Conversational Question Answering with transformers-based models

Pietro Epis, Michele Milesi and **Anna Valanzano**
Master's Degree in Artificial Intelligence, University of Bologna
{ pietro.epis, michele.milesi, anna.valanzano}@studio.unibo.it

Abstract

The goal of Conversational Question Answering is to understand a text passage and to answer a series of interconnected questions that appear in a conversation. We developed a QA system on the CoQA dataset experimenting with two transformer-based models, BERTTiny and DistilRoBERTa. We want to compare these models and understand if the conversational history of QA pairs can improve the performance of the models and which are the most challenging types of questions.

1 Introduction

Transformers improved the state-of-the-art in several NLP tasks (among which Question Answering) without implying the use of any Recurrent Networks. In this project, we explore variants of BERT, a specialization of base transformers. As a first attempt, we implemented an extractive model that answers with a subset directly drawn out from the text passage. However, this type of model was not exactly a generative model. Therefore we moved to a different approach, represented by BertGeneration (following the hints at this Gitlab Repository), but this version was too computationally demanding. Hence, we moved again to a third and final version, that exploits a feature complete training procedure optimized for Transformers and provided by the Huggingface library.

2 System description

We used two transformers-based models, BERT-Tiny and DistilRoBERTa, that are two PyTorch pre-trained models (variants of the BERT base model). In the first approach, the model takes a text passage and a question and generates the corresponding answer, while in the second one, the model exploits also the conversational history of a certain question (i.e., all the previous QA pairs). For the second approach, we built the conversational history by appending all the previous QA pairs $q_0,\ a_0\ \dots\ q_{n-1},\ a_{n-1}$ to the current question q and to the text passage p, i.e., $q_n\ +\ p\ +\ q_0\ +\ a_0\ \dots +\ q_{n-1}\ +\ a_{n-1}$, separating each part with the token SEP (Reddy et al., 2018).
The original training part of the CoQA Dataset has been randomly shuffled and split into a validation set (20%) and a training set (80%). As the dataset is grouped by conversations (i.e. each element of the train list is a conversation), we are sure that each dialogue appears in only one split. For each model we built three DataFrames, one for each split (training, validation and test). Each row of the DataFrames represents a QA pair and the columns store some useful information about it, e.g. the `source` of the dialogue (Wikipedia, CNN, etc...), the `story`, i.e. the part of the text on which the question answers refer to, or the type of question. All the DataFrames have been converted into Datasets objects (from Hugginface's library) to ease the tokenization and training procedure.

3 Experimental setup and results

During the hyper-parameter tuning phase it turns out that setting the correct values for minimum and maximum length of the generated sentence is very crucial for the model performance. We observed that setting a high value for the `min_length` forces the model to generate long sentences even if it is not necessary. As lots of ground truth answers contain very few tokens and someone only contains one token (for instance, "yes"), we set the `min_length` to 1. Similarly, setting high values for `max_length` can lead to the same problem. We therefore set `max_length` to 10 because we observed that only a few thousand (over 127k) ground truth answers have a length greater than this threshold and that the average length of the answers is about 5.5 tokens. The data are tokenized with a general AutoTokenizer from HuggingFace. Setting the parameters of the tokenizer, we pad the

questions and the text passages to a length of 512 (the limit imposed by our models) and we truncate the text passage in case it exceeds 512 tokens.
For what concerns the training, we used the Hugginface Seq2SeqTrainer class that provides a feature-complete training in PyTorch optimized for Transformers. We performed splitting using seed 42 for reproducibility and using seeds 42, 2022, and 1337 for training/evaluation. We evaluated the model with the f1-score and reported the results for the validation and test set for all four models with three different seeds in Table 1, Table 2, Table 3.

		No History	**History**
BERTTiny	**Val**	0.1224	0.1216
	Test	0.1258	0.1254
DistilRoBERTa	**Val**	0.3994	0.4125
	Test	0.4052	0.4239

Table 1: f1 score of the four models with seed 42.

		No History	**History**
BERTTiny	**Val**	0.1221	0.1208
	Test	0.1248	0.1244
DistilRoBERTa	**Val**	0.3826	0.4405
	Test	0.3892	0.4458

Table 2: f1 score of the four models with seed 1337.

		No History	**History**
BERTTiny	**Val**	0.1203	0.1222
	Test	0.1240	0.1250
DistilRoBERTa	**Val**	0.3774	0.4234
	Test	0.3832	0.4392

Table 3: f1 score of the four models with seed 2022.

4 Discussion

4.1 Comparing models

As expected, the smallest BERTTiny based models performed the training/validation loop faster than DistilRoBERTa ones. Indeed, the execution time of BERTTiny has never exceeded an hour, while DistilRoBERTa has always taken more than four hours (on Colab platform). However, DistilRoBERTa guarantees far better results: the average f1 score of DistilRoBERTa with different seeds is 0.41, while for BERTTiny is 0.12.
In BERTTiny models, the presence of the conversational history does not significantly influence the performance, because the model is not powerful enough to exploit the knowledge given by the conversational history. Instead, in DistilRoBERTa models the conversational history always improves the performance and in both cases did not influence the training time. This is because, during the tokenization, the max length of the current question concatenated with the text passage (and eventually with the conversational history) is set to 512 tokens. If this concatenation of tokens is longer than 512, the sequence is truncated causing a possible loss of some QA pairs. However, we verified that the concatenation of inputs exceeds 512 only for 20k samples over 127k and the average length of the text passages plus the questions is about 360 tokens. Even if only a minority of samples is truncated, if the model considered longer sequences of tokens, it would reach better results.
The results obtained in our four models are coherent repeating the training with different seeds. We therefore can conclude that the DistilRoBERTa model with history is the best one.

Type	Yes	No	Count	Multiple	Fluent
F1	0.78	0.54	0.57	0.41	0.38

Table 4: F1 scores for different types of questions obtained with the DistilRoBERTa model with history and seed 1337.

4.2 Error analysis

We analyzed the model's errors on QAs taken from different sources (CNN, Wikipedia, etc.). For each source, we reported the five worst errors of the model and realized that in these worst cases, the model generates completely wrong answers with f1 score equal to zero. Indeed, analyzing the distribution of f1 scores across all the answers, most of the scores are either very near to zero, or very near to one. This means that the model mostly alternates between excellent answers and very bad ones. As the average of the f1 scores is very similar for each source, in order to understand which are the most challenging questions, we tried to identify five types of questions/answers, i.e., *no*, *yes*, *count*, *multiple*, and *fluent* (Reddy et al., 2018). As we can observe in Table 4, certain types of questions are more challenging for the model. According to these results, the easiest type of questions turns out to be *yes*, medium difficulty questions are *no*, *count*, *multiple*, and the hardest type is *fluent*. These results make sense because the *fluent* type requires a more structured and long answer, whereas the *yes* type is clearly easier to learn. Moreover, the results are coherent with the model's five worst errors, which are mainly fluent answers.

5 Conclusion

We can conclude that the Conversational Question Answering task is very challenging and has been hardly handled by our models, whose performance is strongly limited by the use of small BERT variants, which, despite representing the most reduced versions, are anyway very demanding in terms of computational resources. We observed that the presence of the conversational history can help the model, but also that it may not be fully exploited due to the risk of truncation of the text passage and the conversational history. Possible improvements may be reached through the implementation of more powerful transformers-based models as well as better management of the conversational history. For instance, we may use models with larger maximum input size or we may change the order of the QA pairs in the history (to be sure that the most recent ones are always included).

References

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2018. Coqa: A conversational question answering challenge. *CoRR*, abs/1808.07042.