comment
train	130319	33.38%	All taken from the official SQuAD 2.0 training set
dev	6078	52.12%	Roughly half of the official dev set, randomly selected
test	5915	N/A	Remaining examples from the official dev set, plus hand-labeled examples

Table 1: Dataset splits information.

The statistics of context word length, question word length, and answer word length are illustrated in Fig. 6. As can be seen, the majority of the contexts are within 300 words, and the majority of the questions are within 20 words. In terms of answer length, the majority are within 5 words. Both dev and train set exhibit similar statistics.

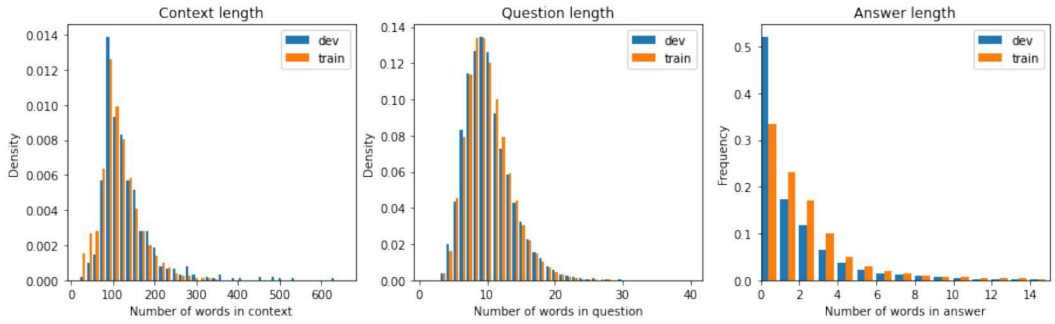

Figure 6: Dataset statistics.

4.2 Evaluation metrics

Exact match (EM) and F1 scores are used as evaluation metrics. EM measures whether the answer span matches exactly with the ground truth answer. F1 scores are computed as the harmonic mean of precision and recall, where precision is calculated as the number of correct words divided by the length of the predicted answer, and recall is calculated as the number of correct words divided by the length of the ground truth answer. For evaluation, the predicted answer is measured against 3 golden answers for each question, and the highest score among the three is recorded.

4.3 Model details

For the sake of fair comparison, in all the models, the character-level embedding dimensionality is 64 and the word level embedding dimensionality is 300. We use a batch size of 64 and a hidden layer size of 100. We train all the models for 30 epochs, with a learning rate of 0.5 and drop-out probability of 0.2.

4.4 Results and discussions

The results of different models on the development set are summarized in Table 2. We evaluate different extensions to the baseline BiDAF model. For pointer network in the output layer, we explore two options for the implementation. One option is to use separate weight parameters for the start position prediction and the end position prediction (PN-1 in the table), and the other option is to share the weight parameters between the two predictions (PN-2 in the table).

We also investigate how the performance of different models vary with respect to the length of contexts and questions and the results are summarized in Table 3. The data points in the development set are categorized into four groups, based on the length of contexts and the length of answers (we do not include results for the impact of question length on performance since it is very similar to the impact of context length). If there are more than 100 words in a context, then it is classified as a long context; otherwise, it is classified as a short context. Similarly, if there are more than 3 words in an answer, it is classified as a long answer, otherwise, it is classified as a short answer.

As can be seen, with only char-embedding enhancement, the baseline BiDAF performance can be improved slightly, and the improvement is mainly for questions with long answers and long contexts as shown in Table 3. With only pointer network modeling enhancement, the performance improvement is quite small. With both char-embedding and pointer network modeling, the model can achieve an EM score of 61.07 and an F1 score of 64.41. Sharing weights in the pointer network modeling causes slight performance degradation and it is quite strange that the degradation mainly happens for questions with short answers.

With deep residual coattention, char-embedding and independent start and end position prediction as in baseline model, comparable performance can be achieved (EM of **61.17** and F1 of **64.97**) as BiDAF with char-embedding and pointer network modeling. For *test set*, an EM of **59.966** and F1 of **63.386** is achieved for this model. However, adding pointer network modeling to the deep residual coattention scheme causes slight performance degradation. The main degradation happens for questions with long answers as shown in Table 3. This suggests that the current pointer network implementation is not optimal for deep residual coattention scheme, and the attention layer and the output layer need to be tuned together in order to achieve good performance.

Models	EM	F1	AvNA
BiDAF (baseline)	58.21	61.59	68.49
BiDAF + char-embed	59.47	62.77	69
BiDAF + pointer network (PN-1)	58.54	61.81	68.29
BiDAF + char-embed + PN-1	61.07	64.41	71.21
BiDAF + char-embed + PN-2	60.01	63.35	69.84
DRC + char-embed	61.17	64.97	70.86
DRC + char-embed + PN-1	60.44	63.52	69.70

Table 2: Model performance results on development set.

Also as shown in Table 3, there is no significant performance difference between long contexts and short contexts for all models. This is possibly due to the fact that all models use bidirectional attention and thus are able to focus on the most relevant words pretty well irrespective of the length of contexts. However, there is notable performance degradation for the group with long answers, especially in terms of EM. This is as expected, as the longer the answer, the harder to predict the correct answer span.

Models	Long context		Short context		Long answer		Short answer	
	EM	F1	EM	F1	EM	F1	EM	F1
BiDAF (baseline)	58.32	61.78	58.00	61.23	46.85	59.89	59.91	61.84
BiDAF + char-embed	60.04	63.18	58.43	62.02	49.42	62.45	60.98	62.82
BiDAF + PN-1	58.58	62.03	58.48	61.39	47.88	60.41	60.15	62.02
BiDAF + char-embed + PN-1	61.41	65.09	59.33	62.94	49.03	62.55	62.43	64.60
BiDAF + char-embed + PN-2	60.45	63.93	59.19	62.30	50.32	63.31	61.46	63.36
DRC + char-embed	61.62	64.97	60.33	63.50	50.32	62.44	62.79	64.75
DRC + char-embed + PN-1	61.23	64.32	59.00	62.06	48.52	60.45	62.23	63.98

Table 3: Model performance for different lengths of documents and answers on the development set.

5 Analysis

To understand more deeply about the behavior and performance of the best model (deep residual coattention with extra character embedding), we examine its performance across question types. The average F1 score and the number of questions for each question type are summarized in Table 4. We can see that the model performs best for "when" questions and does not do well for "why" and "how" questions. This is intuitive since these questions are more complicated. Surprisingly, the performance for "where" questions is also below average. We inspected some of the error answers and several examples with comments are provided in Table 5. We also examined the errors made in the other question types and some examples are listed in Table 6.

Question type	What	Who	When	Which	Where	Why	How	Other	**Total**
Size	3653	687	445	213	252	87	555	59	5951
F1	63.72	66.68	74.25	72.49	60.41	60.28	59.56	49.70	64.45

Table 4: Best model performance across question types.

Context	"Several families of Byzantine Greece were of Norman mercenary origin during the period of the *Comnenian Restoration*, when Byzantine emperors were seeking out western European warriors ....."
Question	"Where were several Norman mercenary familes originate from?"
Prediction	"Comnenian Restoration"
Ground truth	Not answerable
Comment	The model made a mistake on the answer type. The predicted answer is a time/period. Probably it is because the model does not understand the "when" clause followed the predicted answer.
Context	"The legendary religious zeal of the Normans was exercised in religious wars long *before the First Crusade carved out a Norman principality in Antioch*. They were major foreign participants in the Reconquista in Iberia ....."
Question	"Where did the Normans carve out a principality *before the First Crusade*?"
Prediction	"Antioch"
Ground truth	Not answerable
Comment	This is a very tricky question for the model. The model indeed gave the correct answer type (Antioch is indeed a place), but the model does not understand the difference of "before" in the context and the question.
Context	"The customary law of Normandy was developed between the 10th and 13th centuries and survives today through the legal systems of Jersey and Guernsey in the *Channel Islands*. Norman customary law was transcribed in two customaries in Latin by two judges ....."
Question	"Where are Jersey and Guernsey?"
Prediction	No answer
Ground truth	Channel Islands
Comment	The model failed to give an answer for this typical "where" question. It seems that the model does not understand the expression with proposition "in", possibly because it didn't see many such examples in the training set.

Table 5: Some examples of errors in "where" question type.

We also investigate the F1 score distribution on the development set for the best performed model and the result is shown in Table 7. We can see the model predicts a perfect answer (with 100% F1 score) for 61.2% of the questions and a wrong answer (with 0% F1 score) for 33.4% of the questions. It predicts a partial correct answer for only the remaining 5.4% questions. Out of the 1986 completely wrong answers, 1204 (60.6%) are wrong because the questions are not answerable, but the model gives an answer, 530 (26.7%) are wrong because the questions are answerable, but the model predicts no answer.

F1	100%	0%	in between
Size	3640 (61.2%)	1986 (33.4%)	325 (5.4%)

Table 7: F1 score distribution.

6 Conclusion

In this project, we explored different techniques in the encoding layer, the attention layer and the output layer of an end-to-end neural network architecture for question answering. Experiment results show that better performance can be achieved with different enhancements on top of the baseline model. Especially, with extra character embedding and deep residual coattention, we can achieve EM of 61.17 and F1 of 64.97 in comparison to EM of 58.32 and F1 of 61.78 of the baseline BiDAF model. We also investigated the behavior of the best performed model by breaking down F1 score and examining the performance across different context lengths, answer lengths, question types, and error distributions. By inspecting some of the error examples, we found that the model performs poorly mainly when it involves some kind of reasoning or advanced/complicated sentence structures.

Context	"In contrast, an instance of this problem is a rather concrete utterance, which can serve as the input for a decision problem ....."
Question	"Is a problem instance typically characterized as abstract or concrete?"
Prediction	"concrete utterance"
Ground truth	concrete
Comment	The model basically found the right place for the answer. But in order to be more precise, it needs to understand the meaning of "or" structure in the question.
Context	"A function problem is a computational problem where a single output (of a total function) is expected for every input, but the output is more complex than that of a decision problem ....."
Question	"Is the output of a functional problem typically characterized by a simple or complex answer?"
Prediction	"the output is more complex"
Ground truth	complex
Comment	Similar to the previous example, the model can locate the general range of the answer, but failed to give a precise answer to the question.
Context	"A decade after the 1973 oil crisis, *Honda, Toyota and Nissan*, affected by the 1981 voluntary export restraints, opened US assembly plants and established their luxury divisions (*Acura, Lexus and Infiniti, respectively*) to distinguish themselves from their mass-market brands."
Question	"Name a luxury division of Toyota."
Prediction	Acura, Lexus and Infiniti
Ground truth	Lexus
Comment	Again, the model locate the general range of correct answer. But in order for the model to precisely identify Lexus is the division of Toyota, it needs to understand "respectively" and relate it to previous context "Honda, Toyota and Nissan".

Table 6: Some examples of errors in other question types.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, 2018.

[2] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. In *International Conference on Learning Representations (ICLR)*, 2017.

[3] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. In *International Conference on Learning Representations (ICLR)*, 2017.

[4] Microsoft Research Asia Natural Language Computing Group. R-net: Machine reading comprehension with self-matching networks. In *https://www.microsoft.com/en-us/research/wp-content/uploads/2017/05/r-net.pdf*, 2017.

[5] Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. In *International Conference on Learning Representations (ICLR)*, 2017.

[6] Caiming Xiong, Victor Zhong, and Richard Socher. DCN+: mixed objective and deep residual coattention for question answering. *CoRR*, 2017.

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

[8] Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. *CoRR*, 2016.

[9] Yoon Kim. Convolutional neural networks for sentence classification. *CoRR*, 2014.