An Ensemble Model for SQuAD

Yuze He
SUID: yuzehe
Stanford University
yuzehe@stanford.edu

Priyanka Dwivedi
SUID: pdwivedi
Stanford University
pdwivedi@stanford.edu
Coadalab submission: pdwivedi

Abstract

Machine Comprehension is a daunting task, since it requires cross-encoding and exchanging information between a context paragraph and a given query in order to produce an answer span. Recently some powerful models have come closer to attaining human accuracy on the Stanford Question Answering (SQuAD) dataset which consists of 100,000+ (context, question, answer) tuples extracted from Wikipidia. In this project, we implemented two models R-NET and BiDAF on the SQuAD dataset. We also experimented with replacing the RNN Encoder used in baseline with a CNN encoder. In the end, we combined the predictions from our best models to produce an ensemble model that further improves performance.

1 Introduction

Machine comprehension aims to understand a given context paragraph and predict the correct answer for a corresponding query. An accurate execution of a machine comprehension task has many important practical applications and therefore is an active area of research. The state-of-the art implementation often utilizes the RNN model or its variants (GRU or LSTM) to encode the context and the query. Attention mechanisms are useful in capturing the relationship between the context and the query, producing a query-aware encoding for the context. GloVe word embeddings and character embeddings are often used as model inputs.
In this project, we focused on two well performing SQuAD models - BiDAF and R-NET that have quite different attention mechanisms. We also tried to implement a CNN based context and query encoder. We combined predictions from R-NET and BiDAF to create an ensemble model that performs better than either of the single models. The rest of this report is organized in multiple sections: section 1 provides background and covers other related works; section 2 describes our modeling approach; section 3 covers the results from our experiments; we conclude in section 4.

1.1 Problem Definition

The objective in this project is to predict a span of words within the context paragraph that answers a given question.
Formal definition: Given a context paragraph represented by a sequence of d-dimensional word embeddings $C = c_1, c_2, ..., c_N \in R^d$ and a corresponding question represented by a sequence of d-dimensional word embeddings $Q = q_1, q_2, ..., q_M \in R^d$, predict a pair $A = (a_{Start}, a_{End}) \in R^2$ which are the index of the starting and ending position of the answer within the context paragraph.
The SQuAD dataset further constrains answer A to be a continuous sub-span of context C. Answer A often includes non-entities and can be much longer phrases. This setup challenges us to understand and reason about both the question and passage in order to infer the answer.

1.2 Related work

In this assignment, we focused mainly on two of the top encoder-decoder architecture models on the SQuAD[1] leaderboard. The first one we implemented is Bi-directional Attention Flow [5] which uses a bi-directional attention flow between context and question to obtain query aware context representation. It achieves an F1 score of 77.323 and EM score of 67.974 with a single model (81.525 and 73.744 with an ensemble). The second architecture is R-NET [2]. This model implements a Gated RNN for question passage matching and for context self matching and uses a pointer network for answer prediction. It achieves an F1 score of 86.536 and EM score of 79.901 with a single model (88.493 and 82.650 with an ensemble) [3].

2 Approach

Our ensemble model is consists of four main layers: embedding layer, encoder layer, attention layer and output layer. Each layer is described below.

2.1 Embeddings

The baseline model is implemented on Global Vectors for Word Representations (GloVe)[4], which are a set of pre-trained word embeddings on aggregated global word-word co-occurrence statistics from a corpus. We also tested a character level CNN module which can be used to deal with out of vocabulary (OOV) words and capture subword information. In this module we first padded all words to a fix length of 16 characters and then as described in [5] map each word to a vector space using character level CNNs. We initialized a trainable character embedding matrix using random initialization. As suggested in the project handout, we used a kernel size of 5 and filter depth of 100 to perform 1D convolutions. The resulting character level word embeddings is concatenated with the pretrained Glove Embeddings to get a hybrid description for each word. The concatenation of the character and word embedding vectors is passed through a two-layer Highway Network as described in the BiDAF paper [5]. The outputs of the Highway Network are two matrices, one for context and the other for query.

2.2 Context and Question Encoder

The baseline model uses the input from the previous embedding layer and implements a bi-directional GRU (with shared weights) to create an encoded representation for the context and the question. This encoding allows each word to become aware of words before and after it. Recently CNNs have been used quite successfuly for text classification [6]. However we didn't find any high performing SQuAD model use a CNN based architecture. CNNs have an advantage of RNNs in that they are faster and parallelizable. We tested building a CNN Encoder for context and query and replacing it with a RNN Encoder. Our model is described below.

The CNN layer is implemented on the word embeddings from the previous layer. It consists of K convolution steps each with an increasing kernal size from 2 to K+1 and a fixed filter size f. For this experiment we chose a filter size f of 50 and K was 6. We performed 1D convolution, set stride as 1 and used same padding to ensure output at each step is the same dimension as input. We applied non-linearity and dropout on the output of each convolution step and then passed the output as the input to the next convolution step with a bigger kernal size. Finally the output from all the convolution steps was concatenated to give a vector of the size of context or query and depth R^{f*K}. CNNs for text classification typical do maxpooling to create a single vector for each context. In this case we wanted the final output to be per word instead of per context, hence we didn't perform max pooling.

The intuition behind this architecture was replicating the use of CNNs for text classification with some modifications to better suite the task at hand. We used kernel sizes from 2-7 since our analysis had shown that answer length is typically in this range. This allows each word to interact with words in its immediate neighbourhood and also slightly far away. The filter size of 50 was chosen so that the output dimension from CNN encoder was the same as from our RNN encoder.

2.3 Attention

The attention layer takes the encoded context and query output from the previous layer as input and produces query-aware representations of the context words. Attention mechanisms attempt to focus on the most relevant segments from the context with regard to the query. We have tested two additional attention mechanism in this project.

2.3.1 Bi-Directional Attention Flow

The Bi-Directional Attention Flow (BiDAF) has both a context-to-query component as well as a query-to-context component, providing additional information to the model. It does not summarize the context or the question into a single fixed vector representation, instead attention is computed for each time step which reduces early summarization. BiDAF is also theorized to create a division of labor between the attention layer and modeling layer by its memory-less structure (i.e. at each time step attention depends only on current inputs). Such structure may force the attention layer to focus on learning the attention between the context and the question and leaves the task of learning interactions among query-aware representations to the modeling layer.(See Figure 1)
Specifically, given context hidden states $c_1, c_2, ..., c_N \in R^{2h}$ and question hidden states $q_1, q_2, ..., q_M \in R^{2h}$ from the embedding layer, construct a similarity matrix $S \in R^{N\times M}$. Each element $S_{ij} \in R$ is obtained such that $S_{ij} = w^T[c_i; q_j; c_i \circ q_j]$ where $w \in R^{6h}$ is a weight vector and $\circ$ is element-wise product of two vectors. This similarity matrix S is used to create the weights for both Context2Query attention and Query2Context attention.
Context2Query Attention: Context2Query attention focuses on question words that are most relevant for each context word. The Context2Query is a weighted sum of the question word representations for every context word, where the weights are $\alpha_i = \text{softmax}(S_{i:}) \in R^M$. Then Context2Query is $a_i = \sum_{j=1}^{M} \alpha_{ij} a_j \in R^{2h}\ \forall i$.
Query2Context Attention: On the other hand, Query2Context attention allocates more weight to context words that correlate more with the most similar question word. For each example, Query2Context is a weighted sum of the context word representations for each example. Let β =softmax($max_j\ S_{ij}$) $\in R^N$ be the weights for Query2Context. Then, Query2Context is defined as $c^{'} = \sum_{i=1}^{N} \beta_i c_i \in R^{2h}$. The final blended attention representations for the context are: $b_i = [c_i; a_i; c_i \circ a_i; c_i \circ c^{'}] \in R^{8h}$.
The Modeling Layer: A modeling layer is inserted to learn the interaction among the query-aware context word representations. A RNN model is used in this modeling layer.

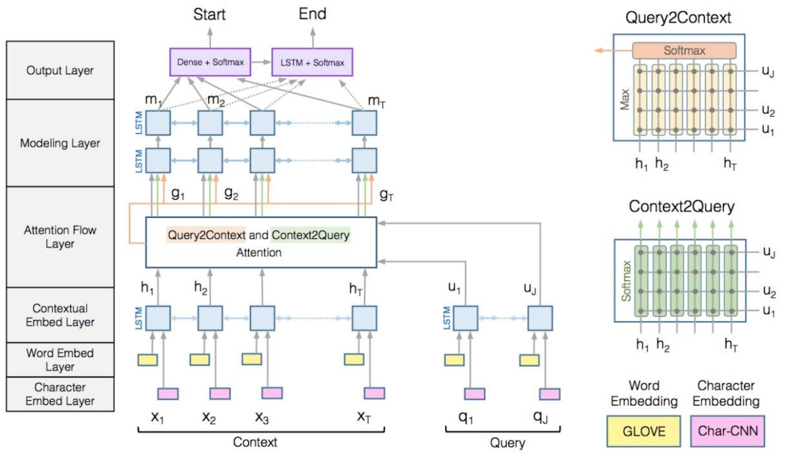

Figure 1: Bi-Directional Model

2.3.2 R-NET Gated Question Passage Matching and Self Matching

R-NET model uses a gated attention based recurrent network to incorporate question information into passage representation in the Question Passage Matching Layer. The output of this layer is a query aware context representation called v_P. It uses additive attention to combine information from context, question and v_P at the previous time step to create an attention pooling vector c_t. Additionally it applies a sigmoid gate on c_t and the context vector to focus on only the most relevant parts of the context. Finally it applies an RNN on v_P from previous time step and the output of the sigmoid gate to calculate the v_P at current time step. R-NET attention module has several differences over the baseline attention module - 1) It uses additive attention instead of dot product attention, 2) It uses a RNN to encode words in the passage with the attention weighted vector from previous time step and 3) It uses a sigmoid gate to assign different levels of importance to context parts depending on the question.

R-Net also performs self attention on the question aware context representation described above to aggregate evidence from the entire context to infer the answer. The output of this layer is a self matching vector h_P. The design of this layer is very simiar to the question passage matching layer and it also uses a sigmoid gate to assign different levels of importance to parts of context. Both these layers are shown in Figure 2. We implemented both the question-passage matching and the self-matching as described in the R-Net paper [2]. Finally we concatenated the vectors from the context encoder states, the question passage matching and the self matching to form blended reps that are passed to the output layer.

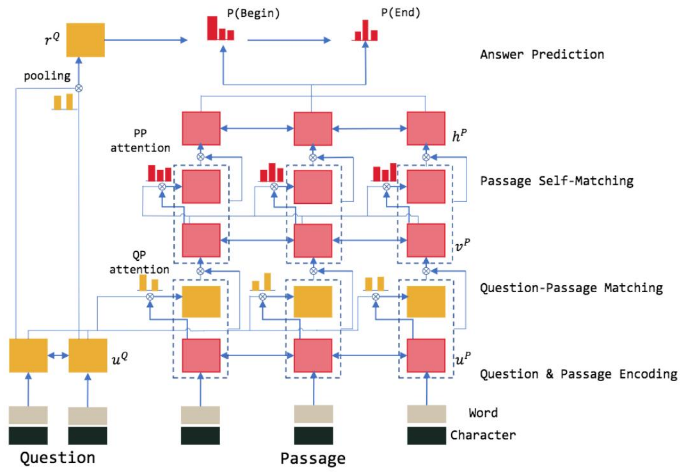

Figure 2: R-NET Model

2.4 Output Layer

In the baseline model the blended representations from the attention layer are fed through a fully connected layer followed by a RELU non linearity. Next we assign a score to each context location by passing it through a downprojecting linear layer. Finally we take a softmax of the logit distribution to identify the start and end indexes. In the baseline model the start and end index are identified independently and in the show examples mode we often found that the end index is before the start index in which case the model doesn't have an output. We implemented a smarter span selection at test time as suggested in the DrQA paper [7]. For each start index, i we looked at the end index, j in a window of 15 words after it and calculated the probability $pstart(i)pend(j)$.By iterating over the context length, we identified start and end index that maximized this product.
We also implemented an Answer Pointer for the output layer as suggested in the R-Net paper [2] and shown in Figure 2. R-Net model uses an attention-pooling over the question representation to generate the initial hidden vector for the pointer network. It then runs for two timesteps. In the first time step it applies attention mechanism on the blended representations as a pointer to select the

start position and on the second step, the same process is repeated with the output from the first step being used as an input to predict the end time step. Thus we condition the end prediction on the start prediction. We observed that conditioning end prediction on start prediction significantly reduced the number of cases where end index was predicted before the start index as in the baseline model.

2.5 Ensemble

Since we were able to get two different models working, we decided to ensemble them by combining their predictions. As stated in section 2.4 above, for each single model, we pick start and end index that maximize the product of the probabilities, $pstart(i)pend(j)$. When we create predictions.json for each model, we output both the predicted answer and the product of probabilities. For the ensemble model, for each question we chose the answer from the model that had the higher product probability associated with it. This led to an ensemble with better performance than either of the single models.

3 Experiments

3.1 Data

We are using the Stanford Question Answering Dataset (SQuAD) which consists of 100,000+ examples created using Wikipeida ariticles. Questions are generated by crowdworkers and answers are also manually created by quoting a segment from the context passage. Three answers are created for each question. The dataset is split into three subsets: train, dev and test. Our models are trained on train set and the hyper parameters are tuned using the dev set. The final model is submitted to Codalab, which evaluates the model on the test set. Figure 3 below shows the distribution of context length, question length and answer length over the training data. From the first plot we can see that majority of passages have context length under 300 so we reduced it to 300 to help remove the memory bottleneck. We kept the max question length to 30. The distribution of answer length plot indicates that about 75% of answers are less than or equal to 4 words long.

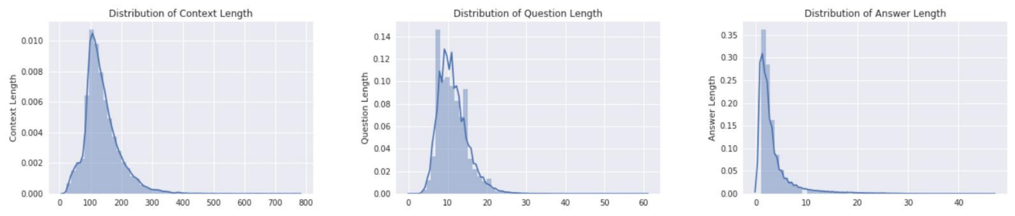

Figure 3: Distribution of context length, question length and answer length.

3.2 Character Embeddings

We initialized the Char-CNN model with random trainable weights and chose a kernel size of 5. Our results are included in the supplement material. We found that the Char CNN module didn't improve performance over baseline or BiDAF and made it overall slower to run. So we excluded this module from our final model. For future work, we would like to experiment with initializing the character CNN with pretrained embeddings.

3.3 CNN Encoder Layer

We experimented replacing the RNN Encoder in our BiDAF model with a CNN Encoder layer as described in section 2.2. Our results were encouraging. The CNN based model had lower performance than RNN based model but ran significantly faster. See results in the table below. The CNN Encoder BiDAF model ran twice as fast as the RNN Encoder BiDAF and 9 times faster than R-NET. Given more time, we would have experimented with including batch normalization and different values of dropout to further improve performance of this module.

Model Name	Dev F1	Time per example(milli seconds)
BiDAF with RNN Encoder	67.57	11.3
BiDAF with CNN Encoder	63.51	6.6
R-NET with RNN Encoder	65.65	63.45

3.4 Memory issues with R-NET Attention

We found that the R-NET model was very resource intensive and frequently led to out of memory errors on our GPU. This model initializes 9 new weight variables for performing additive attention on Question and Context and for self attention on the resulting vector from question-context attention. The original paper [2] states that the hidden size was set to 75 for all the layers. We found that the performance of R-NET wasn't very good if we reduced the hidden size to 75 for the encoder, attention layer and for the fully connected layer as the same variable was used throughout. Also we found that the memory issues were originating in the R-NET attention module so we kept the hidden size of encoder and the fully connected layers at 200, reduced the hidden size to 150 for the question-context attention and 50 for the self-matching attention layer. This led to a big improvement in the model performance but as a result our weights variables have different dimensions compared to the original paper.

3.5 Final Hyper Parameters

The table below shows the final configuration and hyper parameters on the R-NET and the BiDAF models.

Parameter	BiDAF Model	RNET Model
Embeddings	Pretrained Glove Vectors	Pretrained Glove Vectors
Encoder	Bi-Directional GRU	Bi-Directional GRU
Attention	BiDAF Q2C and C2Q attention with modeling layer	R-NET QP Matching and Self Matching
Output Layer	Softmax Output with smart span	R-NET Answer Pointer with smart span
Embedding Size	100	300
Learning Rate	0.001	0.001
Dropout	0.15	0.20

3.6 Main Results

We saw good performance from both BiDAF and R-NET models with BiDAF performing better than R-NET. The Ensemble model had better performance than either of the single models. See results from the dev leaderboard in the table below.

Table 1: Performance of BiDAF and R-NET on dev leaderboard

Model	**EM**	**F1**
BiDAF	63.43	73.95
R-NET	59.89	71.08
Ensemble	64.37	74.49

3.6.1 Performance comparison between BiDAF and RNET

We looked at the loss and F1 score of the models by num of epochs and it can be seen in Figure 4 below that the two models trained differently. The R-NET model was more memory intensive and trained slowly with dev loss still very high after the first epoch. On the other hand BiDAF trained relatively quickly and the loss converged after a few epochs.

We also compared the performance of these models on different question types and answer lengths as shown in Figure 5 below. The metric used for evaluation is Average EM accuracy. We saw that RNET model performed much better than BiDAF on questions starting with when and worser than BiDAF on questions starting with where. This builds on the intuition that an Ensemble model

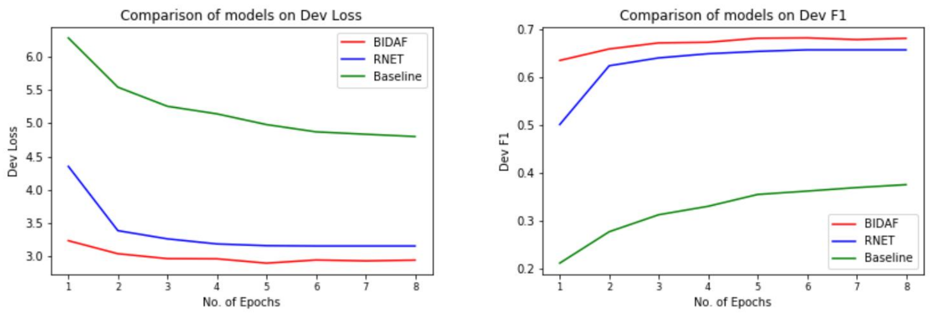

Figure 4: Comparison among baseline, BiDAF and R-Net models

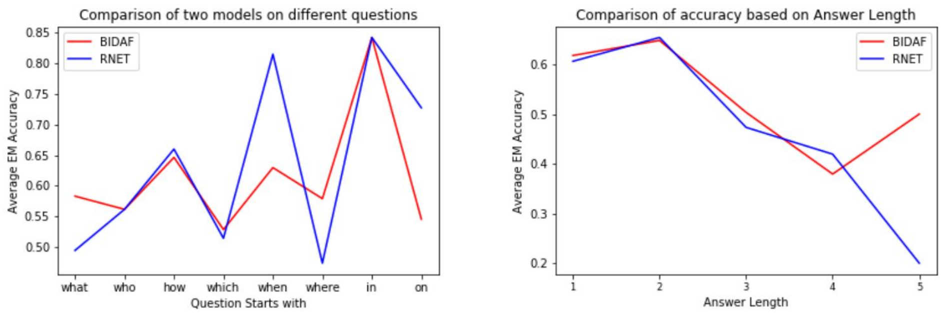

Figure 5: Comparison of BiDAF and RNET

would perform better than the single models. For both models we saw that performance decreases as the answer length increases though the decrease is sharper for RNET model. This highlights that attention mechanism in both models still has challenges in understanding longer sequences.

3.6.2 Qualitative Comparisons between BiDAF and R-NET

We also looked at a few cases where the two models outputted different answers and there was a significant gap in their confidence about their answer (the product of start index and end index probabilities). We have shared 1 case below and another in the supplementary materials section.

Case where BiDAF performed better than R-NET

Context: Westwood One will carry the game throughout North America, with Kevin Harlan as play-by-play announcer, Boomer Esiason and Dan Fouts as color analysts, and James Lofton and Mark Malone as sideline reporters. Jim Gray will anchor the pre-game and halftime coverage.

Question: Who will carry the game throughout all of North America? **True Answer:** Westwood One

Answer BiDAF: westwood one (probability of answer BiDAF: 0.9099)

Answer R-Net: kevin harlan (probability of answer RNET: 0.1871)

In this case R-NET model confused the person making play-by-play announcement with the person carrying the cup

Case where R-Net performed better than BiDAF

Context - There would be no more scoring in the third quarter, but early in the fourth, the Broncos drove to the Panthers 41-yard line. On the next play, Ealy knocked the ball out of Manning's hand as he was winding up for a pass, and then recovered it for Carolina on the 50-yard line. A 16-yard reception by Devin Funchess and a 12-yard run by Stewart then set up Gano's 39-yard field goal, cutting the Panthers deficit to one score at 16ˇ201310. The next three drives of the game would end in punts.

Question: Who had a 12-yard rush on this drive? **True Answer:** Stewart

Answer BiDAF: devin funchess (Probability of answer BIDAF: 0.2017)

Answer R-Net: stewart (Probability of answer RNET: 0.7403)

Comments - In this case BiDAF confused the player taking 16 yard reception with the one taking 12 yard run.

Both these examples show that if there are mutiple named entities together then even these state of the art machine comprehension models can have challenges understanding which entity performed which action.

4 Conclusion and Future Work

Our experiments indicate that both R-NET and BiDAF attention mechanisms improve the performance significantly over the baseline. The two models have different underlying attention mechanisms and different strengths so ensembling them results in an overall better model. For future work, we would like to try the following:

- Further improve the performance of Char CNN module by experimenting with initializing with pretrained weights and/or constructing a multi-layer CNN with different values of kernel size and filter size.
- A CNN based encoder is promising as it can materially reduce training time. We would like to try and increase the performance from this module by experimenting with different layer structure and parameters for kernel size and filter size, including batch normalization, increasing dropout etc. Our current BiDAF architecture uses a RNN modeling layer over the output from the attention layer. It would be interesting to see if we can replace this layer too with a CNN to have a complete CNN based architecture.
- In order to increase computational efficiency, the context paragraphs maybe sorted and then different context maximum lengths maybe used in our further experimentation. This would be more efficient than a single context maximum lengths and at the same time utilize all available data.

References

[1] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ Ques- tions for Machine Comprehension of Text. ArXiv e-prints, October 2016

[2] Natural Language Computing Group, Microsoft Research Asia. R-NET: Machine Reading Comprehension with Self Matching Networks

[3] Stanford NLP Group. The Stanford Question Answering Dataset, 2017. [Online; accessed 21-March-2017].

[4] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In Empirical Methods in Natural Language Processing (EMNLP), pages 15321543, 2014.

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

[6] Yoon Kim. Convolutional Neural Networks for Sentence Classification. arXiv:1408.5882 [cs.CL]

[7] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. arXiv preprint arXiv:1704.00051, 2017.