Attention-based Recurrent Neural Networks for Question Answering

Dapeng Hong
hongdp@stanford.edu
Codalab username: hongdp*

Billy Wan
xwan@stanford.edu
Codalab username: xwan

Abstract

Machine Comprehension (MC) of text is an important problem in Natural Language Processing (NLP) research, and the task of Question Answering (QA) is a major way of assessing MC outcomes. One QA dataset that has gained immense popularity recently is the Stanford Question Answering Dataset (SQuAD). Successful models for SQuAD have all involved the use of Recurrent Neural Network (RNN), and most of them apply the attention mechanism on top of the neural architecture. In this paper, we explore and compare two of such models - Match-LSTM and Bidirectional Attention Flow (BiDAF), and propose an ensemble model which combines these two models together. Our models are able to achieve competitive results on the CS224N Test Leaderboard.

1 Introduction

Machine Comprehension (MC) has always been an ultimate goal in the research of Natural Language Processing (NLP). The task is for the machine to be able to understand blocks of text and perform certain actions. One popular way to evaluate the outcome of MC is through Question Answering (QA). In this task, each instance is a question-context-answer triplet. The machine is given a question and a paragraph serving as the context of the question, and is asked to predict the answer based on its understanding of the question and context.

Recently, the human-written Stanford Question Answering Dataset (SQuAD)[1] with over 100,000 such question-context-answer triplets was released. SQuAD overcomes the weaknesses of previous MC and QA datasets, and is more realistic and challenging[2]. In this dataset, the answer is always a continuous sequence of words within the paragraph. Thus, the QA task can be formulated as predicting the start and end positions of the answer, given the question and context paragraph.

Traditional approaches to the QA task include pipelined NLP models involving upstream tasks like Named Entity Recognition (NER), syntactic and semantic analyses, etc. However, with the recent advance of neural machine learning, the top-performing models on SQuAD are all neural network-based end-to-end models. Many such neural architectures have been proposed, most of which involve the usage of the Long Short Term Memory (LSTM)[3] recurrent network and some form of attention mechanism so that the model can better focus on parts of the context related to the question. We explore and implement two of such models - MatchLSTM[4] and Bidirectional Attention Flow (BiDAF)[5] - with minor variations, and also propose an ensemble of them. Our models have achieved competitive results on the CS 224N leaderboard and better performance than the original SQuAD paper.

*This user's submission to the Test Leaderboard is our final submission.

2 Related work

Most of the state of the art results of the SQuAD dataset come from neural network-based models. As shown in Figure 1, these models all have in common a certain architecture: they encode question and context word embeddings using a bidirectional LSTM (BiLSTM), apply an attention mechanism to the encodings to obtain an attended contextual representation, and finally predict the probabilities of each word in the context being the start and end positions of the answer. The core of the model, and also where the difference lies, is in the attention and prediction layers. We briefly review these two layers of the MatchLSTM and BiDAF models below.

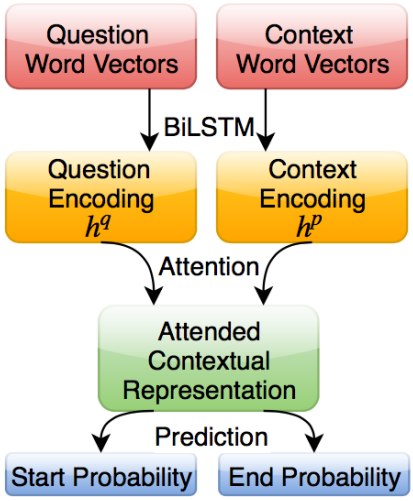

Figure 1: Common architecture of neural QA models.

2.1 Attention Mechanism

Attention Mechanism is extremely important for all successful QA systems because it allows the model to focus on parts of the context that matters the most for the question.

Match-LSTM, as the name suggests, uses a match-LSTM layer to calculate attention. At each context position i, word-by-word attention is calculated in a match-LSTM cell, which is a wrapper around the normal LSTM cell. Specifically, the attention score vector is calculated as follows:

$$\alpha = \text{softmax}(\boldsymbol{w}^{\text{T}}(\tanh(\boldsymbol{W}^q\boldsymbol{H}^q + (\boldsymbol{W}^p\boldsymbol{h}_i^p + \boldsymbol{W}^r\boldsymbol{h}_{i-1}^r + \boldsymbol{b}^p)) + b)$$

where $\boldsymbol{H}^q$ is all the question word encodings (or hidden states), $\boldsymbol{h}_i^p$ is the current context word encoding, $\boldsymbol{h}_{i-1}^r$ is the output of the previous matchLSTM cell, $\boldsymbol{w}, \boldsymbol{W}^q, \boldsymbol{W}^p, \boldsymbol{W}^r$ are all trainable weights, and $\boldsymbol{b}^p, b$ are biases. The match-LSTM is applied in both directions, and the results are concatenated to form the attended contextual representation. This is one example of dynamic attention because the attention score not only depends on the current hidden states, but also the previous attended hidden states.

BiDAF presents a different way of calculating attention, namely static attention. The attention score for all pairs of question word and context word is calculated upfront using only the question and context hidden states from the BiLSTM layer, according to the following equation:

$$\alpha(h^p, h^q) = \boldsymbol{w}^{\text{T}}(\boldsymbol{h}^p; \boldsymbol{h}^q; \boldsymbol{h}^p \circ \boldsymbol{h}^q)$$

where $\boldsymbol{w}$ is a trainable weight vector, [;] is vector concatenation across row, and $\circ$ is element-wise multiplication. This score is used in both context-to-query (C2Q) attention and query-to-context (Q2C) attention, the results of which are concatenated as the attended contextual representation.

2.2 Prediction

Models also differ in the way of making the final start and end predictions. The Match-LSTM model uses a Pointer Net[6] layer, where a similar attention mechanism as in the match-LSTM layer is used again, this time with the attended contextual representation obtained from the match-LSTM layer as the input. This Pointer Net layer also allows the Match-LSTM model to predict whether each word in the context is part of the answer or not, though this is not required in our formulation of this task.

The BiDAF model, on the other hand, uses a simpler but still effective prediction method: the attended contextual representation is fed into 2 more layers of BiLSTM and 1 linear layer to make the start prediction, and another layer of BiLSTM and a linear layer is used to make the end prediction.

3 Approach

For all our models, we use the pre-trained 100-dimensional GloVe[7] vectors, Adam[8] optimizer, and standard softmax cross-entropy loss between the predicted start and end probabilities and true positions. Based on the distribution of question and context lengths in the training data, and considering the tradeoff between loss of information and training efficiency, we cap the question and context length at 30 and 300, respectively. Padding and masking are done accordingly during preprocessing. For evaluation, we use the official SQuAD metrics - F1 score and Exact Match (EM) score, and also adds a third metric called Score, which is the harmonic mean of F1 and EM. We start with a simple baseline model, and then implement both the MatchLSTM and BiDAF models with some variations detailed below, and finally create an ensemble model of Match-LSTM and BiDAF.

3.1 Baseline

For the baseline model, we use the simplest way to calculate attention: for each pair of question and context word encodings, we compute the dot product as the attention score. We then concatenate the attention vector for each context position with the context encodings to form the attended contextual representation. Then we pass this representation to another LSTM layer to predict the score of each position being start point and end point.

3.2 Match-LSTM

We use the exact same encoder as in the original paper. In the decoder, we use a slightly modified pointer network to decode the output of the Match-LSTM layer. The pointer network is similar to a normal decoder with attention applied to encoder's hidden states, except that its output is the attention score. We replaced the tanh-softmax attention score function in the original paper with bilinear similarity, which gives us a validation loss drop of 0.5 with the same hyperparameters.

3.3 BiDAF

We omit the 2 layers of Highway Network[9] used in the original BiDAF paper. The original purpose of Highway Network is to facilitate the training of very deep neural networks. The BiDAF model is not very deep, and the paper does not mention what benefit it provides. Based on empirical training evidence, we decide that the added training cost of additional parameters and hyperparameter tuning outweighs the potential benefit. We also choose not to use character embeddings, which, based on the ablation analysis, only brings down the performance by 2.7% F1 and 1.9% EM.

3.4 Ensemble

Based on the assumption that dynamic attention used by the MatchLSTM model and static attention used by the BiDAF model both have their own strengths and weaknesses, we propose a combination of the 2 models: the training happens separately in the same way as before until each model makes its own predictions, and we then take a weighted average of the 2 models' predictions as the final predictions. The weighting vector is trainable.

4 Experiments

4.1 Hyperparameter Search and Model Details

Our final model details are influenced by multiple experiments on hyperparameter tuning and ablation analyses. Figure 2 below shows the results of these experiments on hidden state size, dropout rate, and whether exponential masking is included performed on the BiDAF model.

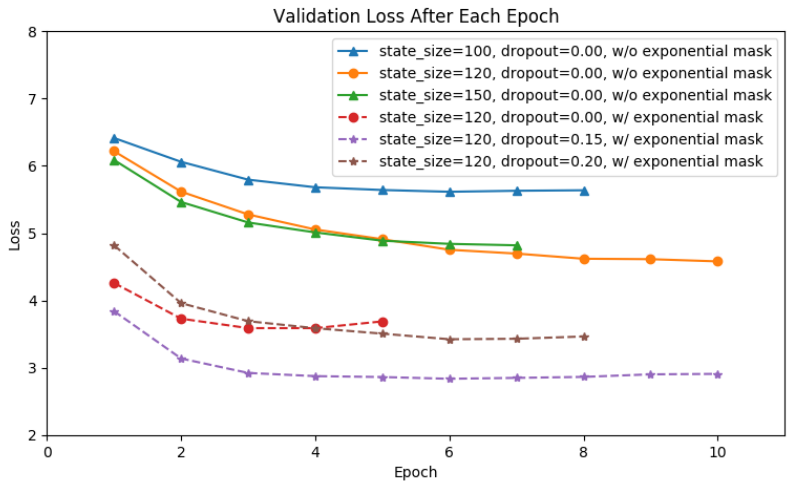

Figure 2: Training curves for different hyperparameters and model components.

After increasing the hidden state size from 100 to 120, we see a significant increase in the model's ability to learn - the model converges at a validation loss of $\sim$5 instead of 6 and converges faster, but we get diminishing returns and even some negative effect in later epochs if we keep increasing it to 150. Thus, we decide to use 120 as the state size for later models. The original BiDAF paper only uses a state size of 100, and we conjecture that we get better results with 120 partly because our model does not use character embeddings, and the model needs to be more expressive in some other ways, which in this case manifests itself in the hidden state size.

Regarding the effect of exponential masking, we see a dramatic increase in performance after incorporating it in our model. This is expected because in BiDAF, all attention scores are calculated upfront before applying the softmax function to normalize them for use in both C2Q and Q2C attention. Without exponential masking, the difference in values of the masked and unmasked elements would be skewed after taking the softmax, leading to inaccurate predictions.

A dropout rate of 0.15 instead of 0.2 also lead to a much better model performance and is thus chosen. Dropout is applied to all LSTM layers and the linear layers before the final predictions.

Our best BiDAF model is trained with a batch size of 20 for 10 epochs, and a learning rate of 0.001 with an annealing rate of 0.9 per 1500 steps. Gradients are clipped to a global norm of 5.0. We perform similar experiments on our MatchLSTM model, and the results are a learning rate of 0.001 with 0.95 decay every 2500 steps, and a dropout rate of 0.15. Gradients are clipped to a global norm of 10.0. Our ensemble model uses the same hyperparameters as the BiDAF model.

4.2 Results

Table 1 shows the best results we obtained on each model on the SQuAD dev set compared to the models in the original published papers.

Our best model (BiDAF) also achieves an F1 score of 63.63 and EM score of 49.80 on the test set. For both Match-LSTM and BiDAF models, our implementation does not achieve the same

Table 1: Results on the SQuAD dev set

	F1	EM	Score
Baseline	48.7	35.2	40.8
Match-LSTM (ours)	58.8	44.6	50.7
Match-LSTM (original)	71.2	61.1	65.8
BiDAF (ours)	62.8	48.6	54.8
BiDAF (original)	77.3	67.7	72.2
Ensemble (ours)	58.4	43.6	49.9

Table 2: Weights of the ensemble model

	BiDAF	MatchLSTM
Start	1.01	0.08
End	0.68	0.35

performance as the original implementations. This suggests more extensive hyperparameter search as well as other improvements to the models not mentioned in the paper.

Our ensemble model fails to outperform either of its components, which can be attributed to the simple and not very effective way of ensembling. As can be seen in Table 2, the final trained weights are very one-sided, especially for the start prediction, suggesting that a more sophisticated and organic way of combining the 2 models is needed. One possible way to better incorporate the models is to combine them at the attention layer, e.g. exploring a more advanced attention mechanism that has the strengths of both the dynamic match-LSTM layer and the static attention used by BiDAF.

4.3 Visualizations

Figure 3 shows the predicted start and end probabilities of the BiDAF model on one example of SQuAD. The question here is:"In early 2012, Goodell said that Super Bowl 50 would be what?" and the ground truth answer is "an important game for us as a league". The model correctly identifies, with the aid of attention, that what follows "it would be" is highly likely to be the answer, and that double quotation in general has a lower probability to be the start position than the actual word after it. It also correctly identifies the end of the sentence to be the end position of the answer, again assigning a lower probability to the quotation mark.

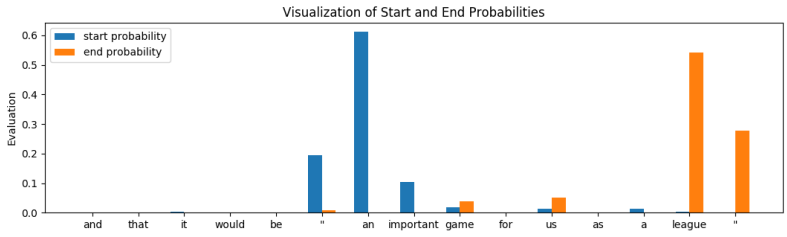

Figure 3: Example start and end predictions of BiDAF

Figure 4 shows the performance of BiDAF and Match-LSTM models grouped by question type. Consistent with the overall results, BiDAF outperforms Match-LSTM in almost all categories. The one category that Match-LSTM performs noticeably better is for the question type "why". This result is also consistent with the original BiDAF paper, where it shows that the model performs the worst on "why" questions. "Why" questions are innately more difficult than other types of questions because the answers to them tend to be longer. We conjecture that since the Match-LSTM model uses dynamic attention and takes into account the previous attended hidden state to calculate the next attention vector, it is naturally better at making longer predictions than BiDAF with static attention, where all attention weights are set in stone just based on the question and context encodings.

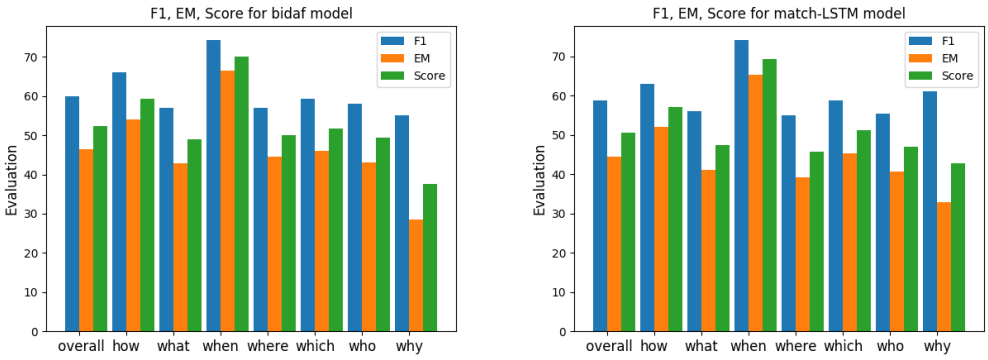

Figure 4: BiDAF and Match-LSTM performance by question type

5 Conclusion

In this project, we implement 3 QA models on the SQuAD dataset: a simple baseline, Match-LSTM, and BiDAF, and also propose a way to combine Match-LSTM and BiDAF. Although the performance of our implementations are not on par with what is in the literature, we achieve competitive results on the CS224N Test Leaderboard. The results visualization shows that the model is calculating and applying attention correctly, and the comparison visualization shows that each model has its own strengths and weaknesses. Time permitting, we would like to conduct a more extensive hyperparameter search and fine tune our models. We would also like to develop a more advanced way of incorporating the strengths of dynamic and static attention, so that Match-LSTM and BiDAF can be combined more meaningfully for a better outcome.

References

[1] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016.

[2] Zhiguo Wang, Haitao Mi, Wael Hamza, and Radu Florian. Multi-perspective context matching for machine comprehension. *CoRR*, abs/1612.04211, 2016.

[3] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

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

[5] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2016.

[6] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700, 2015.

[7] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[8] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[9] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *CoRR*, abs/1505.00387, 2015.