Question Answering

George Smoc, Radu Ştefan Mihalache
Faculty of Computer Science, "Al.I. Cuza" University, Iaşi, Romania

Keywords: bert, tokens, RoBERTa, sliding window, caching

Abstract: Training and researching the use of pretrained BERT models for Question Answering with text span prediction - In this paper we will present our solution that scored 0.70415 on TensorFlow 2.0 Question Answering, having the score to be placed in the 4th place.

1 INTRODUCTION

Question Answering in NLQ consists of providing and answer based on a context and a question. There are different subtypes of question answering tasks:

- Multiple Choise Question answering - where the model has to choose the correct option from a list of possible answers.
- Span which answers the question - where the model has provide one or more pairs of (start index; end index). These indexes contain a span of the context that answers the question. For this problen, the context can be rather large, surpassing 6000 tokens. Standford Question Answering Dataset (SQuAD), represents an example of a dataset used for such a task.
- Language model generated answers - where a language model (usually a LLM) generates the answer based on the context.

In this paper we will present the model that we trained for the competition TensorFlow 2.0 Question Answering. This model has to provide answers about Wikipedia article based on real-user questions. This model is required to predict spans of the original article. There are several challenging aspects in this competition. Given that the context represents Wikipedia article, it may have a length of up to 6000 tokens. This makes the context size to big for many models that have under 70B parameters.

Another challenge is that a question may have either a short answer consisting of spans, a short answer consisting of Yes/No, or a long answer that may consist of an entire paragraph, and the model has to choose which answer it should provide.

2 RELATED WORK

LongFormer Information Gathering Span Bert

RoBERTa: It removes the MLM objective and trains on a larger dataset with more unidirectional data. It utilizes dynamic masking, where the masking pattern changes for each training epoch. It uses as separator $<s> and </s>$ and excludes Next Sentence Prediction (NSP) , because of empirical observations. In the context of our problem, training time increases for obtaining a better loss and also needs some extra processing, like using html tags as tokens recognized by tokenizer. Approaches in this direction include adding idioms as single tokens, using SentencePiece for tokenizing data from different languages.

Mistral7b: a classical usage that makes it usbale for our approach is classifying tweets based on their content. However, there is a clear difference at the level of params used: 7 billion parameters, which makes it hard to train and use in the current context.

3 SOLUTION

We use an approach based on Bert and a seq2seq arhitecture that is made for inputs and outputs of variable length. The data passed into the model is made out of 3 components: input_ids that are ids from the vocabulary corresponding to the tokens of the sequence, token_type_ids that marks whether the token is part of the question or part of the answer and attention_mask that indicates which tokens should be attented (having value 1) and tokens that should be ignored (marked with 0). The output of the bert has shape of batch_size x 512 x 1024 from which we do a mapping with 2 linears of sizes: (hidden_size, 2) that should give us a start and an end for the short answer and (hidden_size, 1) that should give us the long answer.

On the extracted results we compute the top k starts and top k ends, in our case k = 10, then generate all the possible pairs for them. After that, scores of the pairs (defined as the sum of the start and end values) are computed and another top is made to extract the best possible answers. Also, an edge case is included for questions that do not have an answer: if the score for the null token using the forumla described above is higher than all the other scores, an empty prediction is sent. In order to obtain better results, html tags were also included in the known tokens of the tokenizer and a custom library was created in order to pass the tokens at the very beginning of the tokens list for space efficiency. We also added special tokens for bert that can be understood from the general format of the data: [CLS] question [SEP] paragraph [SEP] Several caching methods and improvements were made for speed efficiency.

- Automatic Mixed Precision for Deep Learning that reduces training time by 4 times
- Caching of the very next documents to be iterated by the generators created on the input using a threadpool mechanism and numpy optimizations on getting the mapping from character to its associated token index
- Caching mechanism for generated doc spans that are used for handling large documents. The value used is 256 and the max sequence length is 512. For data variety documents with no long answer are used in dataset with a probability of 0.1
- Caching mechanism for the values distribution given by the model for a specific input
- Caching mechanism for pretrained BertModel and BertTokenizer

The optimizer used is AdamW that adds a regularization term to the loss function during optimization to penalize large weights. The precision of the results is computed usig F1 score

In addition, we used a tokenizer with lower_case=True and the model is uncased.

Figure 1 displays how BERT is trained to predict a label with bidirectional context, and Figure 2 shows how we connected Dense layers to the BERT output sequence in order to predict probabilities for start and end positions.

One important thing to mention is that BERT has a max context length of 512 tokens. Given that Wikipedia articles may have more that 6000 tokens, we used a scheme that splits the article context into multiple crops and appends the question to the crop. All the crop are fed through the model which outputs a probability distribution for start answer index, end

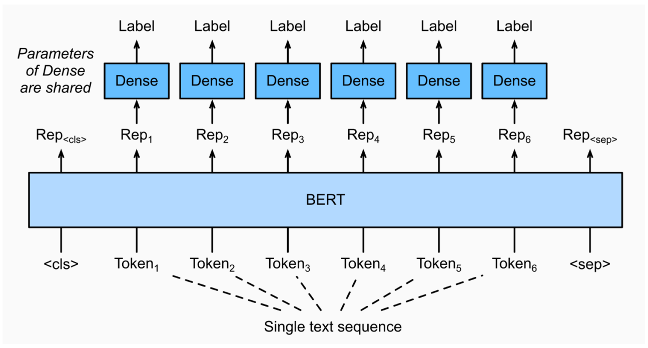


Figure 1: Proposed solution

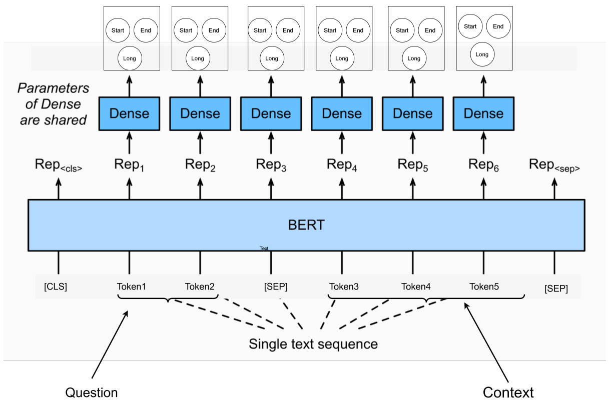


Figure 2: Proposed solution

answer index and long answer index. Probabilities form all the crops of an article are combined in order to provide the best response to the question.

4 RESULTS

The target for the model is to predict either a short answer, or a long answer. It is very important that FP rate and FN rate to be balanced for each type of answer and acrorss all submission types. Thta is why in this competition, F1 score is messured for both long answers and short answers and MicrF1 score is computed from both.

The compotition dataset consists of 1000000 insances of QA, messuring 20GB of data. The test dataset consists of 384 instances of QA, messuring 18MB.

Given the vast amount of data and the complexity of the task, we had to to use 10% of the data for training with 6 epochs.

In order to benchmark our model for validation we used 295 intances of that training dataset that was not included in our training process.

For test, we used the Kaggle submission score and we obtained 0.70415 . This score puts us on the

	Value
Micro F1 Score	0.735
Short Ans F1 Score	0.597
Long Ans F1 Score	0.829

Table 1: Results on validation split (computed by us)

	Value
Micro F1 Score	0.70415

Table 2: Results on test split (computed on kaggle)

4th place on the TensorFlow 2.0 Question Answering competition Leaderbord.

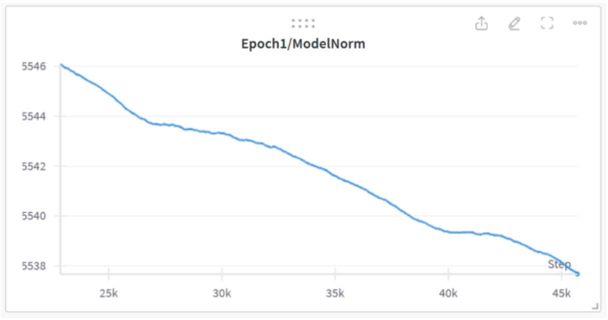


Figure 3: Model Norm Evolution on epoch 1

What influenced our results is the choice of tokenizer , the special tokens used by BERT (when making a comparison with first version - a RoBERTa model, a clear improvement could be observed) and the custom tokens (html tags), which we included in the vocabulary of the tokenizer.

5 CONCLUSIONS

Taking into consideration the optimizations at both the level of the pipeline via caching and multi threading mechanisms and the complexity of the model and the processing logic behind it and the results of the predictions comparable with models trained for bigger ammounts of time, we would suggest 10 points for our project.

REFERENCES

https://dl.acm.org/doi/pdf/10.1145/35037.35059 Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*

https://medium.com/@mehdi.iraqui/comparing-the-performance-of-llms-a-deep-dive-into-

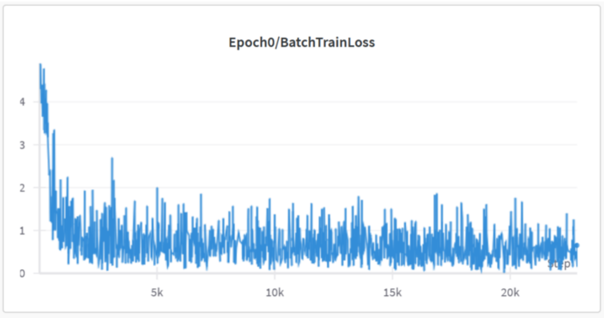
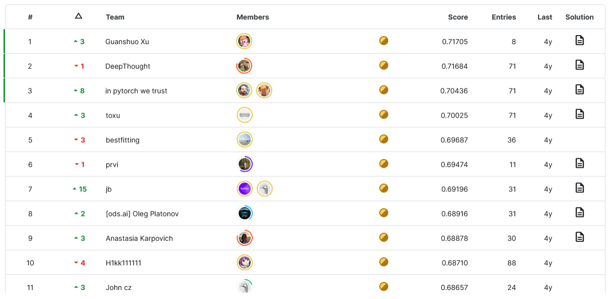


Figure 4: Loss on train evolution on epoch 1

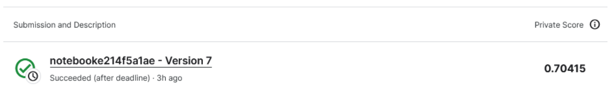


#	△	Team	Members		Score	Entries	Last	Solution
1	▲ 3	Guanshuo Xu		●	0.71705	8	4y	🗎
2	▼ 1	DeepThought		●	0.71684	71	4y	🗎
3	▲ 8	in pytorch we trust		●	0.70436	71	4y	🗎
4	▲ 3	toxu		●	0.70025	71	4y	🗎
5	▼ 3	bestfitting		●	0.69687	36	4y	
6	▼ 1	prvi		●	0.69474	11	4y	🗎
7	▲ 15	jb		●	0.69196	31	4y	🗎
8	▲ 2	[ods.ai] Oleg Platonov		●	0.68916	31	4y	🗎
9	▲ 3	Anastasia Karpovich		●	0.68878	30	4y	🗎
10	▼ 4	H1kk111111		●	0.68710	88	4y	
11	▲ 3	John cz		●	0.68657	24	4y	

Figure 5: Leaderbord

roberta-llama-and-mistral-for-disaster-tweets-8069e717548a Mehdi Iraqi *Comparing the Performance of LLMs: A Deep Dive into RoBERTa, Llama-2, and Mistral-7b for Disaster Tweets Analysis with LoRa*

https://pages.cs.wisc.edu/ remzi/OSTEP/dist-afs.pdf Peng Sucorresponding author and K. Vijay-Shanker *Investigation of improving the pre-training and fine-tuning of BERT model for biomedical relation extraction*



Submission and Description	Private Score
notebooke214f5a1ae - Version 7 Succeeded (after deadline) · 3h ago	0.70415

Figure 6: My submission