Assignment 2

Davide Brescia, Daniele Marini and **Iulian Zorila**
Master's Degree in Artificial Intelligence, University of Bologna
{ davide.brescia, daniele.marini3, iulian.zorila }@studio.unibo.it

Abstract

Question-answer tasks have many pitfalls (e.g. implicit reference to elements in the conversation history). In the paper we are going to analyse two types of architectures (`BERTTiny` and `DistilRoBERTa`) with two differently structured inputs (with and without history) and go over how performance varies.

1 Introduction

Question answering tasks involve the use of neural networks to answer natural language questions based on a given context, such as a paragraph or document. Our goal is to find the best performing model capable of performing these tasks.

We used the `CoQA` (2) dataset which contains 127k questions with answers collected from 8k conversations in seven diverse domains.

After meticulously importing and analyzing the dataset, we used the `Experiment` class to perform models training. Specifically, we used 3 different seeds (42, 2022, 1337), 2 different architectures `BERTTiny` (1) (5) and `DistilRoBERTa` (3) with or without past conversations for a total of 12 trainings. Lastly we implemented an evaluation process by obtaining some information regarding issues and strengths of every model.

2 System description

We can summarize our work in four main areas:

- **Import and Analyse Dataset**: we used the functions `DownloadProgressBar`, `download_url` and `download_data`, provided in the Assignment 2, to import the dataset. We analysed in detail the dataset noticing a small variance in the length of paragaphs (an average of 271 words, with a standard deviation of 65), the distribution of number of questions is between 10 and 20, while the source distribution (*wikipedia*, *cnn*, *mctest*, *race*, *gutenberg*) is balanced in both training and test set apart from *mctest* which has 7.6% conversations in the training set compared to the others, containing roughly 20% each. When looking at the most frequent words (removing the stopwords) thourgh the `wordcloud` module, it turns out that "said" and "one" are the most recurrent words within the whole dataset.

- **Preprocess**: before feeding the data into the models we removed the unanswarable questions, added the history (past conversations) and splitted it into training, validation and test set. Then we expanded them such that in each row we have one story and one conversation. As requested, we defined two different functions:

  - $A = f(Q, P)$
  - $A = f(Q, P, H)$

  Where A, Q, P, H represent respectively the answer, question, context and history. The input and the output have been encoded as follows, if considering the second function (otherwise simply remove the history):

  - **input**: $[CLS]Q_i[SEP]P_i[SEP]H_i[EOS]$
  - **output**: $[CLS]A_i[EOS]$

  $H_i = Q_{i-1}[SEP]A_{i-1}[SEP]...Q_{i-n}[SEP]A_{i-n}$ with $n = \texttt{max\_history} = 3$. CLS and EOS are specific tokens for the begin and end of sentence while SEP is the separator.

- **Training**: we trained all the 12 models (4 for each seed) on different notebooks (one for each team member) to save time and

	Test				Validation			
	BertTiny		**DistilRoberta**		**BertTiny**		**DistilRoberta**	
	History	**No History**	**History**	**No History**	**History**	**No History**	**History**	**No History**
42	0.1866	0.1872	0.5102	0.4626	0.1748	0.1747	0.4899	0.4482
1337	**0.1915**	0.1843	**0.5105**	0.4596	0.1848	0.1823	0.4918	0.4472
2022	0.1845	0.1853	0.5097	0.4600	0.1819	0.1823	0.4822	0.4369

Table 1: This table summarizes the F1 scores obtained on test and validation set using different seeds with different models.

to split resources usage. The overall procedure was executed by the means of the class `Experiment` which takes care of model and trainer creations with their configurations. First splits the data according to the specified seed and after training saves the results on file.
Moreover, for each seed we saved the preprocessed data, along with its tokenized version (one when the history is present and one when it is not) to use later for evaluation.

- **Evaluation**: after the training process, we saved the models and evaluated them on the test set, then proceeded to compare the performance of the worst ones, focusing on their weaknesses by ranking the most recurrent errors.

3 Experimental setup and results

Taking inspiration from the HuggingFace tutorial (6) we configurate the model by setting the token ids (`cls, eos, pad`) to correspond with the tokenizer's ids. Then `max_length` and `min_length` were set to 99-th quantile of the answers length and its minimum value respectively.
As for the trainer arguments, we have changed the learning rate and the batch size: 2×10^{-4} and 16 for `BERTTiny` and 2×10^{-5} and 64 for `DistilRoBERTa`. The latter being a more complex model, needed a smaller learning rate and we have decided to increase the batch size, since we used a more powerful GPU (the premium one on Colab) to train the model. To pad the tokenized sentences we have used the `DataCollatorForSeq2Seq` which allows to pad them dynamically. To support the model answering questions which need context knowledge (past conversations), we tried to include 5 previous conversations, but resulted to be too much, confusing the model more than helping it, therefore we set the `max_history` parameter to 3.
After training all models for different seeds, both best and worst models resulted to be the ones of seed 1337. the SQUAD-F1 scores are depicted in the table above 1.

4 Discussion

The evaluation found that the models can tell when questions require a dry answer (yes or no) and when asking for a date or place and in some cases the answers are approximately correct, even if the score is zero, apart from some syntax errors.
We show some meaningful examples to demonstrate how models understood to answer with a similar pattern, but erroneously picked the wrong option (*Q, A, TA* are question, answer and true answer):

- ***Q:*** "What is its population?" ***A:*** "4,76,055" ***TA:*** "476,015".
- ***Q:*** "What is the main topic?". ***A:*** OCC ***TA:*** OCLC
- ***Q:*** "What Yale student was hired by the organization?". ***A:*** "Fred Kilgourg" ***TA:*** "Frederick G. Kilgour".
- ***Q:*** "What color was Cotton?" ***A:*** "brown" ***TA:***"white".

As expected `DistilRoBERTa` outperformed `BERTTiny` since it is a more complex model 178 million parameters vs 8,8 million, considering the full encoder-decoder architecture.

5 Conclusion

All in all, despite the fact that models are fine-tuned for only 3 epochs we obtained decent results, (consedering that the state-of-art reached around 90% of SQUAD-F1 score (4)). In the future it might be useful to tow the models for more epochs, or even better try different architectures, since the ones tested in this project are not suitable enough for text generation.

References

[1] Prajjwal Bhargava, Aleksandr Drozd, and Anna Rogers. 2021. Generalization in nli: Ways (not) to go beyond simple heuristics.

[2] Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266.

[3] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108.

[4] StanfordNLP. 2018. CoQa leaderboard.

[5] Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: The impact of student initialization on knowledge distillation. *CoRR*, abs/1908.08962.

[6] Patrick von Platen. 2020. Leveraging Pre-trained language Model Checkpoints for Encoder-Decoder Models.