Trying to build a Question Answering model with SQuAD 2.0

Boris BELLANGER
ENSAE
boris.bellanger@ensae.fr

Yann DE COSTER
ENSAE
yann.decoster@ensae.fr

Abstract

The aim of this project was to try to build a Question Answering model with the public SQuAD dataset using knowledge from the lectures, lab sessions, and the abundant literature covering this topic. (Spoiler alert : unfortunately we didn't succeed...)

1 Problem Framing

Imagine being able to ask an AI system questions so you can better understand any piece of text – like a class textbook, or a legal document... It would be a precious and time saving help ! This question answering task, which implies the ability to read text and then answer questions about it, is challenging as well as a central goal of NLP. One way of doing it is called Reading Comprehension, where the system tries to provide the correct answer to the query with a given context paragraph. This means that if the question is answerable, the answer is a chunk of text taken directly from the paragraph. The system doesn't have to generate entirely the answer. In order to build and train such systems, the most frequently used dataset is the Stanford Question Answering Dataset, abbreviate as SQuAD (Rajpurkar et al., 2016) [1]. It consists of around 150,000 questions posed by crowdworkers on a set of Wikipedia articles. Among them 50,000 question cannot be answered using the provided paragraph (Rajpurkar et al., 2018). SQuAD has led to many research papers and effective reading comprehension systems. Looking at the public leaderboard on the web page shows that the performance of many systems are even "surpassing" human performance !

[1]https://rajpurkar.github.io/SQuAD-explorer/

2 Experiments Protocol

We proceeded as follows. We downloaded the SQuAD dataset (a complex json file with multiple levels), and built a dataframe with it. Each row contains a context paragraph, a question, an answer, and the index of the character that starts the answer (remember the answer is a span from the context). We tokenized all these texts, and use the index of the word starting the answer. We used the answer text to compute also the index of the end of answer. We did so because we need our model to predict both the location of the start and the end of the answer in the context paragraph. We explored the dataset. We constructed a vocabulary using a pretrained embedding (Glove), and adding tokens for unknown/padding/end of question. Then we concatenated the question and the answer, such as to have a fixed size of 300 tokens. We designed a model as simple as possible consisting of a layer of embedding (to transform the sequence of token indices into a tensor of vectors), a layer of LSTM to incorporate temporal dependencies between timesteps of the embedding layer, and two layers : one to predict the start token, one to predict the end token, using a Softmax function. The aim was to produce two vector of probabilities corresponding to each token in the context paragraph : one vector being the probability that the answer starts at this position, and the other one the probably that the answer ends at this position. We then aim to train our model using cross-entropy as a loss function, in order to compute the errors between our predicted start and end positions with the true values.

3 Results

First, let's start with some descriptions on the SQuAD dataset. We established the length distribution of the contexts, the questions and the answers, which have a mean of 134, 11, and 3 tokens respectively (see graphs in Appendix). We use this knowledge to define the maximum length of our input (most of the paragraphs are under 280 tokens, and most of questions below 20, so we define 300 tokens as a maximum). We also looked at the most common words, bi-grams and tri-grams. We found that the most frequent bi-gram in the context paragraphs is ”United States” (1750+ occurences). More interestingly we distinguish the main types of question, and found that the vast majority consists of ”What” questions. (”what is the”, ”what was the”, ”in what year”, are the top 3 most frequent tri-grams for the questions)

4 Discussion/Conclusion

Regarding the most important part, the modelling one, we had issue running our Baseline model and managed to fix it very lately.. We couldn't train our model by lack of time. An important thing is that the loss function to minimize should be the sum of the cross-entropy for the start and the end locations (cumulative loss). We are not sure that it is what we were computing...

In order to improve our model, we spent countless hours looking at models using SQuAD on github repositories [2], [3], [4], but didn't want to simply copy-paste it. We particularly focus on the BiDAF model (Seo et al., 2017), and though we understand now its theoretical basis, we weren't able to implement it with Pytorch due to the complexity of the layers (3 layers of embedding for start, plus highway network...). One of the important component is a bidirectionnal attention layer. The main idea is that attention flow both ways – from the context to the question and from the question to the context. It is based on a similarity matrix containing the similarity for each pair of context and question embedding vectors. With that you can access to which query words are most relevant to each context word. And in the other way, which context words have the highest similarity to one of the query words and are hence critical for answering the question. We aimed to build a much more simple form of this architecture but didn't manage to.

Alternatively, faced with our inability to train a proper model, we used BertForQuestionAnswering with a pretrained model taken on Hugginface [5]. It allowed us to play a little bit around with the metrics specific to Question Answering. By doing so, we had the opportunity to get familiar with the Exact Match metric (EM) and the ROUGE metrics. EM is the the most ”severe”, since it gives either 1 or 0 point depending on the fact that the predicted answer is strictly equal to the true answer. In the second notebook, you can see how some regex can drastically improve the EM score (for example, getting rid of a punctuation). The ROUGE F1-score is an alternative that combines the precision (number of n-grams in the prediction matching the true answer, normalized by the number of n-grams in the reference) and the recall (number of n-grams in the prediction matching the true answer, normalized by the number of n-grams in the prediction). We tested the Bert model on a portion of the dataset and found quite good results.

5 Concluding remarks

We recognize that we might have been too ambitious when defining our project, but we were very enthusiastic about this field of research. We didn't anticipate the complexity of the task, even for elaborating a baseline model. As frustrating as it is to fail building our model, we still consider that we have learnt some interesting notions by reading papers and codes made by more experienced ”pytorchists”.

6 links

The two notebooks can be found here : https://github.com/ydecoster/Machine-Learning-for-Natural-Language-Processing-2022—ENSAE

References

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv:1606.05250v3*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hananneh Hajishirzi. 2017. Bi-directional attention flow for machine comprehension. *arXiv:1611.01603v6*.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. *arXiv:1806.03822v1*.

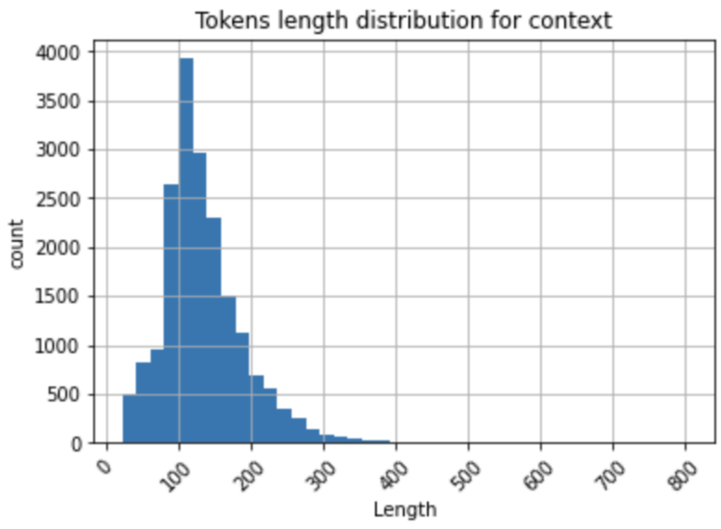
[2]https://github.com/galsang/BiDAF-pytorch

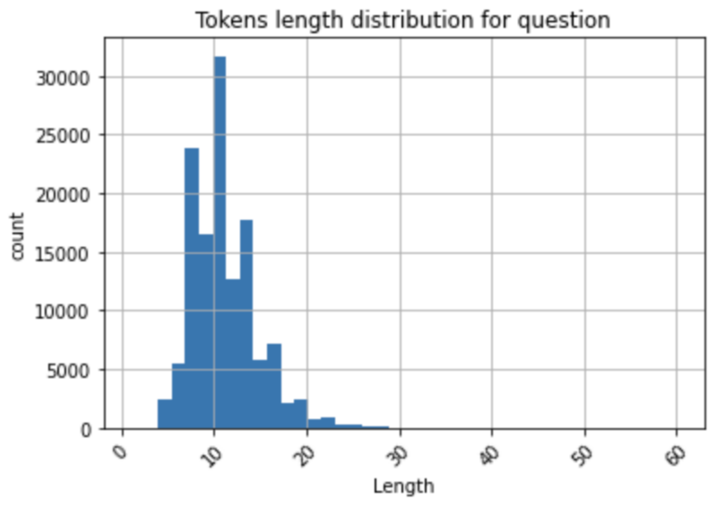
[3]https://github.com/kushalj001/pytorch-question-answering

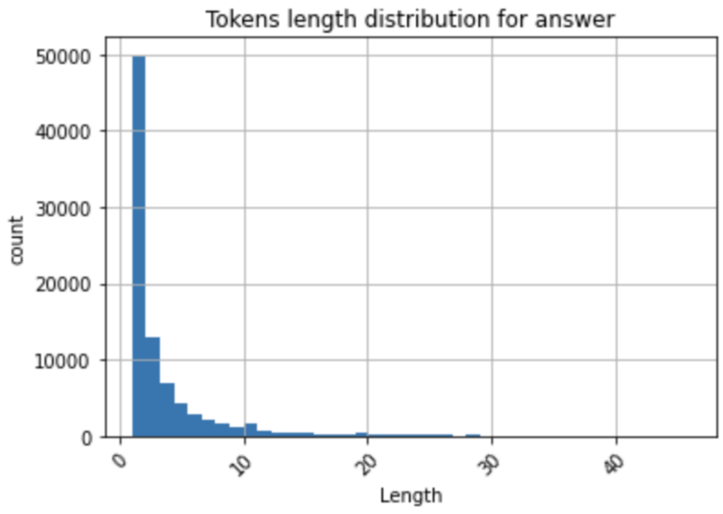
[4]https://github.com/michiyasunaga/squad

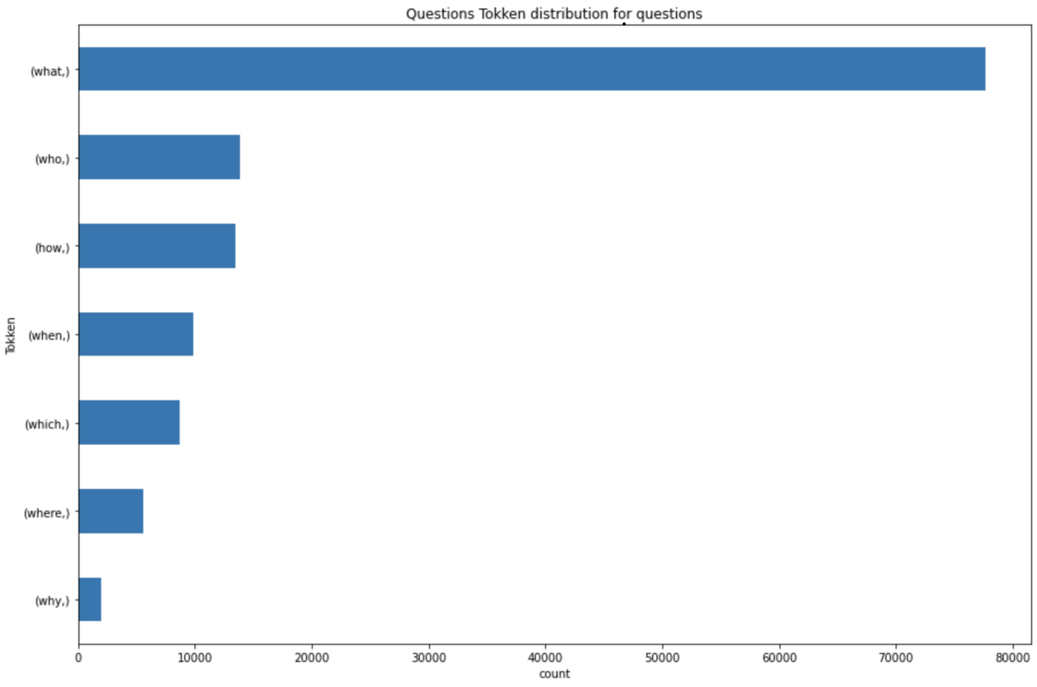
[5]https://huggingface.co/docs/transformers/model$_{doc/bert}$

A APPENDIX .


Tokens length distribution for context
count
Length
0
500
1000
1500
2000
2500
3000
3500
4000
0
100
200
300
400
500
600
700
800


Tokens length distribution for question
count
Length
0
5000
10000
15000
20000
25000
30000
0
10
20
30
40
50
60


Tokens length distribution for answer
count
Length
0
10000
20000
30000
40000
50000
0
10
20
30
40



Questions Tokken distribution for questions
Tokken
(what,)
(who,)
(how,)
(when,)
(which,)
(where,)
(why,)
0
10000
20000
30000
40000
50000
60000
70000
80000
count

BiDAF flow model, from (Seo et al., 2017)

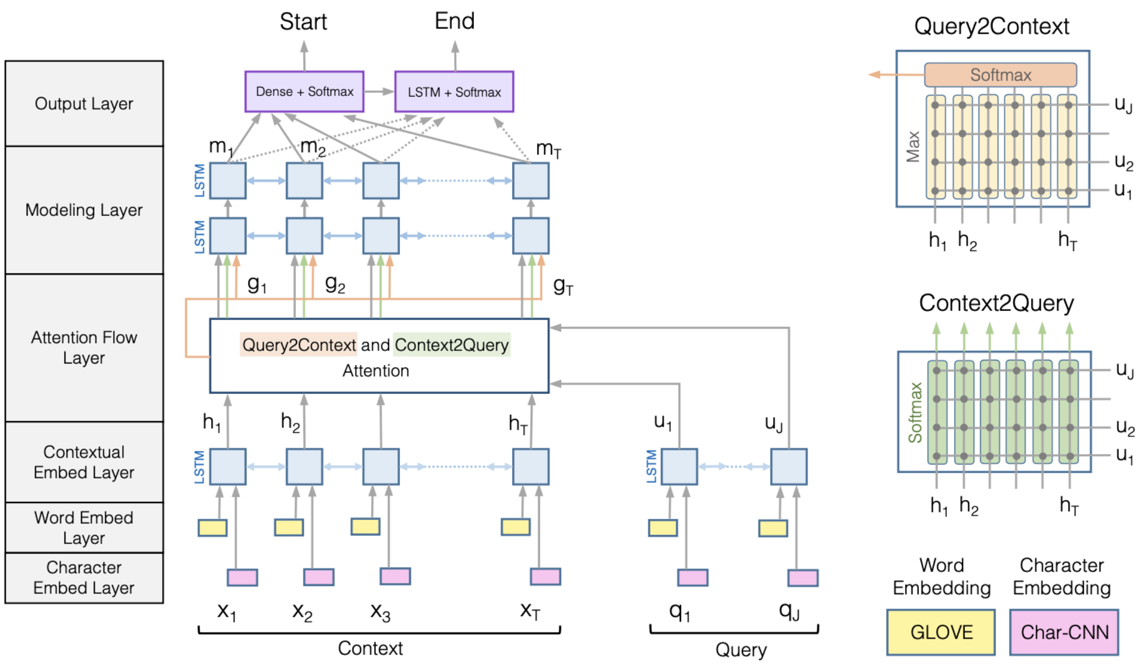


Figure 1: BiDirectional Attention Flow Model *(best viewed in color)*