A Deep Learning System for the Stanford Question Answering Dataset (SQuAD)

AmirMahdi Ahmadinejad
MSandE Department
Stanford University
ahmadi@stanford.edu

Abstract

SQuAD is a reading comprehension dataset. It means that given a paragraph and a question, our goal is to generate an answer by comprehending the paragraph. Being able to accurately perform this task shows that we have developed systems that can understand texts which is one of the fundamental goals of Natural Language Processing. In this project we use deep learning with additional ideas of Bidirectional Attention Flow [1] and Conditional Ending [2] to train a model over SQuAD that is able to deal with the task of question answering.

1 Introduction

Question answering (QA) problem is a fundamental and difficult task in natural language processing, since it requires both knowledge of language and knowledge of the world. In this project our goal is to design a deep learning model for QA over SQuAD [3].

1.1 Dataset and Problem Definition

We use SQuAD as our dataset. SQuAD has this feature that the answer exactly lies in some part of the given paragraph. Each record in this data set has three (character string) fields: "Context" (i.e. given paragraph), "Question", and "Answer" which exactly appear somewhere in the context. This data set is divided by the developers into three parts: "train", "dev", and "test". The first two are available to the public in order to train and fine-tune their models. The "test" set is private and is only used to evaluate and rank final models. Figure 1 shows an example of a question, its context, and its answer.

We use two measures to evaluate the accuracy of the models. i) EM is the exact match measure which is one if the predicted answer exactly matches the ground truth answer. F1 is the harmonic mean of precision and recall. For example if the answer to a question is "Microsoft Corp". If the model output is "Microsoft", then EM score is 0 but F1 is 1/2.

1.2 Related Work

Although the SQuAD is less than two years old there has been a lot of amazing approaches that researchers tried in order to beat this challenge. Here are the two that are used in this project:

- Bidirectional Attention Flow: This method uses attention technique to both attend from question to context and from context to question [1].
- Machine Comprehension Using Match-LSTM and Answer Pointer: Answer pointers are used to generate the end position conditioned on the start position[2].

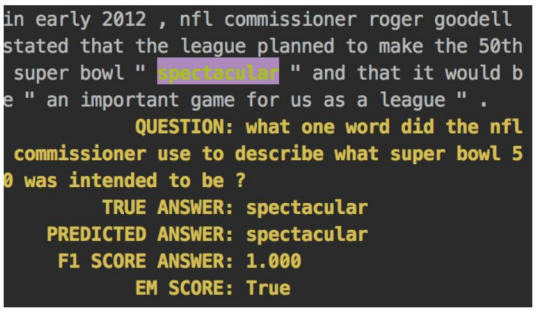


Figure 1: Question - Answer Sample

Here are a bunch of other interesting techniques that are used in this area:

- R-Net: This is a attention technique developed by Microsoft researchers which include a layer of self attention (context-to-context) in addition to the context-to-question layer [4].
- Chunk Extraction: Since the answer in SQuAD is basically a chunk of the context, this paper tries to predict the joint probability distribution of start and end position (or span of the chunks of text) [5].
- There are a lot more models built around question answering task and SQuAD [6, 7, 8, 9, 10].

2 Approach

The general architecture of my network can be seen in Figure 2. It is a simple version of the model provided by Minjoon et al. [1]. We use the basic framework layer provided by course staff and improved it by adding to enhanced model. First I replaced the basic attention layer with a bidirectional attention layer (BiDAF) [1]. Moreover, instead of using a independent distributions over start and end positions we use a conditional model to predict the end position based on the start position using an additional layer of RNN networks. The detailed description of the model is provided in the following subsection.

2.1 Model

RNN Encoder Layer

For a single record in the dataset, the context is represented by a sequence of d-dimensional word embeddings x_1, x_2, ..., x_T, and the question is also represented by a sequence of d-dimensional word embeddings q_1, q_2, ..., q_J. We use GLOVE [11] word embedding for this task. We then use a 1-layer bidirectional RNN with GRU/LSTM cell and feed the word embeddings to it to get hidden forward and backward states of context and question. Note that the weights for this layer is shared among question and context (See Figure 2). The context hidden state i is called $h_i = [\overrightarrow{h}_i; \overleftarrow{h_i}] \in R^{2h}$, and the question hidden state j is called $u_j = [\overrightarrow{u}_j; \overleftarrow{u_j}] \in R^{2h}$.

Attention Layer

We use Bidirectional Attention Flow (BidAF) as described in [1]. This method generate both context-to-question and question-to-context attentions. This layer is replaced with the BasicAttn layer provided in the Baseline model. The addition of Q2C attention helps to provide more information about the relevance of a part of context to the question. In what follows we describe in more details how this attention is computed[1].

To compute the bidirectional attentions, we first compute a similarity matrix $S \in \mathbb{R}^{T \times J}$, where S_{ij} provides a similarity score between h_i and u_j.

[1]The default project handout description is used for the formulas.

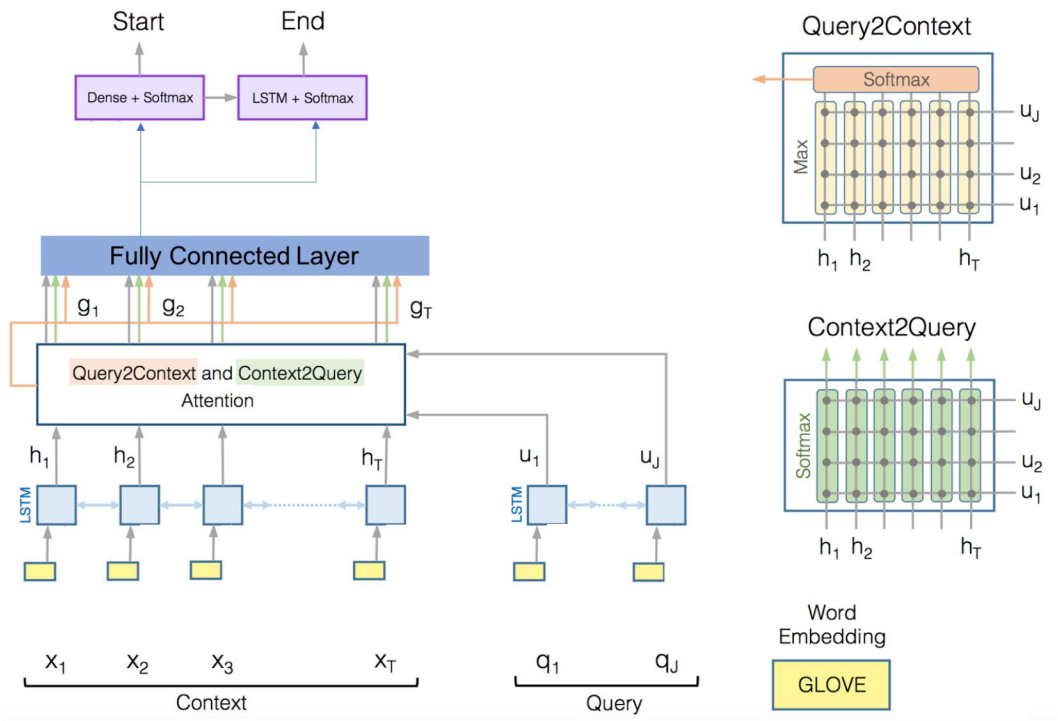


Figure 2: The model used for training is similar to [1]. It does not have the character-level CNN and the LSTM RNN layer after Attention layer is replaced with a fully connected layer.

$$S_{ij} = w_{sim}^{\top}[h_i; u_j; h_i \circ u_j] \in \mathbb{R}$$

Now to produce context-to-question attention (look at the green box in Figure 2), we generate softmax over the rows of the matrix S, and use the resulting probability distribution to take the convex combination of the questions hidden states u_j's. Specifically,

$$\alpha^i = softmax(S_{i,}) \in \mathbb{R}^J \quad \forall i \in \{1, \ldots, N\}$$
$$a_i = \sum_j \alpha_j^i u_j \in \mathbb{R}^{2h} \quad \forall i \in \{1, \ldots, N\}$$

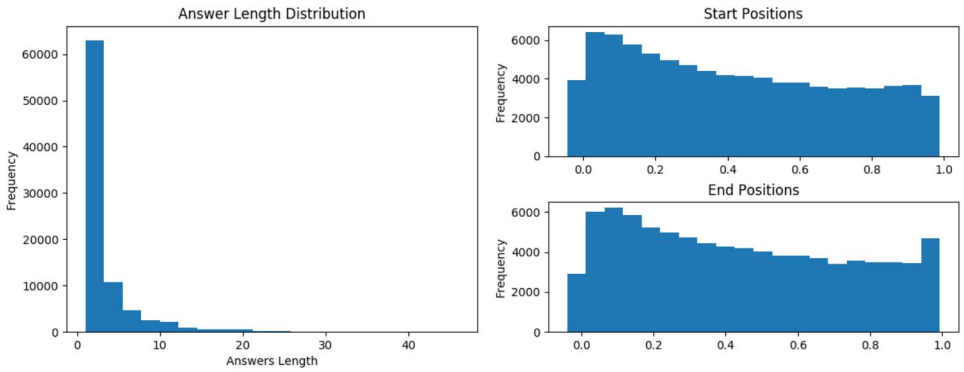
Note that β provides a probability distribution over context tokens, which is obtained based on importance of each context in the view of question (this is how we compute m).

To produce question-to-context attention (look at the orange box in Figure 2), we compute m_i, β, and h' as follows:

$$m_i = \max_j S_{ij} \in \mathbb{R} \quad \forall i \in \{1, \ldots, N\}$$
$$\beta = softmax(m) \in \mathbb{R}^T$$
$$h' = \sum_{i=1}^{T} \beta_i h_i$$

Finally the output of this layer for each context location i would be (See Figure 2):

$$g_i = [h_i; a_i; h_i \circ a_i; h_i \circ h']$$



(a) Answers length distribution (b) Start-end position distribution inside the text

Figure 3: Dataset Analysis

Output Layer

The output of attention layer is fed to a fully connected layer with a ReLU activation function as follows:

$$g'_i = ReLU(W g_i + b) \in \mathbb{R}^h$$

Where $W \in \mathbb{R}^{h \times 8h}$ and bias $b \in \mathbb{R}^h$. Then p^{start} the predicted probability distribution of the start position of the answer is computed as follows:

$$p^{start} = softmax(w_{start}^\top g'_i + b_{start})$$

To obtain p^{end} we do something different than the base model. We use the idea of conditional ending [1, 2] to produce p^{end} conditioned on p^{start}. To do this we use a recurrent neural network (RNN) with Long Short Term Memory (LSTM) cell. The input to each cell is $[g'_i; p_i^{start}]$. Let y_i be the hidden state of cell i. Then p^{end} is computed as follows:

$$p^{end} = softmax(w_{end}^\top y_i + b_{end})$$

Loss

Our loss function is the cross-entropy loss for start and end position which can be stated with the following formula:

$$loss = -\log p^{start}(i_{start}) - \log p^{end}(i_{end})$$

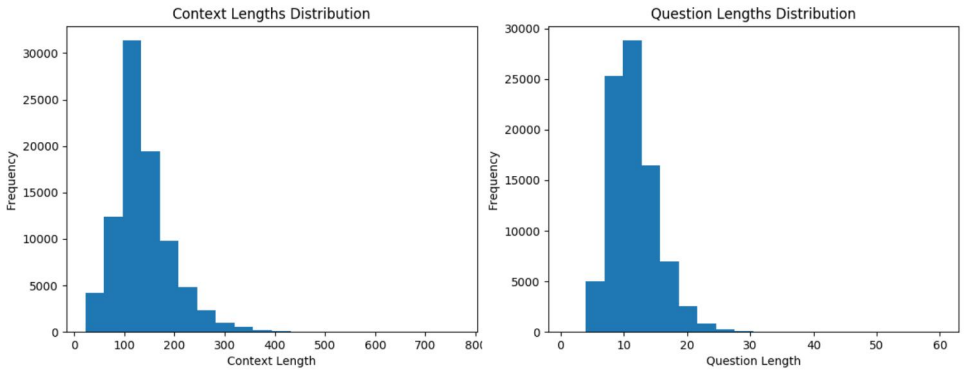
Where i_{start} and i_{end} are actual start and end positions of the answer.

3 Experiments

We use the basic framework provided by the course staff as baseline and develop it with the model specified above. We first do some analysis over the dataset to get an understanding over it and be able to set the parameters of the model accordingly. Moreover, we apply the improvements over baseline incrementally to be able to measure the effect of each improvement.

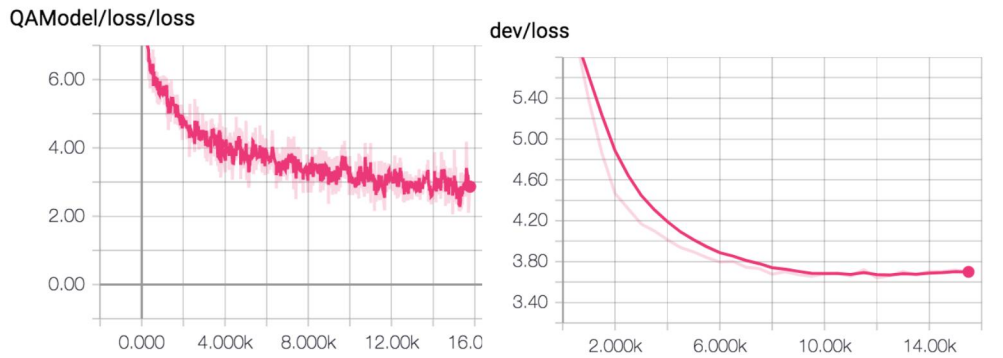
3.1 Dataset Properties

Figures 3, 4 show some statistics about the dataset. In Figure 3 we observe that most of the answers has length less than 20. Moreover, Figure 4 shows that most of the contexts has length less than 400. We used this observation to reduce the context_len in the baseline to 400 to increase the speed of training and scale to larger networks.

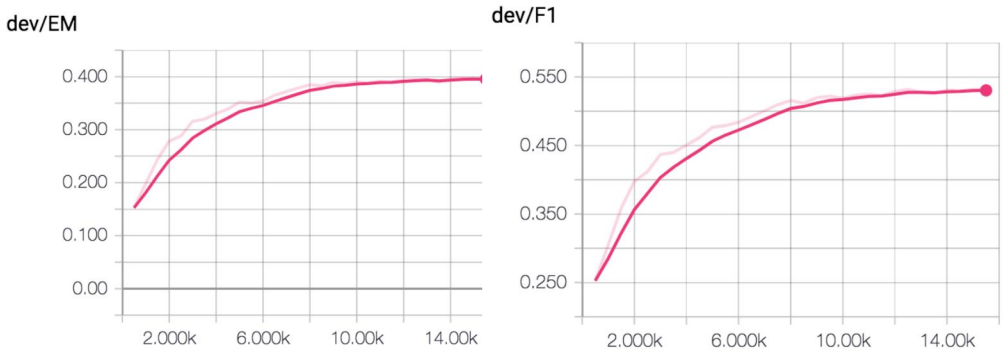


(a) Contexts length distribution (b) Contexts length distribution

Figure 4: Dataset Analysis - length of questions and contexts



(a) Train loss (b) Dev loss



(c) Dev EM score (d) Dev F1 score

Figure 5: Model Training for best model

3.2 Incremental Improvements and Fine Tunings

The final F1/EM scores of the models we built over this project are reported in Table 1. We take an incremental approach to measure the impact of every improvement:

- **Baseline**: This is the basic model provided by course staff, which includes a basic attention layer.
- **BiDaF-GRU**: This is the model which has improved attention layer with BiDaF. The RNN Encoder cell used for this model is GRU.

- **BiDaF-GRU1**: The difference with previous model is only in batch size and supported context length (which is reduced to 400). We see that the performance is almost similar but, we note that the training time is decreased with these parameters.
- **BiDaF-ConEnd-LSTM**: This is the model with conditional ending attached. The RNN Encoder cell used for this model is LSTM.
- **BiDaF-ConEnd-GRU**: The RNN Encoder cell used for this model is GRU.
- **BiDaF-ConEnd-GRU1**: The hidden layer size increased to 250.
- **BiDaF-ConEnd-GRU2**: The hidden layer size increased to 400, batch size of 70, and dropout of 0.2.

Figure 5 shows the progress of our best model in this project. This is BiDaF-ConEnd-GRU2 which achieves dev-F1=57.24 and dev-EM=46.59.

Model Name	Hid. Size	Bat. size	dropout	context length	F1	EM
BiDaF-GRU	200	100	.15	600	50.97	41.14
BiDaF-GRU1	200	80	.2	400	49.96	40.20
BiDaF-ConEnd-LSTM	200	100	.2	400	54.54	43.98
BiDaF-ConEnd-GRU	200	100	.2	400	55.25	44.56
BiDaF-ConEnd-GRU1	250	100	.2	400	55.72	45.2
BiDaF-ConEnd-GRU2	400	70	.25	400	57.24	46.59
Baseline	200	100	.15	600	39.813	31.173
BiDaf Paper[1]	-	-	-	-	77.3	68.0

Table 1: Model parameters and their F1/EM dev scores

4 Conclusion

The framework we used in this project can be used to build further complex network. Two improvements that seems pretty reasonable are:

- Character level CNN: Minjoon et al. [1] used this method along with the word embeddings to feed to the RNN Encoder.
- Joint prediction of start and end: It is possible to build the output layer so that it can provide the join probability distribution of start and end position.

References

[1] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, "Bidirectional attention flow for machine comprehension," *arXiv preprint arXiv:1611.01603*, 2016.

[2] S. Wang and J. Jiang, "Machine comprehension using match-lstm and answer pointer," *arXiv preprint arXiv:1608.07905*, 2016.

[3] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "Squad: 100,000+ questions for machine comprehension of text," *arXiv preprint arXiv:1606.05250*, 2016.

[4] W. Wang, N. Yang, F. Wei, B. Chang, and M. 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)*, vol. 1, 2017, pp. 189–198.

[5] Y. Yu, W. Zhang, K. Hasan, M. Yu, B. Xiang, and B. Zhou, "End-to-end answer chunk extraction and ranking for reading comprehension," *arXiv preprint arXiv:1610.09996*, 2016.

[6] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in neural information processing systems*, 2015, pp. 649–657.

[7] C. Xiong, V. Zhong, and R. Socher, "Dynamic coattention networks for question answering," *arXiv preprint arXiv:1611.01604*, 2016.

[8] D. Chen, A. Fisch, J. Weston, and A. Bordes, "Reading wikipedia to answer open-domain questions," *arXiv preprint arXiv:1704.00051*, 2017.

[9] X. Liu, Y. Shen, K. Duh, and J. Gao, "Stochastic answer networks for machine reading comprehension," *arXiv preprint arXiv:1712.03556*, 2017.

[10] B. Dhingra, H. Liu, Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Gated-attention readers for text comprehension," *arXiv preprint arXiv:1606.01549*, 2016.

[11] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.