CS230

Question-Answering System for SQuAD

Weiquan Mao
Department of Electrical Engineering
Stanford University
mwq@stanford.edu

Abstract

In this paper, we produce a question answering system that works well on SQuAD. BiDAF model is used as our baseline model, which pushes Dev F1 score to 60 and Dev EM score to 57. And SRU architecture is applied to accelerate the training process as well as improve the performance. Then we applied a language representation model called BERT(Bidirectional Encoder Representations from Transformers) on SQuAD dataset. With one additional output layer, we experiment with different hyper-parameters in fine-tuning pre-trained BERT representations. Aiming to improve upon a standard BERT implementation, we have tried adding additional layers after BERT, appling L1 regularization. After ensembling all models, we now pushes SQuAD 2.0 question answering Dev F1 score to 79.944, Dev EM score to 73.643, Test F1 score to 78.841 and Test EM score to 76.010.

1 Introduction

Question-Answering System is one of the most popular natural language process tasks due to the creation of large question answer datasets. This can be used in many practical applications such as virtual assistants and automated customer service. The release of the Stanford Question Answering Dataset [3] has facilitated rapid progress in this field. The input to our model is a paragraph and a question about that paragraph. Our model uses BiDAF as baseline, Simple Recurrent Unit-BiDAF as a method to speed up the training, BERT as the core, L1 as regularization. The goal is to answer the question correctly - select the span of text or N/A if there is no answer in the paragraph. Another method to improve the performance of our model is ensemble, where we tried multiple models with different hyperparameters and mechanisms.

2 Related work

In the past few years, reading comprehension with neural networks has been studied thoroughly. Most of the high-performing models uses neural attention mechanism to combine the representations for the context and the question. Bi-Directional Attention Flow network[4] is one among them, which represents the context at different levels of granularity and uses a bi-directional attention flow mechanism to achieve a query-aware context representation without early summarization. Besides BiDAF, there are also other attention mechanism such as self-attention[5] and coattention[7]. However, since last year, Bidirectional Encoder Representations from Transformers [1] (BERT) has achieved state-of-the-art performance for eleven NLP tasks, like Question Answering[3] and Question Natural Language Inference[6].

CS230: Deep Learning, Winter 2019, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

3 Dataset and Features

We use SQuAD 2.0 as the reading comprehension data set. The paragraphs in SQuAD are from Wikipedia. The questions and answers are using labeling from Amazon Mechanical Turk. There are around 150k questions in total, and roughly half of the questions cannot be answered using the provided paragraph. However, if the question is answerable, the answer is a chunk of text taken directly from the paragraph. This means that SQuAD systems don't have to generate the answer text – they just have to select the span of text in the paragraph that answers the question.

The SQuAD dataset has been split into three sets:

- Train set with 129,941 examples, all taken from the official SQuAD 2.0 training set.
- Dev set with 6078 examples, randomly selected from the official dev set. For the milestone, we are only evaluating on the dev set, and has not tested anything on the test set yet.
- Test set with 5921 examples, the remaining examples from the official dev set along with some hand-labeled examples.

Here is an example of data:

- **Question**: What does not depend on the immune system's ability to distinguish between the self and others?
- **Context**: Both innate and adaptive immunity depend on the ability of the immune system to distinguish between self and non-self molecules. In immunology, self molecules are those components of an organism's body that can be distinguished from foreign substances by the immune system. Conversely, non-self molecules are those recognized as foreign molecules. One class of non-self molecules are called antigens (short for antibody generators) and are defined as substances that bind to specific immune receptors and elicit an immune response.
- **Answer**: N/A

4 Methods

4.1 Basline: BiDAF

Our baseline model based on BiDAF [4]. It is composed of Embedding Layer, Encoder Layer, Attention Layer, Modeling Layer and Output layer.

Specifically, the embedding layer performs an embedding lookup to convert the indices into word embedding, which is done for both the context and the question. A Highway Network is also used to refine the embedded representation. The encoder layer uses a bidirectional LSTM to allow the model to incorporate temporal dependencies between timesteps of the embedding layer's output. The main idea of attention layer is that attention flows both ways - from the context to the question and from the question to the context. The modeling layer is tasked with refining the sequence of vectors after the attention layer. It integrates temporal information between context representations conditioned on the question. The output layer is tasked with produing a vector of probabilities corresponding to each position in the context.

Our loss function for the baseline model is the cross-entropy loss for the start and end locations. We average across the batch and use Adadelta optimizer to minimize the loss.

4.2 SRU-BiDAF

The recurrent architectures like LSTM we used in the baseline use gating to control the information flow to alleviate vanishingand exploding gradient problems. However, the computation of the feed-forward netword, especially the matrix multiplication is the most expensive operation in the process. We applied a Simple Recurrent Unit(SRU)[2] architecture. The core idea is making the gate computation dependent only on the current input of the recurrence. This leaves only the point-wise multiplication computation as dependent on previous steps, which is relatively lightweight.

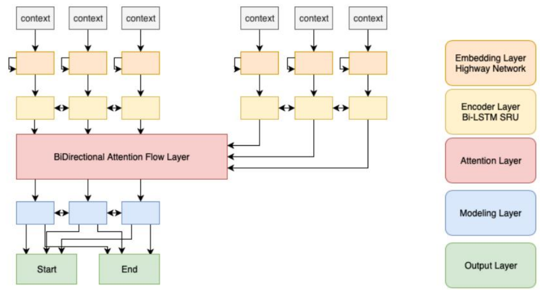

Figure 1: The structure of SRU-BiDAF

4.3 Bidirectional Encoder Representations from Transformers

BERT achieves state-of-the-art performance for eleven NLP tasks, like Question Answering and Question Natural Language Inference, through only fine-tuning the last layer. BERT has such noteworthy achievement because it learns a more powerful bi-directional representation than most of the previous approaches. BERT's architecture is mainly multi-layer bidirectional Transformer encoder with bidirectional self-attention mechanism. The encoder of BERT is pre-trained with two tasks, “masked language model” (MLM) and Next Sentence Prediction. These two objectives help encoder to learn both left and right contextual of a word in the sentence and provides significant support for downstream tasks like question answering.

4.4 BERT-additional Layer

With just one additional output layer, the pre-trained BERT representations can create state-of-the-art models. It is natural to try a "deeper" neural network after the BERT instead of a single output layer. It is widely believed that deep models are able to extract better features than shallow models and hence, extra layers help in learning features. In the experiment, we have added one extra layer before the output layer and feed it to the ensembling experiment.

4.5 Ensembling

In modern Machine Learning, Ensembling Methods are extensively used to combine multiple learning algorithms, preferably from different model classes, into an aggregate model with better performance than any single model. Each of the BERT and BIDAF-based model we built make different predictions on probabilities of start and end positions, and therefore we use Ensembling Methods in hope of utilizing the information extracted from all models. Guided Random Search for Weighted Average Ensembling

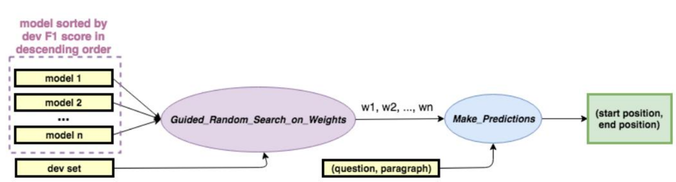

Figure 2: Pipeline for Guided Random Search + Weighted Average Ensembling

Assume that we have a total of n models to ensemble, and for each inputting (Question, Paragraph) pair, model k outputs ${}^{k}p_{start_i}, {}^{k}p_{end_i}, 0 \leq i \leq len(paragraph)$ where ${}^{k}p_{start_i}$ is the probability that the i-th position is the start position, and ${}^{k}p_{end_i}$ is the probability that the i-th position is the end position. Following this notation, ${}^{k}p_{ij} := P(start_pos = i, end_pos = j) = p_{start_i} \times p_{end_i}$predicted by the k-th model $p_{ij} := P(start_pos = i, end_pos = j) = \sum_k w_k \times p^k_{ij}, w \in R^n$predicted by the weighted average ensembling model $(start\_pos, end\_pos) = argmax_{(i,j)} p_{ij}$ So our goal is now reduced to finding the best $w \in R^n$. With this motivation, we develop a pipeline that learns the weights w, and make predictions on the prediction set. The details are described in Algorithm1. In high level, our algorithm randomly assign weights to each model, with the only restriction that a better model should never be assigned a lower weight than a model not as good. With the weights learned, we re-do the predictions by taking the weighted average of the probabilities predicted by each model.

Algorithm 1 Guided Random search + Weighted Average Ensembling

---

Input:
1. set of k models: $M \in R^k$
2. dev set: $\{(X_l, Y_l = (start\_pos, end\_pos)_l)\}_{l \in 1,2 \ldots n_{dev}}$
3. prediction set: $\{X_m\}_{m \in 1,2 \ldots n_{pred}}$
4. max_num_iter

Output: Predictions on the prediction set
$best\_weight = Guided\_Random\_Search\_on\_Weight$(M, max_num_iter, dev set)
ReturnMake_Predictions(M,best_weight, $predictionset$)

---

5 Experiments/Results/Discussion

5.1 Evaluation Method

We mainly use two types of evaluation metrics, Exact Match and F1 score.

Exact Match(EM) is a binary measure (i.e. true/false) of whether the system output matches the ground truth answer exactly. In our evaluation, EM stands for the percentage of outputs that match exactly with the ground truth.

F1 is the harmonic mean of precision and recall, more specifically:

$$F_1 = \frac{2 \times precision \times recall}{precision + recall}$$

$$precision = \frac{truepositives}{truepositives + falsepositives}\ ;\ recall = \frac{truepositives}{truepositives + falsenegatives}$$

For questions that do have answers, we take the maximum F1 and EM scores across the three human-provided answers for that question. And for those without answers, both the F1 and EM score are 1 if the model predicts no-answer, and 0 otherwise.

5.2 Baseline

First, we trained the baseline model and compared the loss, AvNA(Answer vs. No Answer), EM, and F1(official SQuAD evaluation metrics) for both train and dev sets. Over 3 million iterations we find that:

- The train loss continues to improve throughout
- The dev loss begins to rise around 2M iterations(overfitting)
- The dev AvNA reacches about 68, the dev F1 reachess about 60 and the dev EM score reaches around 57.
- Although the dev NLL improves throughout the training periodd, the dev EM and F1 scores initially get worse at the start of training, before then improving.

5.3 BERT

1. **Fine-tuning**: BERT is the first fine-tuning based representation model that achieves stat-of-the-art performance on a large suite of sentence-level and token-level tasks. For fine-tuning, most model hyperparameters are the same as in pre-training, with the exception of the batch

size, learning rate, and number of training epochs. Due to the issue of out of memory, when we change the batch size, we need to change the maximum sequence length accordingly. The dropout probability was always kept at 0.1. We visualize the loss curves of all BERT fine-tuning experiments below:

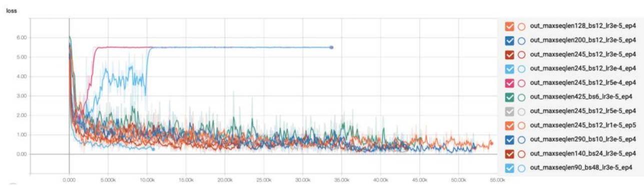

Figure 3: Learning curve of all BERT fine-tuning experiments

We can find that that with big learning rates in the scale of e^{-4} ($5e^{-4}$ and $3e^{-4}$), the learning curves spike after around 5k iterations, and the losses fail to converge. Moreover, when learning rate = $5e^{-5}$ or $3e^{-5}$, the losses converge the fastest. It turns out that when $max\_seq\_length = 245$, and $batch\_size = 12$, $learningrate = 3e^{-5}$the performance is the best, whose Dev F1 score achieved 77.166.

2. **L1 regularization**: In order to experiment the effect of L1 regularization, we have fixed the maximum sequence length to 140, batch size to 24, learning rate to $3e^{-5}$ and epoch to 4. By changing the L1 regularization parameter from $1e^{-4}$, $1e^{-3}$ to $1e^{-2}$

Table 1: different L1 regularization parameter result comparisom

L1 regularization parameter	Dev F1	Dev EM
$\lambda = 0$	74.679	71.915
$\lambda = 1e^{-4}$	75.705	73.001
$\lambda = 1e^{-3}$	76.666	73.824
$\lambda = 1e^{-2}$	76.76	73.955

As we can see from the table, when we applied L1 regularization and increasing the L1 regularization parameter, the performance becomes better.

5.4 Ensembling

We run the ensembling algorithm on all 26 models we have (BiDAF, SRU-BiDAF, fine-tuning BERT, BERT-additional layer, BERT-L1 regularization) and the performance of the best model is listed in Table 2.

Table 2: Ensembling Results

Ensembling Method	Dev F1	Dev EM	Test F1	Test EM
Guided Random Search for Weighted Average	79.944	77.081	78.841	76.010

6 Conclusion/Future Work

As we can see from SQuAD leaderboard, almost every leading submission uses BERT. In our report, a lot of methods we have tried are based on BERT, which can outperform even the best non pretrained contextual embeddings models. After training about 26 BiDAF-based and BERT-based models, and ensemble them with guided random search for weighted average algorithm, we can rank 30 with a relativily small dataset. For future work, we would combine the BERT and BiDAF together, which means that we replace BiDAF's GloVe word embedding with BERT last layer's output as as contextual word embedding. Hopefully we can improve our performance more with this idea.

References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[2] Tao Lei, Yu Zhang, and Yoav Artzi. Training rnns as fast as cnns. *arXiv preprint arXiv:1709.02755*, 2017.

[3] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*, 2018.

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

[5] Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 189–198, 2017.

[6] Adina Williams, Nikita Nangia, and Samuel R Bowman. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*, 2017.

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

Appendix I: Details of all models

Table 3: All models

ID	Experiment Name							Dev F1	Dev EM
lightgray	BERT Models	Pre-trained model	Number of Epoches	Learning Rate	Batch Size	Max Sequence length	Note		
1	out_maxseqlen245_bs12_lr3e-5_ep6l1	BERT-Base, Cased	6	$3e^{-5}$	12	245	$\lambda = 1e^{-4}$	77.206	73.478
2	out_maxseqlen245_bs12_lr3e-5_ep4	BERT-Base, Cased	4	$3e^{-5}$	12	245		77.166	73.643
3	out_maxseqlen140_bs24_lr3e-5_ep4_l1l1e-2	BERT-Base, Cased	4	$3e^{-5}$	24	140	$\lambda = 1e^{-2}$	76.76	73.955
4	out_maxseqlen140_bs24_lr3e-5_ep4_l1l1e-3	BERT-Base, Cased	4	$3e^{-5}$	24	140	$\lambda = 1e^{-3}$	76.666	73.824
5	out_maxseqlen245_bs12_lr3e-5_ep6	BERT-Base, Cased	6	$3e^{-5}$	12	245		75.925	72.343
6	out_maxseqlen140_bs24_lr3e-5_ep4_l1l1e-4_uncased	BERT-Base, Uncased	4	$3e^{-5}$	24	140	$\lambda = 1e^{-4}$	75.899	72.902
7	out_maxseqlen140_bs24_lr3e-5_ep4_l1l1e-4	BERT-Base, Cased	4	$3e^{-5}$	24	140	$\lambda = 1e^{-4}$	75.705	73.001
8	out_maxseqlen245_bs12_lr3e-5_ep4_l1l1e-4	BERT-Base, Cased	4	$3e^{-5}$	12	245	$\lambda = 1e^{-4}$	75.671	72.606
9	out_maxseqlen245_bs12_lr3e-5_ep5_l1+	BERT-Base, Cased	5	$3e^{-5}$	12	245	$\lambda = 1e^{-4}$, add one layer	75.354	71.685
10	out_maxseqlen245_bs12_lr3e-5_ep4_uncased	BERT-Base, Cased_uncased	4	$3e^{-5}$	12	245		75.071	71.372
11	out_maxseqlen140_bs24_lr3e-5_epoch4	BERT-Base, Cased	4	$3e^{-5}$	24	140		74.679	71.915
12	out_maxseqlen290_bs10_lr3e-5_epoch4	BERT-Base, Cased	4	$3e^{-5}$	10	290		74.633	71.372
13	out_maxseqlen245_bs12_lr5e-5_ep4	BERT-Base, Cased	4	$5e^{-5}$	12	245		74.546	71.092
14	out_maxseqlen200_bs12_lr3e-5_ep4	BERT-Base, Cased	4	$3e^{-5}$	12	200		74.356	71.241
15	out_maxseqlen245_bs12_lr1e-5_ep5	BERT-Base, Cased	5	$1e^{-5}$	12	245		73.885	70.829
16	out_maxseqlen400_bs6_lr3e-5_epoch4	BERT-Base, Cased	4	$3e^{-5}$	6	425		73.725	70.5
17	out_maxseqlen128_bs12_lr3e-5_ep4	BERT-Base, Cased	4	$3e^{-5}$	12	128		73.638	71.142
18	out_maxseqlen245_bs12_lr3e-5_ep4+	BERT-Base, Cased	4	$3e^{-5}$	12	245	add one layer	73.292	69.908
19	out_maxseqlen90_bs48_lr3e-5_ep4	BERT-Base, Cased	4	$3e^{-5}$	48	90		72.954	70.813
lightgray	BIDAF Models	Word Embeddings	Number of Epoches	Learning Rate	Encoder	Note			
20	baseline_sru	GloVe	30	0.5	SRU			64.08	
21	baseline	GloVe	30	0.5	LSTM	Baseline		61.508	57.99
lightgray	Ensembling Models								
21	Guidede Random Search for Weighted Average							79.944	77.081