SQuAD Question Answering Problem : A match-lstm implementation

Philippe Fraisse
Department of Computer Science
Stanford University
California
philipp2@stanford.edu

Abstract

As a first experience in neural network modeling, and tensor flow usage, we try to implement a match-LSTM attention model, which is known for achieving interesting results on the SQuaD dataset leader board.

1 Introduction

1.1 Task comprehension

The task of machine comprehension is, in the most simple expression, in the SQuAD experiment, finding in a text what is related to a question about it.

There has been found multiple ways to achieve this task : we can use the syntactic structure of the texts. We can use the meaning of the words and look for semantic bridges. We can use named entities to capture explicit reference to location, person or time in the paragraphs.

Recent developments shed light on the very high potential of using deep neural networks for this task. The reason of this success seems to be explained by :

The fact that these models are end to end, such that the objective function directly translates the goal that we are trying to achieve. This is opposite numerous previous structures that uses several class of models connected together. Each of them is focusing on optimizing performance on their own task, which is indirectly related to the final goal.

A second reason could be because of more powerful machines to take over the task, as well as GPU computing which allows to parallelize much more the process and speed up computations. This is improved by the release of several high performance and very complete deep learning libraries such as Theano or TensorFlow, which allows us to focus more on the deep network architecture and methodology.

These features have proven to be very useful to make models more powerful at predicting the answer of a given pair of paragraph and question.

1.2 Data comprehension

The dataset is composed of 100 000+ question asked by crowd-workers on a set of selected wikipedia paragraphs. It has been asked to the crowd-workers to ask a question about the text and point the shortest part in the original paragraph that would be a correct answer.

The result is a set of 100 000+ triplets of paragraphs P_n, questions Q_n, and answers a_n.

We can see that the paragraph length is heavily skewed so we will need to truncate paragraphs, and make sure the padding and the masking is done correctly.

	Max	mean	Median
Paragraphs	766	138	126
Questions	60	11	11
Answers	46	3	2

Figure 1: statistics on the SQuAD dataset lengths

The questions can be classified by starting words :

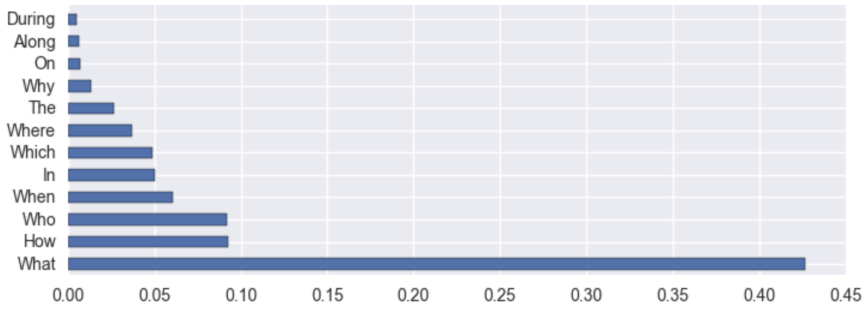

Figure 2: Proportion (truncated) of question by starting word

We can see that there is a majority of "what" questions. Unfortunately this type of question does not refer to an explicit entity name, so the effort put into these feature could have a lower impact due to the composition of the dataset.

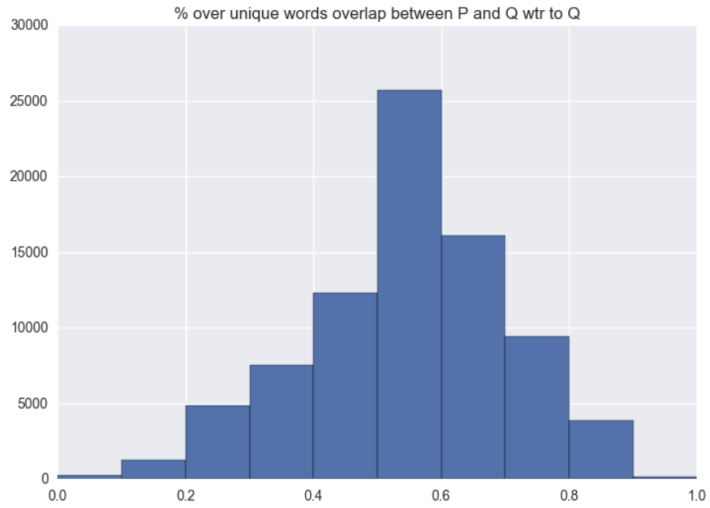

Figure 3: % of unique word overlap between the question and the answer, with respect to the question

We can see that there is quite an important word overlap between the paragraph and the question. This could mean that many questions are rephrased from the text, and thus, a relevancy matrix approach could bring useful information on how things are related to each other between the paragraph and the question, and thus focusing more effectively on parts of the paragraphs

2 Related work

At this point the readings about all the models that have been published on the squad dataset brings us the following insights :

+ Attention is an important contributor to the model's performance (Stanford Attentive Reader, MPCM, DCN), notably in reducing the negative impact of answer length on the models performance. The attention mechanism is used to reduce the quantity of information to look at when choosing which is relevant. It is a way to make a selection over all or some of the inputs regarding to the fact the model should make use of it or none.

+ Usage of a relevancy matrix can be an interesting way to make usage of the statistics on the data. The relative meaning proximity (embedding similarity) of every tokens in a paragraph with respect to every token in the question, and vice versa, seems to be a cheap and intuitive way to shed light over words that makes links between the paragraph and the question. (DCN, MPCM)

+ NER can play a role in some of the question answering since questions can be related to a PERS (who is ..) a LOC (were is ..), a DATE (when ..) etc.. .It seems that these entity-explicit cases represents nearly 40% of the dataset, but might not be the most complex one to predict whereas the "why" questions seems to be the more complex ones.

+ Computing the start-end words seems to be an easier task than predicting each words of the answer (Match-LSTM sequence vs boundary)

+ End to end models seems to have better performance (Stanford attentive reader)

3 The model

In this work, we will implement a version of the match-LSTM model developed by Wang and Jiang, as a first dive into the problem of machine comprehension, and the new tools like Google's tensorflow to achieve it.

This model principally consists in 3 parts :

The first part is a contextualization layer. The aim of this part is to contextualize the question and the paragraph regarding themselves. This way, a word is not only encoded by the meaning contained in its word vector representation, but also by the information about the words next to it. This results in a richer representation of each of the paragraph and question terms. At this point, the new information is added by the input itself, and there is no blending between inputs.

To achieve this, it is made use of long-short-term memory recurrent networks in the original paper. In this work, for speed implication during training, we will try to use gated recurrent units instead which have less parameters to train.

Let P be the paragraph word vector representations of dimension d, and Q the question word vector representations of dimension d. $P(d \times P)$, and $Q(d \times Q)$

$$H^p = GRU(P)$$
$$H^q = GRU(Q)$$

The resulting matrices are $H^p(l \times P)$ and $H^q(l \times Q)$ with l the size of the LSTM.

The second part is an attention layer, which is where the match-LSTM takes place. This attention layer takes the contextualized representation of the question in its whole (which is the concatenation of all its LSTM -respectively GRU in our version- hidden layers).For each term in the contextualized paragraph representation, a measure of how much they are related to the question is computed.

To achieve this, it is made used of a customized LSTM (resp. GRU) which is composed of a first blending element : G_i for the i-st term of the paragraph. This element is a tanh of a linear combination of the i-st term of the paragraph, the entire question and the preceding state of the LSTM network to which the final result will be input. This G_i matrix is then used to compute a weight vector α_i which is in turn used to weight the elements of the question, regarding to the i-st element of the paragraph.

This weighted question representation is concatenated with the enriched representation of the i-st term of the paragraph (h_i^p) (coming from the result of the first layer transformation). This will be the input to an LSTM, which will build a contextualized representation of these results for all i terms in the paragraph. So the matching is 2 fold : between the paragraph and the question, and between the result of this matching and all the preceding results.

$$G_i = tanh(W^q H^q + (W^p h_i^p + W^r h_{i-1}^r + b^p) \otimes e_Q$$

$$\alpha_i = softmax(w^T G_i + b \otimes e_Q)$$

z_i is then the concatenation of h_i^p and $H^q \alpha_i^T$

$$h_i^r = GRU(z_i, h_{t-1}^r)$$

This process is done in a forward and backward LSTM to have a bidirectional representation. The 2 resulting hidden layers for each paragraph terms are concatenated together to provide a matrix of size (P) x (2 * l)

H_f^r is the concatenation of all the hidden vectors in the forward GRU H_b^r is the concatenation of all the hidden vectors in the backward GRU H^r is the concatenation of the matrices above.

In the third layer : the answer pointer layer, the match LSTM principle is reused, but this time, only the preceding "attention added paragraph term representation" (H^r) is used. The second difference is that the weighting vector will be used as the probability distribution of all the terms of the paragraph, to be the k-ist word of the answer (for the sequence version of the model), or to be the first, or last word of the span (in the bounded version of the model). In this last case there are only 2 positions, and therefore, k will take only values 0 or 1.

$$F_k = tanh(VH^r + (W^a h_{k-1}^a + b^a) \otimes e_P$$

$$\beta_k = softmax(v^T F_k + c \otimes e_P)$$

With z_i the concatenation of h_i^p and $H^q \alpha_i^T$

$$h_k^a = GRU(H^r \beta_k^T, h_{k-1}^a)$$

Then in the bounded case that we have chosen, the probability of picking the right start word and the right end word is resulting from the combined probabilities from the $beta_k$ vector. Considering a to be the expected answer, it comes :

$$p(a|H^r) = p(a_s|H^r)p(a_e|a_s, H^r)$$

The model is then optimized by minimizing the cross-entropy :

$$-\log p(a|P, Q)$$

And summing all over the n observations in the dataset

4 Results

Our implementation has obtained an f1 score of 35.48 and exact match of 25.58 on the SQuAD test set leader board, which shows that there is a lot of room for improvement comparing to the results of the original paper.

By examining ours model's performance we can see that : The training loss function does not show significant anomaly, and there might not be a learning rate tuning problem:

The model is better at answering "when" questions, and quite bad answering "what" and "why" which requires more complex answers :

The performance on "what" question is especially critical since it represents half of the questions.

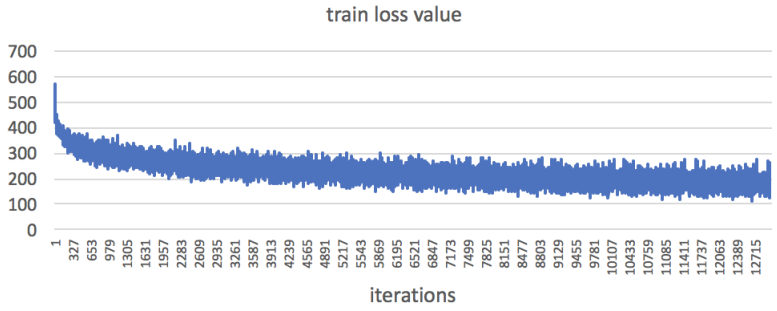

Figure 4: training loss curve

Question	em	f1	Proportion
What	18%	29%	52%
How	35%	43%	12%
Who	20%	32%	12%
When	51%	55%	8%
Which	20%	31%	5%
In	39%	46%	5%
Where	24%	36%	5%
Why	9%	28%	2%

Figure 5: performance by question types

By looking at the performance measure regarding to the length of the texts we can see that : A longer paragraph quite constantly decrease performance, which can be due to the difficulty to catch very long term memory :

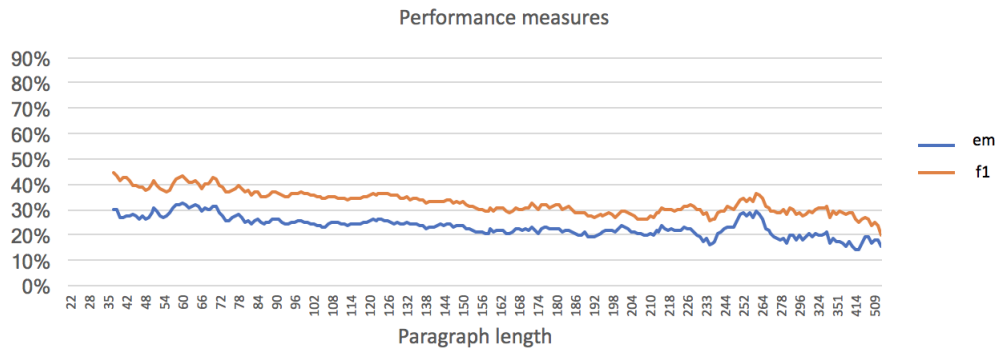

Figure 6: performance by length of paragraphs

The performance regarding to the question length tends to tell us that too few or too many information hurts. That might be because the questions are always taken as a whole, and there is no specific term by term attention for it.

The answer length is correlated with more error, which seems intuitive since more terms to find out also means more possibilities to make mistakes.

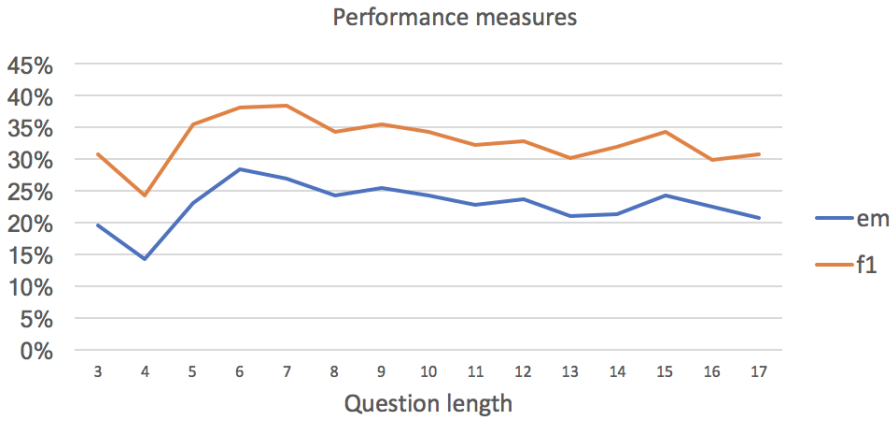

Figure 7: performance by length of questions

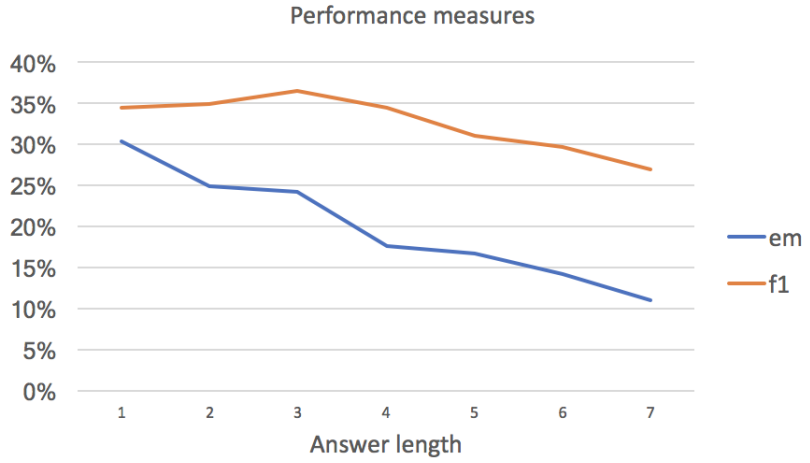

Figure 8: performance by length of questions

5 Insights

The analysis of the results may leads us to several explanations for the model's performance : Comparing to the original paper, our model was trained only on 100 dimensions hidden GRU layers, vs 150 to 300 in the original paper, which might have hurt the capacity of the model to capture all the complexity, especially for 'what" questions.

The use of GRU, might have hurt the performance since they might be less able to capture very long term memories than LSTM in these very long sequences.

The use of truncated back propagation trough time may have helped to avoid vanishing gradients over the very long paragraph sequences.

The use of a relevancy matrix could have been helpful to highlight global semantic links at the beginning of the network, in particular on explicit questions like "where" or "who".

The use of dropout has been tested without success since it seamed to hurt the training process quite importantly. The original paper did not use l2 regularization and did not mention usage of dropout.

6 Conclusion

The task of machine comprehension seems to have progressed rapidly due to new hardware, software and methodology approaches. Even if it seems hard to learn a machine to understand language, results on the SQuAD leaderboard shows performance that are getting closer to human understanding. Our implementation of the match LSTM model, needs some important improvements to get closer to these achievements, but was a great exercise to get started in deep natural language processing with tensorflow. This experience is a huge step forward from a starting point, and will be useful to build skills on getting the intuition of the architecture of these networks, and the methodology to to optimize their learning process.

Thank you for reading

References

[1] Danqi Chen, Jason Bolton & Christopher D. Manning (2016) A Thorough Examination of the CNN/Daily Mail Reading Comprehension Task.

[2] Shuohang Wang, Jing Jiang (2016) Machine comprehension using MATCH-LSTM and answer pointer

[3] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, Hananneh Hajishirzi (2017) Bi-directional attention flow for machine comprehension

[4] Caiming Xiong, Victor Zhong, Richard Socher (2017) Dynamic co-attention networks for question answering

[5] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, Hananneh Hajishirzi (2017) Bi-directional attention flow for machine comprehension

[6] Yang Yu, Wei Zhang, Kazi Hasan, Mo Yu, Bing Xiang, Bowen Zhou (2016) End-to-End Answer Chunk Extraction and Ranking for Reading Comprehension

[7] Zhilin Yang, Bhuwan Dhingra, Ye Yuan, Junjie Hu, William W. Cohen, Ruslan Salakhutdinov (2016) Words or characters? Fine grained gating for reading comprehension

[8] Kenton Lee et al. (2017) Learning recurrent span representatinos for extractive question answering

[9] Zhiguo Wang, Haitao Mi, Wael Hamza and Radu Florian (2016) Multi-Perspective Context Matching for Machine Comprehension

[10] Yelong Shen, Po-Sen Huang, Jianfeng Gao, Weizhu Chen (2016) ReasoNet: Learning to Stop Reading in Machine Comprehension

[11] Pranav Rajpurkar and Jian Zhang and Konstantin Lopyrev and Percy Liang (2016) SQuAD: 100,000+ Questions for Machine Comprehension of Text