GE-BiDAF: A Question Answering model for SQuAD

Binbin Xiong
Stanford University
bxiong@stanford.edu

Minfa Wang
Stanford University
minfa@stanford.edu

Abstract

In this project, we learned from several state-of-art Question Answering (QA) models and proposed Gated Extension of Bi-Directional Flow (GE-BiDAF) model, a novel attention based end-to-end neural network for QA. On the SQuAD dataset[1], our single model achieved 76.6 F1 score and 66.4 EM score on dev set, and the ensembled model achieved 78.3 F1 score and 68.6 EM score on dev set and 79.9 F1 score and 70.5 EM score on test set.

1 Introduction

Recent years, question answering systems has gained increasing popularity due to its thrive in application and boost to academic researches. A question answer system is a system designed to answer questions posed in natural languages. Among several types of such systems, one special type is that the answer is a segment of text from the corresponding reading passage. In such case, given a context and question, the system highlights a span within the context as the answer to the question. SQuAD[1] is such a reading comprehension dataset which consists of 100,000+ questions posed by crowd-workers on a set of Wikipedia articles. Until early 2018, the state of art model has already surpassed human level performances.

In this project, we experimented and analyzed with what we learned from some of the top performing models, and proposed our own constructed model. The rest of the paper is organized as follow: Section 2 introduces related work and common technologies used in state of art models, section 3 presents our model methodology and implementation details. In section 4 we present experimental results and model analysis. We then conclude our proposal in section 5 and discussed some future works.

2 Background

One popular problem definition used by many state of art models[2,3,4] is to predict the start word and end word position for the given passage and question, this can be formulated as minimizing loss over $index^{start}$ and $index^{end}$.

In BiDAF[2], they proposed a hierarchical multi-stage architecture for modeling the representations of the context paragraph at different levels of granularity. In R-NET[3], they leveraged gated attention-based recurrent networks to obtain question-aware passage representation then refine it using self-matching attention mechanism. In Dynamic CoAttention Networks[4], they generated both question-aware passage and passage-aware question representation, then used dynamic pointing decoder to produce answer span.

In general, these models can be decomposed into four layers, embedding layer, attention (encoding) layer, modeling layer and output layer. For the embedding layer, words or even characters will be mapped into embedded vectors, then in encoding layer, context passage ad questions will be encoded

[1] http://web.stanford.edu/class/cs224n/project.html

with various of attention calculation modules. The general rule of thumb here is to encode questions information into contexts and contexts information into questions. Later, in the modeling layer, these encodings will be further fused or connected to let them interact with each other. Afterwards, they will be put to the final output layer to predict the start and end index of the answer span.

3 Modeling

Inspired by previous works and BiDAF[2] in particular, we propose an end to end model that is illustrated in figure 1. In a nutshell, it uses the original BiDAF architecture as the skeleton and built several extensions on top.

We chose BiDAF model as our improved baseline model because of its modularity and the clear intent and intuitive interpretation for every single layer in the architecture. These two features made the model very extensible.

In the figure, we highlighted our extensions to the BiDAF model in blue background. The **FC + Batch Norm** unit is used to condense the information into lower dimensions. The **CoAttention** unit is used to enhance query-aware context representations. The **Self-Attention** unit is used to capture more interactions between context words. The **gate** is used to re-scale context representations based on its relevance to the question.

In the following sections, we will elaborate more about each layer and the rationale behind the design.

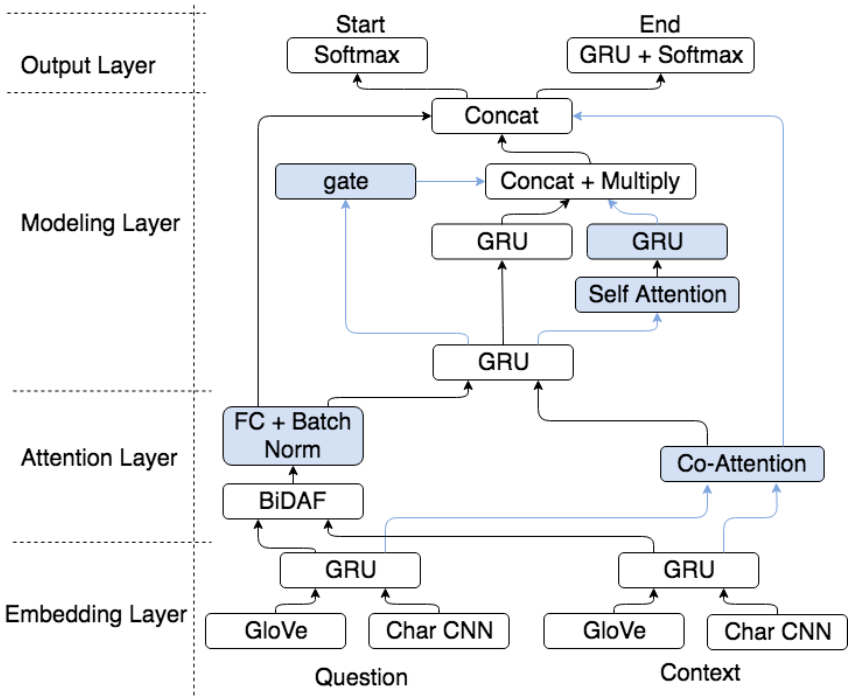


Figure 1: Architecture Overview.

3.1 Embedding Layer

The embedding layer is primarily responsible for constructing word level vector representations for all words in context and question.

In this layer, we use both word embedding and character level embedding, hoping that character level embedding can help with out-of-vocabulary words during testing. The word embedding is pre-trained with GloVe[10] with dimension 100 while the character level embedding is trained from

scratch using CNN[5] . Note that since our training data is limited, to avoid over fitting, we keep the word embedding fixed. As can be seen in section 4, character level embedding indeed helps with the model performance.

3.2 Attention Layer

The attention layer is primarily responsible for constructing query-aware context representations.

In our model, we first applied bi-directional attention following instructions from BiDAF[2]. It is a very efficient approach to encode question information into the context. The BiDAF output $\mathbf{b}_i$ is computed as follow:

$$\mathbf{S}_{ij} = \mathbf{w}_{sim}^T[\mathbf{c}_i; \mathbf{q}_j; \mathbf{c}_i \circ \mathbf{q}_j]$$

$$\tilde{\mathbf{c}_i} = \sum_{j=1}^{M} softmax(\mathbf{S}_{i,:})_j \mathbf{q}_j$$

$$\tilde{\mathbf{q}} = \sum_{i=1}^{N} softmax(max_j(\mathbf{S}_{ij}))_i \mathbf{c}_i$$

$$\mathbf{b}_i = [\mathbf{c}_i; \tilde{\mathbf{c}_i}; \mathbf{c}_i \circ \tilde{\mathbf{c}_i}; \mathbf{c}_i \circ \tilde{\mathbf{q}}]$$

Note that $\mathbf{b}_i$ has a very large dimension of $8 * hidden\_size$. It brings 3 problems to the subsequent RNN layer: 1) large number of parameters, 2) dimension imbalance between input and hidden vectors and 3) slow training speed. So we inserted a fully connected layer with batch norm after this layer with dimension equals $2 * hidden\_size$. It effectively addresses problems with memory and speed, and based on our isolated experiment, the model performance remains identical. The output of this layer is $\tilde{\mathbf{b}}_i$:

$$\tilde{\mathbf{b}}_i = BatchNorm(FC_{ReLU}(\mathbf{b}_i))$$

In parallel to the BiDAF branch, we created a co-attention layer[4] to extract more query-aware context information.

The reason we chose co-attention here is because 1) it has proven to be an efficient learning structure for similar tasks[4], 2) it only contains a relatively small number of trainable parameters hence it would not cause too much burden to the system, and 3) its structure is very different from the BiDAF attention structure so it might be able to learn some distinct language patterns.

For simplicity we did not include the sentinel vector proposed from the original CoAttention paper[4]. The output $\mathbf{s}_i$ is:

$$\mathbf{L}_{ij} = \mathbf{c}_i^T \mathbf{q}'_j$$

$$\mathbf{a}_i = \sum_{j=1}^{M} softmax(\mathbf{L}_{i,:})_j \mathbf{q}'_j$$

$$\mathbf{b}_j = \sum_{i=1}^{N} softmax(\mathbf{L}_{:,j}) \mathbf{c}_i$$

$$\mathbf{s}_i = \sum_{j=1}^{M} a_{i,j} \mathbf{b}_j$$

The output of the whole attention layer is simply a concatenation of $\tilde{\mathbf{b}}_i$ and $\mathbf{s}_i$: $\mathbf{a}_i = [\tilde{\mathbf{b}}_i; \mathbf{s}_i]$

3.3 Modeling Layer

The modeling layer is primarily responsible for capturing the interactions between context words.

First, we followed the advice from the BiDAF model to create a GRU layer suggested by the BiDAF model:

$$\mathbf{r}_i = biGRU(\mathbf{a}_i)$$

In parallel, we added a self-attention layer with GRU to capture more sophisticated interactions. The self-attention is implemented using the scaled multiplicative attention[8]. It has all the nice features from the original multiplicative attention[7] and is more robust on larger dimensions. The output of this layer is **e**:

$$\mathbf{S}^e_{ij} = \mathbf{a}_i W^e \mathbf{a}_j$$
$$\alpha^e_i = softmax(\mathbf{S}^e_{i,:}/D_{dim})_i$$
$$\mathbf{e}_i = \sum_i^M \alpha^e_i \mathbf{a}_i$$
$$\tilde{\mathbf{e}}_i = biGRU(\mathbf{e}_i)$$

We created another gated GRU layer to the concatenation of $\mathbf{r}_i$ and $\tilde{\mathbf{e}}_i$ to enable more complex interactions among context words. The gate is used to determine the importance of words to answer the question. The output $\mathbf{o}_i$:

$$\mathbf{g}_i = FC_{sigmoid}(\mathbf{r}_i)$$
$$\tilde{\mathbf{r}}_i = biGRU([\mathbf{r}_i; \mathbf{e}_i])$$
$$\mathbf{o}_i = \mathbf{g}_i \circ \tilde{\mathbf{r}}_i$$

3.4 Output Layer

The output layer is responsible for making the decision of the start and end position of the answer in context.

Similar to BiDAF[2], we first built residual connection for $\tilde{\mathbf{b}}_i$ and $\mathbf{s}_i$, and then simply use a softmax layer for making start predictions, and GRU plus softmax layer for making end predictions.

Although there is no explicit dependency between start and end predictions, we believe the GRU could resolve a lot of this information implicitly. The formula to make the predictions are:

$$\tilde{\mathbf{o}}_i = [\tilde{\mathbf{b}}_i; \mathbf{s}_i; \mathbf{o}_i]$$
$$\mathbf{p}^{start}_i = softmax(FC_{ReLU}(\tilde{\mathbf{o}}_i))$$
$$\mathbf{p}^{end}_i = softmax(FC_{ReLU}(\tilde{\mathbf{o}}_i))$$

3.5 Model Ensemble

Based on our single model, we trained it multiple times and combine the output of all of them to produce the final output. We have tested with several different combine schemes, one is to sum up the probability directly while the other is to use weighted sum, where the weight is the F1 score on dev set for each of the model. Further more, we implemented beam search during testing: instead of using $index^{start/end} = argmax^n_{i=1}(p^{start/end}_i)$, we keep top *beam_search_size* $p^{start/end}_i$ predictions as candidates and find $index^{start/end} = argmax^{beam\_search\_size}_{i,j=1}(p^{start}_i * p^{end}_j)$.

3.6 Other design choices

We chose GRU as the basic RNN cell in the architecture, because it shows comparable performance as LSTM while having much fewer parameters.

In the literature, sometimes people might put RNN cell as part of the attention layer. In this article, we separate out RNN and attention for consistency and modularity.

4 Results and Analysis

In this section, we present our model settings, list the performances of all the models we have tried and also analyze some error examples for our best performing model.

4.1 Setup

We developed GE-BiDAF in incremental steps. Each time we add one module, validate its performance, and then re-iterate. In order to better compare models, we explored a set of fixed parameters that are applicable to all model variants we experimented with.

- **Adam optimizer with learning rate: 0.001.** In practice, because Adam Optimizer has the ability to automatically adjust the learning rate, the model is able to converge reasonably fast in a relatively wide range of learning rate. This default learning rate usually allows the model to reach close-to-optimal performance in a few thousand iterations.

- **Batch size: 100.** A smaller batch size will create more stochastic behavior and cause model to converge slower. A larger batch size will lead t o out-of-memory error for large and complex models. A batch size of 100 is a sweet spot we found appropriate in our experiment.

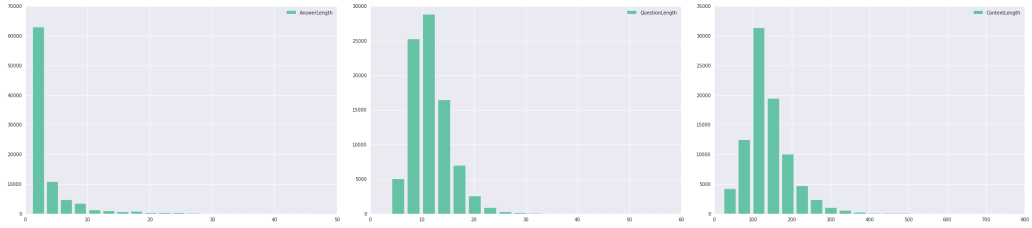
- **Hidden size: 100.** A smaller hidden size will restrain the model capacity. A larger hidden size will cause out-of-memory error and make and model more prone to over-fitting. For reference, r-net used a hidden size of 75 and it managed to get competitive performance.

- **Context length: 400.** As shown in Figure 2. Based on the training set, this context length will drop 0.19% of examples, but speed up the training process noticeably.

- **Question length: 27.** As shown in Figure 2. Based on the training set, this question length will drop 0.27% of examples, but speed up the training process noticeably.

- **Smart span selection with beam search.** In practice we found that beam_search_size=5 is mostly useful. Further more, as suggested by [6], during beam search, we require $0 \leq index^{end} - index^{start} \leq 15$ since the naive model is trained to predict start and end positions independently so it is even possible to have a predicted end index smaller than the start index. Although in the training set there are 2.3% answers that are longer than 15, in practice we found setting threshold to 15 gives the best boost, which is about 0.5 point on dev set. Adding this requirement can help especially when the answer appears more than once in the context, for e.g., the answer is a single word.

Figure 2: Length distribution in training set. From left to right: answer, question, context



4.2 Model Iteration

As listed in table 1 and 2, started from the given code as baseline, we iteratively added modules to our model, and only kept the ones that work well. We discovered significant boost effect of beam search, so we included it in all BiDAF variants. The additions of batch norm, co-attention and self-attention all contributed to small fractions of the model performance. The final gating mechanism contributes to most of the improvements on the basis of BiDAF.

Model	Dev		Test	
	F1	EM	F1	EM
GE-BiDAF(single)	76.6	66.4	-	-
GE-BiDAF(ensembled)	78.3	68.6	79.9	70.5
BiDAF(single)	77.3	67.7	77.3	68.0
BiDAF(ensembled)	80.7	72.6	81.1	73.3
DCN(single)	65.4	75.6	66.2	75.9
DCN(ensembled)	70.3	79.4	71.2	80.4
RNet(Jan, 2018)	-	-	82.6	88.4

Table 1: Performance comparison with other models

Model[a]	DevF1	DevEM
BL	38.0	26.4
BL+CNN	41.2	31.2
BL+CNN+BS	45.1	36.9
BiDAF-CNN+BN	74.0	63.5
BiDAF+CA+SA+BN	74.7	64.0
GE-BiDAF(single)	76.6	66.4

Table 2: Performance comparison between ablation models

[a]BL: baseline, CNN: character level CNN embedding, BS: beam search, CA: co-attention, SA: self attention, BN: batch norm

4.3 Error Analysis

The questions in the SQuAD dataset could be grouped into the following categories based on the type of questions: *when*, *what*, *who*, *where*, *how*, and *others*. In figure 3 we illustrated holistically how well does the model behave on different types of questions respectively by comparing the question length with the ground truth data.

It appears the model performs well for *when*, and *who*, question types, but shows noticeable headroom for *what* and *where* question types. We looked into the biggest loss (lowest prediction F1 score) examples under these two categories and summarized the following error patterns. For each pattern, we also show one example, where the correct answer with highest F1 score is highlighted in the context.

	What	When	Who	Where	How	Other
F1	0.751	0.894	0.825	0.758	0.789	0.789
EM	0.641	0.847	0.762	0.650	0.691	0.689
Count	4763	695	1096	431	1090	2495

Table 3: F1 and EM for different questions

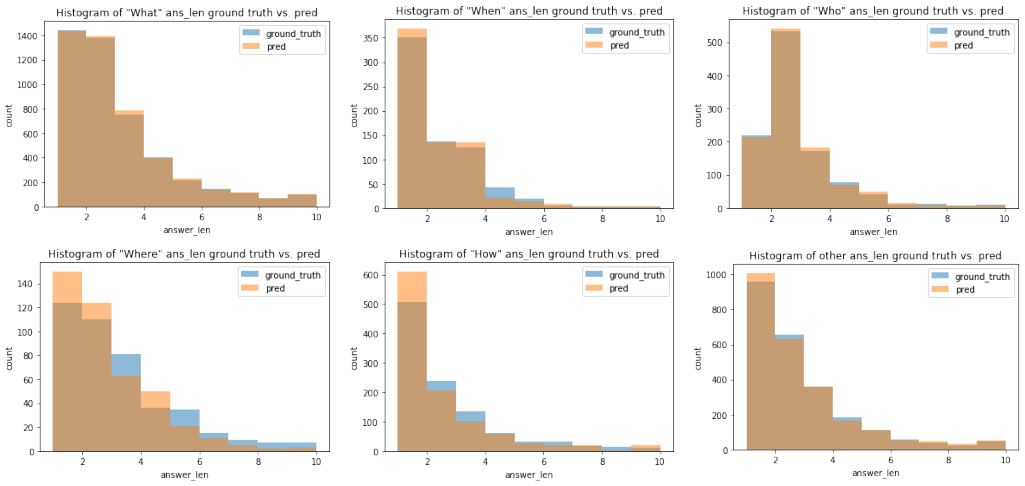


Figure 3: Histogram of answers length comparison between ground truth and prediction for different types of questions.

Biased towards preposition words

This is a common pattern mostly seen in **where** type questions. The learned model tends to give strong bias for phrases after preposition. As can be seen from the following example, the model is confused by **near** which actually tells time, not a location.

Question: Where did Tesla go to feed the pigeons daily?

Context: Near the end of his life, Tesla **walked to the park** every day to feed the pigeons and even brought injured ones into his hotel room to nurse back to health.

Prediction: "near the end of his life"

Ambiguous answers

This pattern happens most for **what** type questions. For some questions, there are more than one phrase in the context that could be regarded as answer, it's just the rater's favor.

Question: What is notable about the Amazon forest when it is seen from space?

Context: Deforestation is considerable, and **areas cleared of forest** are visible to the naked eye from outer space.

Prediction: "deforestation"

Incomplete answer phrase

This is another pattern that happens frequently for **what** type questions. For some of the answers the model generates are not of complete phrase. Though this type of error mostly won't hurt F1 much, they should be relatively easy to fix. Adding POS tag might help deal with this error,

Question: What drives down wages in a job with many workers willing to work a lot?

Context: A job where there are many workers willing to work a large amount of time (high supply) competing for a job that few require (low demand) will result in a low wage for that job. This is because **competition between workers** drives down the wage.

Prediction: "competition between workers drives"

Wrong attention

This error pattern applies to all type of questions. Once the attention is wrong, the prediction is usually far away from the real answer in context passage.

Question: What is the most important type of Norman art preserved in churches?

Context: In Britain, Norman art primarily survives as stonework or metalwork, such as capitals and baptismal fonts. In southern Italy, however, Norman artwork survives plentifully in forms strongly influenced by its Greek, Lombard, and Arab forebears. Of the royal regalia preserved in Palermo, the crown is Byzantine in style and the coronation cloak is of Arab craftsmanship with Arabic inscriptions. Many churches preserve sculptured fonts, capitals, and more importantly **mosaics**, which were common in Norman Italy and drew heavily on the Greek heritage.

Prediction: "stonework or metalwork"

Unable to handle too complex sentences.

This is a common error pattern that applies to all type of questions. Some passage contains too complex structure that the model seems to be unable to handle. As shown in the following example, there are multiple person entities and locations appear in the context.

Question: Where did Jebe's division of Genghis Khan's army campaign in Khwarezmia?

Context: The Mongol army under Genghis Khan, generals and his sons crossed the Tien Shan mountains by entering the area controlled by the Khwarezmian Empire. After compiling intelligence from many sources Genghis Khan carefully prepared his army, which was divided into three groups.

His son Jochi led the first division into the northeast of Khwarezmia. The second division under Jebe marched secretly to **the southeast** part of Khwarzemia to form, with the first division, a pincer attack on Samarkand. The third division under Genghis Khan and Tolui marched to the northwest and attacked Khwarzemia from that direction.

Prediction: "the northwest"

5 Conclusion And Future Work

In this paper we presented a novel model which leverages multiple attention techniques including BiDAF, Co-Attention, Self-Attention as well as various tweaks. We showed how we came up with this model with iterative experimental results, analyzed the best model with examples. Based on our experiences, we are interested in the following directions in future work.

Explicit dependency between start and end predictions

In R-NET[3] and answer-pointer network[9], they proposed to construct more explicit dependency between start and end predictions.

During our development, we tried to replace the output layer of the current architecture with the above convention. We didn't observe a particular faster convergence in our experiment. One explanation is that we have an additional GRU in the output layer, it's powerful enough to learn the implicit dependency of the start prediction indices, and it makes the overall computational graph simple and easy to optimize.

Despite the fact that it didn't illustrate immediate gain the overall idea of conditioning end prediction on the start prediction still appear to be a sounding approach and worth more explorations.

Modeling on specific loss patterns

In the error analysis, we observed that the model performs differently on different types of questions. We could explore more ideas on learning specific loss patterns for a subset of questions. For example, our model performs weak for *where* questions, the answer for such questions usually refers to locations. Adding NER-LOC feature might help the model to better deal with such quesitons.

Acknowledgments

We would like to thank the instructor and TAs for all the help during the course. It is the tailored and well designed course material and projects that helped us to learn and reach this far.

6 References

[1] Rajpurkar P, Zhang J, Lopyrev K, et al. Squad: 100,000+ questions for machine comprehension of text[J]. *arXiv preprint* arXiv:1606.05250, 2016.

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

[3] Appers in *https://www.microsoft.com/en-us/research/wp-content/uploads/2017/05/r-net.pdf*

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

[5] Yoon Kim. Convolutional neural networks for sentence classification. In EMNLP, 2014.

[6] Chen, Danqi, et al. Reading wikipedia to answer open-domain questions. *arXiv preprint* arXiv:1704.00051 (2017).

[7] Minh-Thang Luong, Hieu Pham, Christopher D. Manning. Effective Approaches to Attention-based Neural Machine Translation. *arXiv preprint* arXiv:1508.04025

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. *arXiv preprint* arXiv:1706.03762

[9] Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. *arXiv preprint* arXiv:1608.07905, 2016b

[10] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.