Assignment 2

Yuri Noviello, Enrico Pallotta, Flavio Pinzarrone and **Giuseppe Tanzi**
Master's Degree in Artificial Intelligence, University of Bologna
{ yuri.noviello, enrico.pallotta, flavio.pinzarrone, giuseppe.tanzi }@studio.unibo.it

Abstract

Conversational question answering is a task in natural language processing where the goal is to generate a natural language response to a question based on a given context and previous conversation history. Transformer-based models have recently shown promise in this task due to their ability to handle long-range dependencies and encode rich contextual information. In this report we present results on the task achieved using seq2seq transformer-based models with DistilRoBERTa and BERT-tiny, fine-tuned on the CoQA dataset.

1 Introduction

Many existing studies tackle this problem simply by highlighting passages in the context that contain the answer, in our case instead we had to solve a text-generation task, for this purpose there are tailored models like T5, T0 and BART. Our approach exploits seq2seq transformers-based model for text generation, initializing an **Encoder-Decoder** model with a pretrained BERT-like language model for both parts, following the hugging face tutorials. Given the limited computational resources (and the assignment instructions) we **fine-tuned** DistilRoBERTa and BERT-tiny, expecting at least some decent results from the former one since it has a larger number of parameters compared to the second. We used the **CoQA** (**Co**nversational **Q**uestion **A**nswering) dataset, that differently from other QA datasets (e.g. SQuAD) provides as answers not only a span of the text but also a free-form text. For each model we've considered two different generative functions (with/without history) and, to avoid lucky results, we ran everything with 3 different seeds = {42, 2022, 1337}, reaching a total of 12 experiments. In the end we got an **F1 score** of **55.5%** using DistilRoBERTa and **14.8%** using BERT-tiny. The various experiments allowed us to understand better this type of architectures and how different combinations of the input and training recepies can affect performance.

2 System description

As any NLP task's pipeline we started with some data inspection and preprocessing, during the latter, we removed the unanswerable questions and we ignored the field *additional_answers*, as requested by the assignment. No text preprocessing was performed since the tokenizer does the common one by itself. The span text provided in the dataset, considered at first, was discarded since that the model didn't gain any improvements using it at training time. As requested, we defined two different generative functions:

$$A_i = f_\theta(Q_i, P) \tag{1}$$
$$A_i = f_\theta(Q_i, P, H_i) \tag{2}$$

where $H_i = \{Q_0, A_0, ..., Q_{i-1}, A_{i-1}\}$.
In the first case, given a context P, a question about it Q_i and its answer A_i, the encoded input and output are respectively:

$$"[CLS]\ Q_i\ [SEP]\ P\ [SEP]"$$
$$"[CLS]\ A_i\ [SEP]"$$

where the special token [CLS] and [SEP] represent respectively the begin of sequence and the separator token. In the second case, we have to consider also the dialogue history that will be encoded like this:

$$H_i = "Q_0[SEP]A_0[SEP]...[SEP]Q_{i-1}[SEP]A_{i-1}"$$

so the encoded input and output are respectively:

$$"[CLS]\ Q_i\ [SEP]\ P\ [SEP]\ H_i[SEP]"$$
$$"[CLS]\ A_i\ [SEP]"$$

For the Encoder-Decoder[1] model definition and training we adapted the code from Rothe et al.[4][3] that shows how to load large pretrained

language models for sequence-to-sequence task-specific fine-tuning for "Text Summarization" task. We tested mainly 2 different language models:

- **M1** : DistilRoBERTa [5]
- **M2** : BERT-tiny [2]

We decided to not share the parameters between encoder and decoder parts since we registered better results with this configuration.

3 Experimental setup and results

The training set was split into train and val , respectively 80% and 20%. To reduce the search space of text generation we decided to fix the max length of a generated answer to the 99% *quantile* value over the answers' lengths contained in the train set (=18), while the minimum value correspond to the length of the shortest answer (=3 considering the start and end tokens). We also enabled the beam search setting as number of best partial solutions the value of 4, and the maximum size of repeated n-gram to 3. The best batch size and learning rate found were respectively 16 and 2×10^{-5}.
The learning rate scheduler used was the constant one, since we noticed that with linear or cosine decreased too fast , this resulted in +6/7% F1 score. The optimizer used was the PyTorch standard one *"AdamW"*. Below there are the results (averaged over the seeds) obtained on validation (Table 1) and test set (Table 2), the main metric used to evaluate the models' performances was the squad implementation of the F1 score, in addition to that we used also the exact match score that gave us also a glimpse onto the general performance of the model, allowing us to know the percentage of well predicted answers.

	No history		With history	
	F1	Match	F1	Match
M1	48.38	35.64	53.37	38.93
M2	14.86	12.22	14.91	12.28

Table 1: Performances on the validation set

	No history		With history	
	F1	Match	F1	Match
M1	50.22	37.87	55.47	41.11
M2	14.82	12.32	14.72	12.28

Table 2: Performances on the test set

4 Discussion

Having a look at Table 2 we can see our expectations being met, DistilRoBERTa outperforms BERT-tiny. The additional information about the dialogue history seems to have a non-negligible impact on the performance, leading a ~5% F1 score improvement for the first model. The second one seems like it's not affected by this additional info, leaving the results unchanged. The first model reaches an exact match score of almost 41% which means that two answers over five were perfectly predicted. The second one instead, gave us very poor results, considering that a model that answers only "yes"/"no" does a 15% circa of F1 score (due to the QA distribution of the CoQA dataset). This model seems to be indifferent to the additional information provided by the history, probably such a low number of parameters implies a limited capability of knowledge representation and subsequent exploitation. Our **error analysis** was performed having a look at the worst five errors for each source of information (*cnn, gutenberg, mctest, race, wikipedia*) made by the models trained and evaluated with the history. For both models we can see that even if the samples' F1 scores are 0s, the answers reflect the main Q&A patterns (where:location, what:object, how many:number ...), furthermore in the first model we can notice the fact that many answers are not totally wrong, they are simply an alternative answer that is by the way correct w.r.t. the question and the context.
A clear example is:
Context : "I know what is inside the bag–a thermos with hot soup and a stainless-steel container with rice, vegetables and either chicken, meat or shrimp, sometimes with a kind of pancake."
Question : "What is in the bag?"
Answer$_{GT}$: "food"
Prediction : "a hot soup"
(other examples provided in the notebook).

5 Conclusion

In this report we described the experiments done to solve a conversational question answering task. Using seq2seq models we registered acceptable results, even though we did not use the most indicated models as encoder/decoder. As expected, the results obtained with the DistilRoBERTa based model were definitely better compared to the BERT-tiny's ones, maybe due to it's too small size. Finally, a possible improvement for the model may be adding a span extractor before the encoding/decoding phase, as well as using one of the generative models cited before.

References

[1] Huggingface. Encoder decoder models.

[2] Prajjwal1. Pre-trained bert-tiny.

[3] Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. Code of "leveraging pre-trained checkpoints for sequence generation tasks".

[4] Sascha Rothe, Shashi Narayan, and Aliaksei Severyn. 2020. Leveraging pre-trained checkpoints for sequence generation tasks. *Transactions of the Association for Computational Linguistics*, 8:264–280.

[5] 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.