Answering SQuAD
Codalab - faraz333

Atishay Jain
Department of Computer Science
Stanford University
Stanford, CA 94305
atishay@stanford.edu

Faraz Wasim
Department of Computer Science
Stanford University
Stanford, CA 94305
faraz333@stanford.edu

Abstract

Question Answering is a very important cornerstone of natural language processing where a machine should be able to understand human language to an extent that an individual could ask questions to the machine in the language he/she understands and then get the answers back in the same language without the involvement of experts.

In this project we evaluate a neural network based Question Answering system using a simple model and see how a very basic model performs on the Stanford Question Answering dataset.

1 Introduction

Question Answering is a fundamental problem in Natural Language Processing where we want the computer system to be able to understand a corpus of human knowledge and answer questions provided in Natural Language. Understanding human language enough to interpret provided questions and then to search for an appropriate answer is a cornerstone of artificial intelligence that the NLP community has been trying to achieve. Natural Language is very complicated because of a lot of missing information that we fill in from our experience to make the sentences faster to communicate.

The current state of the art QA systems provide a level of intelligence below that of an average middle grader. The SQAUD dataset and competition explores a subset of general question-answer challenge. The SQAD challenge act as good indicator of significant of deep learning architecture as all top performing submissions uses end to end deep learning architectures instead of feature syntactic/semantic extraction and classic machine learning. The purpose of our work is to explore different state of art deep learning architectures, which resulted in top submissions and apply ideas from these papers to build a simple question answer system. . Learning models based on deep neural network without sophisticated linguistic and semantic features are able to beat Pranav Rajpurkar benchmark of 51 by 25+ margins, which shows strength of deep neural networks. But it also introduced new challenges like dying gradients, slow learning, lot of hidden layers and choice of initial weights.

1.1 SQuAD dataset

Stanford Question Answering Dataset (SQuAD)[1] is a reading comprehension dataset consisting of over 100,000 question-answer pairs on 500 articles. The articles range from a variety of sources like Wikipedia and the answers are directly based on the contents of the passage and can be directly picked up from a portion of the passage. The context were chosen from Wikipedia articles and crows workers were encourages to use their own language to ask a question. The dataset is roughly portioned in 80k test set, 10k training set and 10k validation or dev set. SQuAD does not provide a list of answer choices for each question as compared to previous datasets. The dataset is varied in

terms of the length of the contexts, questions and answers as well as the type of reasoning required for solving the question.

One difference between SQUAD and close style queries is that answers to close style queries are single words or entities while answer in SQAD can be non-entities and can contain long phrase. SQUAD dataset uses distance in dependency trees to quantify answer by difficulty so it has collection of questions of varying difficulty. SQUAD dataset uses distance in dependency trees to quantify answer by difficulty so it has collection of questions of varying difficulty. The SQAD is tested against engineered features and linear regression. Their best model scores around 51 and can act as base line for neural network architecture. The current state of arts neural networks using ensemble models have scored result in 80+ range in F1 but they are behind human 91 match so we are still behind our goal of deep machines beating human performance. The benchmarked established by Pranav Rajpurkar using logistic regression and engineered features like dependency tree path features.

The squad dataset has a varied length of questions although most of them lie in the 5-15 word range. The answers almost all single word and therefore there may be cases where a model learns to predict only a single word and still achieve a significant score. The context length is more evenly distributed mostly between 20 - 250 word range. The variety of the context length, makes fitting to some piece of the context difficult which is a good characteristic of the dataset. The distribution of the dataset is shown in figure 1.

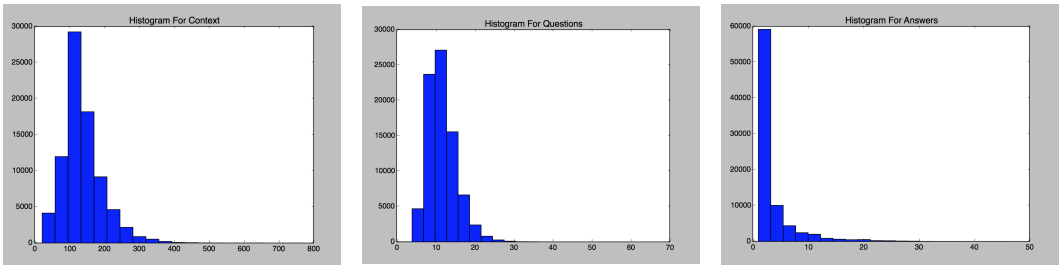

Figure 1: Context, Question and Answer length distributions in the SQuAD dataset

1.2 Traditional Approaches to QA

The traditional approaches to QA have been to build multiple layers parsers from syntactic to semantic for both question and answer and then combine the results from both the question and answer parsing to try and figure out the results. There have been separate algorithms based on the type of question and different approaches are taken to look for an answer based on the question. In these approaches each layer of parsing added its own set of losses which accumulated over a complete stack leading to subtle failures that required a lot of manual tweaking to get to the right results.

1.3 Neural Network based Approaches

Compared to the traditional approaches the neural network approaches are simpler. In the neural approaches the entire end to end system is divided into two layers - the encoder and the decoder. In the encoder phase the question and the context are parsed and merged together to produce a singe tensor(multi dimensional array) called attention. In the decoder phase, this attention vector is used calculate the answer.

One popular approach for solving question answers is Dynamic Coattention networks(DCN)[2]. In this approach, a coattention matrix is generation out of normalized attentions across the question and the context. Both the question and the context vectors are first encoded into a LSTM[3] and then after adding a sentinel node, the two matrices are multiplied to get the attention. This attention is normalized across rows and columns to get affinity for the question and answer. Then the document as well as the question is multiplied by this affinity so that we look into both of them in context of the affinity. This is then passed through a LSTM to get the encoded output.

In DCN, this encoded output is passed through a highway network[4] using maxout[5] as the activation function and the tanh non-linearity to create what DCN states is the highway maxout network. Separate networks are used for the start and end positions and multiple passes are created across the highway maxout network where each pass creates a new set of start and end position based on the previous pass and the consistent output is selected.

Another approach we looked closely is Bi-Directional Attention Flow for Machine Comprehension[6]. There are differences between DCN and Bi Directional attention flow. First it uses both word and character embeddings. It uses CNN for character embedding. Second it avoid early summarization of fused context and question by combining attention context and question before sending it to decoder layer. Also it uses 2-layer of biLSTM[7] in modeling layer in place of maxout networks of DCN. It does uses fusion query-to-context and context-2-query attention.

2 Our Approach

2.1 High Level Design

We use the traditional Encoder-Decoder Sequence to Sequence neural network to map the question answering task. We create word embeddings that are passed onto the encoder that generated an attention vector which the decoder decodes to produce the final output.

2.2 Preprocessing

The sentences are split by words and a dictionary is created where each word maps to the glove[8] vectors to create the word vector representation of the question as well as the context. We chose 100 dimensional glove embeddings. This data is then divided equally into 300 epochs and each epoch is further divided into batches of 10 each. Each forward pass is applied to a single batch the and gradients are applied back.

2.3 GloVes

This project make use of word embedding trained using gloves. The statistics of co-occurrence in corpus is primary source of all word vector models. Models like Skip gram[9] and CBOW[10] tries to capture co-occurrence of words one at time. While models like LSA tries to capture global co-occurrences. Models like LSA and PCA make efficient use of statistics and are faster to train but can not capture semantic and complex relationship beyond simple word similarity. Glove provides best of both words with faster training GloVe are scalable and captures semantic relationships just like Skip Grams and CBOW. We tried GloVes size of 50, 100 and 300 and finally settled down to 300 because of minor but consistent improvement in model.

2.4 Encoder

Our encoder(See Figure 2) is modeled after the DCN encoder where we have essentially removed the sentinel additions and also reduced the dimensions of the encoder. We experimented replacing the LSTM with a bidirectional LSTM which allowed us to have a smaller hidden layer size and achieve the same output. Since we had a lot of processing down the line, having a smaller hidden layer gave us a slight performance boost. The LSTM encoded question was additionally allowed a feed forward layer as in the DCN paper. This is essentially to model the fact that the question is small in size and therefore we would like to allow proper dissemination of information without the performance penalty of doing this over the larger context paragraph.

The affinity matrix is obtained just like in DCN and it is normalized across rows and columns following the DCN concepts to the final co-attention is achieved. This coattention is then passed on to the decoder.

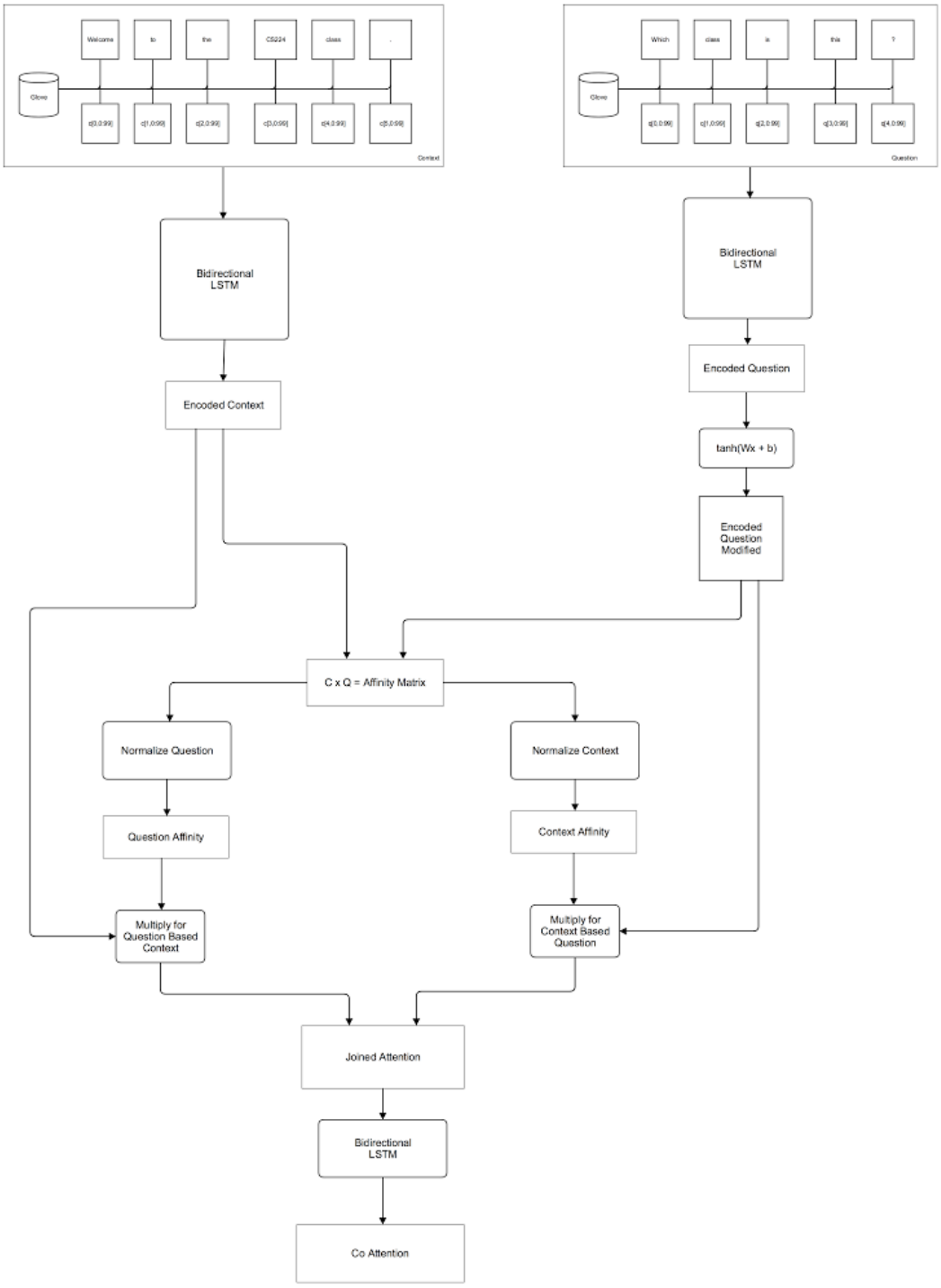

Figure 2: Encoder

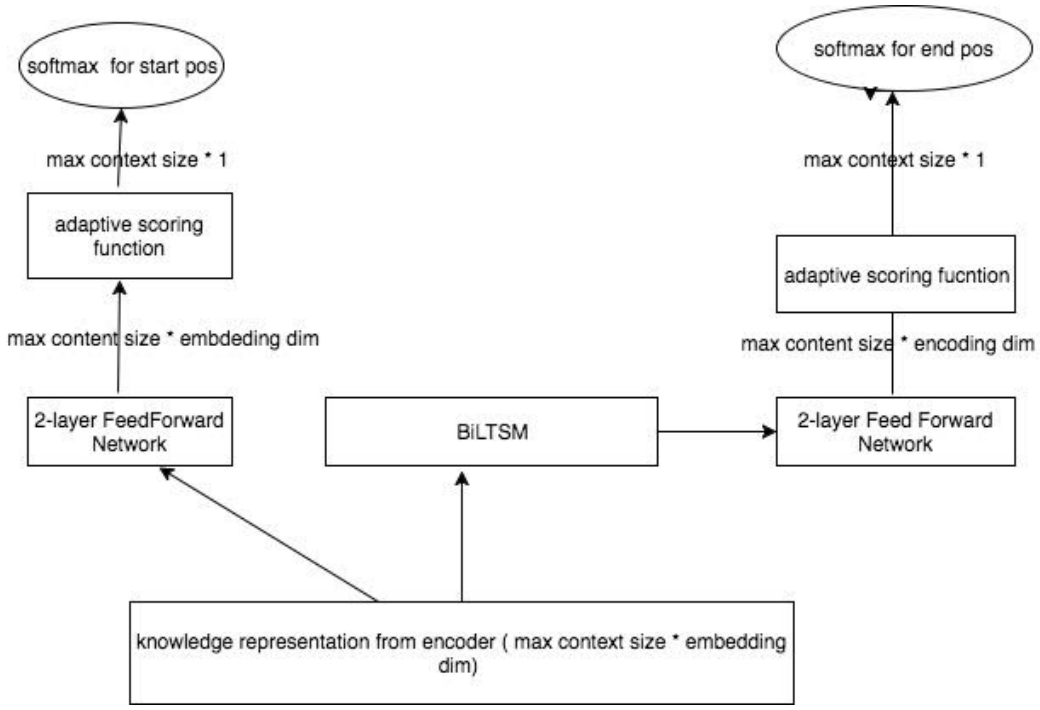

Figure 3: Decoder

2.5 Decoder

Our decoder (See Figure 3) is loosely inspired from ideas in DCN paper. It received knowledge representation from encoder layer. It then two different submodules to predict start and end symbols. Between start and end symbol there is LSTM. This LSTM tries to capture semantic/temporal relationship of words between start and end symbol. Both start and End symbol are then input to 2-layer feed forward networks.

The reason multi layer feed-forward was chosen because it is universal generalizer and we want to be capture different diversity in questions and context. In end it changes output of feed-forward network to adaptive scoring systems (with weights). The output of adaptive scoring system is then input to softmax classifier layer. It is important to mentioned that we did tried different decoder architectures like using increasing number of layers in feed forward network. Trying maxout highway networks as well trying out LSTM instead of feedforward network. We also tried simple $y = mx + b$ decoder with dynamic adaptive scoring system. The 2-layer feed-forward network yields highest performance with reasonable trainable time. We did played with hidden layer size as well.

2.6 Loss Calculation

We use an individual cross entropy loss with softmax activation functions for both the start and the end positions. This loss is then added up before we use the Adam optimizer to minimize the combined loss. We apply gradient clipping to loss calculation clipping every gradient to a value of 2. The learning rate is kept constant at 0.01.

2.7 Evaluation

We used the F1 and the EM score for evaluation. The F1 score measures the precision and the robustness of the classifier while the EM score gives the exact matches that the classifier correctly classified.

Table 1: Results (Leaderboard faraz333)

Set	F1	EM
Dev Set	9.31	.728
Test Set	9.743	0.892
Training	20	11

2.8 Validation & Hyperparameter Tuning

We tried various dimensions to the hidden layer from 256 to 8 vectors in this layer. The F1 scores fall off sharply after increasing the hidden layer size above 128 as they do below a hidden layer size of 16. We also played with varying the learning rate between 0.5 and 0.001

2.9 Variations in Approach

Once our end to end system start working we start trying different approaches from ideas from bi-attention network to DCN networks. We made use of feed forward networks and maxout highway networks in modeling layer decoder. We also played with embedding size for vectors and maximum size of context and question.

3 Results

The experimental scores are shown in Table 1. We ran many short epochs for evaluation and chose the best models after $\approx 150/300$ epochs and ran the selected model through the full dataset. While the leader-board shows $\approx 9\%$ F1 score, during training, it was fit to a smaller set to a F1 score of ≈ 20 and an EM score of ≈ 11.

4 Conclusion

The state of the art neural network models have been pushing the limits of what can be achieved by machines in the question answering tasks. Even basic neural models provide a great score in comparison to a random guess and fit the SQuAD dataset brilliantly.

That being said, SQuAD is far from being a real world dataset as the problem domain that is being tackled is very simple. A 99% SQuAD algorithm would still be a life changer for people looking up facts from a piece of text like wikipedia.

The epiphany that the world desires from the machine learning systems is still far away as we need more computer power as well as stronger models to understand the world to a level that an average 10 year old can understand.

Future Work

We can try use advance tensor tools like tensor debugger to dig deep inside code and can make more use of tensorboard to track gradients and activations in different layers. We will also experiment with sentence level representation and ensemble of models. Also if we have cluster of GPU, we can try hyper parameter search by building and trying hundreds of models with different combination of hyper-parameters and selecting best models. We will also like to dig deep into TensorFlow and understand internal mechanics specially how it can be deployed on large clusters so we can do hyper parameter tunning.

Contributions

Both the team partners had an almost equal contribution. Both spend enough time with the research papers and looking into the best known solutions to the given problem:

Atishay - Atishay did the initial setup of getting the code up and running and getting a baseline model where the F1 was 2-3%. This involved filling in the placeholder code, setting up the iteration batches and calculating the train time score, losses, save/restore functionality and understanding of the basic codebase including the GPU machine. Atishay also worked on the encoder and set it up modeling based on DCN to get to a co-attention that could be used in the decoder. He also worked on the loss function, and tuning the model with different hyper parameters.

Faraz - Faraz spent a huge amount of time with the decoder, in the hyper parameters and the Codalab submission. He also did the GloVe and the SQuAD analysis. He worked on tuning the model, swapping in the encoder and decoder with different options and then cleaning up the output so that it could be submitted. He also ran various passes on the GPU machine, got the train time scores to improve from 5-6% to 30-40% for the small batch we were practicing with.

Both had equal contributions to the writing part of the paper and each wrote in detail the piece that they worked on the most, including the diagrams. The poster was also a joint effort and Faraz took the extra effort to take a leave from work and show up for the session and present it to everyone.

We lacked the time and experience with tensorflow to tune the model to the extent, as to translate the wins in a small dataset to scale to the entire dev/train sets.

Acknowledgments

We are very grateful to CS-224N staff including Professors, Dr. Manning and Dr. Socher for a wonderful course, the TAs as well our colleagues on Piazza who posted and solved a lot of our doubts. We also want to thank our families for allowing us time to work on this project.

References

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

[2] C. Xiong, V. Zhong, and R. Socher, "Dynamic coattention networks for question answering," *CoRR*, vol. abs/1611.01604, 2016.

[3] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[4] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *arXiv preprint arXiv:1505.00387*, 2015.

[5] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, and Y. Bengio, "Maxout networks.,"

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

[7] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[8] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation.," in *EMNLP*, vol. 14, pp. 1532–1543, 2014.

[9] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[10] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.