Assignment 2

Antonio Politano, Enrico Pittini, Riccardo Spolaor and **Samuele Bortolato**
Master's Degree in Artificial Intelligence, University of Bologna
{ antonio.politano2, enrico.pittini, riccardo.spolaor, samuele.bortolato }@studio.unibo.it

Abstract

We define a model for addressing the QaA task which consists in two pre-trained transformer-based modules: the token importances extractor and the encoder-decoder. The training of such model is based on the combination of two different losses, with the usage of a teacher-forcing-like strategy. The results are satisfactory with respect to the average SQuAD F1 score.

1 Introduction

Question Answering (QA) is the task consisting in generating answers for questions from the passages containing the needed information. Optionally, also the history of previous question.answer turns can be used for producing the answer.

In our work, we use a model which consists of two modules: the **tokens importances extractor** and the **encoder-decoder** (i.e. seq2seq). The first module computes an importance score in $[0, 1]$ for each passage token, representing the likelihood that the token is in the span of the passage containing the answer. Then, the encoder-decoder takes as additional input these tokens importances, and it generates the answer. The reason of following this approach is to help the encoder-decoder in finding the interesting information in the passage, since it can be very long. Both modules are built from a pre-trained transformer-based architecture.

The dataset taken into account is the CoQA dataset, using the given training-test splitting and further splitting the training into training-validation, with proportions $0.8 - 0.2$. The unanswerable questions are deleted.

Two different pre-trained transformer-based architectures have been used, namely **DistilRoBERTa** and **BERTTiny**. For both models, three different random seeds have been set. Overall, 12 different experiments have been run, since there is the possibility to consider or not the history.

The results are quite similar among different seeds. With `DistilRoBERTa`, we achieve much better performances, in particular using the history. Regarding the worst answers, it can be noticed that the computed tokens importances poorly match the actual passage span. Nonetheless, even the worst answers generated by `DistilRoBERTa` are quite coherent with the question, while the worst answers generated by `BERTTiny` are not.

2 System description

Our model consists of two modules: the **Tokens Importances Extractor** (TIE) and the **Encoder-Decoder** (ED).

- The TIE is a pre-trained transformer-based encoder, with on top a linear layer. Basically, the encoder computes the contextual embeddings of the input tokens and then the linear layer computes the scalar importances scores out of them.
- The ED is a pre-trained transformer-based encoder-decoder modified to make it accept the tokens importances as second input of the encoder. The importances are injected inside the model by combining them to the input hidden states of every encoder block using a linear layer. We have chosen to use a different linear layer for every block of the encoder.

In the appendix are present two images, for the visualization of the tokens importances and of the overall model architecture.

Both modules are built from a pre-trained transformer-based architecture, either `DistilRoBERTa` or `BERTTiny`, taken from Hugging Face (Wolf et al., 2020). For implementing the ED, the Hugging Face encoder-decoder has been modified in order to take in input also the tokens importances and using them in each encoder block. For making this modification, the Hugging

Face code has been taken and adapted for our needs, both for `DistilRoBERTa` (HuggingFace, 2020b)and `BERTTiny` (HuggingFace, 2020a).

The used loss function is the sum of two losses.

- The first loss is about the goodness of the TIE. It measures the difference between the true span in the passage and the importances over the passage assigned by the extractor. In particular, this loss consists in computing a variant of the binary cross-entropy for imbalanced classes, where the true label of each passage token is 1 if it is in the span, 0 otherwise.
- The second loss is about the goodness of the ED. It measures the difference between the generated and the true answer. This is measured using the same standard loss function of the encoder-decoder: the cross-entropy loss.

Going more in depth, the tokens importances scores given in input to the ED are a combination of the output of the TIE with the true span labels of the input tokens: this is a teacher forcing mechanism, since we are giving the ground truth as input to the model during training. This combination is simply a weighted sum, where the weight related to the ground truth part is computed using a predefined cosine schedule, such that the weight is gradually decreased during training. The reasons of using this approach are the following.

- At the beginning of the training, when the TIE has not learned anything yet, passing the importances calculated by the TIE to the ED is useless, so the ground truth is used.
- On the other hand, training the ED only using perfect importances does not prepare it to the actual task, so at the end would be better to use the predicted importances instead.

3 Experimental setup and results

Regarding the training procedure, the Adam optimizer has been used, with learning rate $5 \cdot 10^{-5}$, with batch size 8 and 3 epochs. Three different random seeds have been set and also whether to use or not the conversational history has been tested. Overall, 12 different experiments have been run, evaluating them both on the validation and test sets, by using the average SQuAD F1 score (robinjia, 2018) on all the instances. Table 1 sums up the results for `DistilRoBERTa`, while table 2 sums up the results for `BERTTiny`.

4 Discussion

The best `DistilRoBERTa` configuration according to the validation score is the one with seed 2022 and using the QaA history, while the best `BERTTiny` configuration is the one with the seed 2022 and not using the QaA history.

An error analysis on the worst five errors for each source on the test set has been carried out for both models.

The model `DistilRoBERTa` is particularly good even in the worst cases. The less satisfactory results refer to the source *McTest*, while the ones of the other sources are less serious. Some errors are just an interpretation which is correct, although different from the expected one. The majority of the errors are either a wrong choice between multiple possibilities or incorrect answers that are still valid out of context for the given questions.

The model `BERTTiny` is particularly bad with respect to `DistilRoBERTa`. This is expected, since the former model is extremely lightweight and less complex than the latter. The TIE seems to be pretty inconsistent and unreliable for what concerns the worst errors. Most of the given answers are completely unrelated to the questions even out of context and the model often wrongly predicts "yes" as an answer. It is interesting to notice that for the given cases "yes" is the only present choice between "yes/no" answers. This can be possibly explained by the fact that the experiment of the paper *CoQA: A Conversational Question Answering Challenge*(Reddy et al., 2018) illustrated that positive answers are more common than negative ones among *free-form answers* (48.5% vs 30.3%).

For both models, most of the errors are obtained from questions that are pondered in an already established conversation.

5 Conclusion

Overall the `DistilRoBERTa` model obtains quite good results according to the average SQuAD F1 scores. Even considering the "worst" errors, the model produces logically coherent answers. On the other hand the `BERTTiny` model, as expected due to its much smaller dimension, has worse results and some of the generated answers are totally off the context of the questions.

In order to obtain better results increasing the training epochs could be a determining factor.

	Validation dataset SQuAD F1 score		Test dataset SQuAD F1 score	
	No QaA History	QaA History	No QaA History	QaA History
seed 42	0.48132	0.49752	0.49185	0.51392
seed 2022	0.47511	0.50281	0.49350	0.52342
seed 1337	0.44120	0.48739	0.44890	0.51066

Table 1: Average SQuAD F1 score evaluation for `DistilRoBERTa`

	Validation dataset SQuAD F1 score		Test dataset SQuAD F1 score	
	No QaA History	QaA History	No QaA History	QaA History
seed 42	0.23645	0.22185	0.23854	0.23488
seed 2022	0.24117	0.23626	0.24610	0.24361
seed 1337	0.24062	0.23203	0.25225	0.23502

Table 2: Average SQuAD F1 score evaluation for `BERTTiny`

6 Links to external resources

Link to GitHub repository.
Link to model weights folder.

References

HuggingFace. 2020a. transformers/models/bert/modeling_bert.py.

HuggingFace. 2020b. transformers/models/roberta/-modeling_roberta.py.

Siva Reddy, Danqi Chen, and Christopher D. Manning. 2018. Coqa: A conversational question answering challenge.

robinjia. 2018. Official evaluation script for squad version 2.0.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Appendix

Image 1 shows the token importances produced by the extractor for a certain sample versus the true passage span.

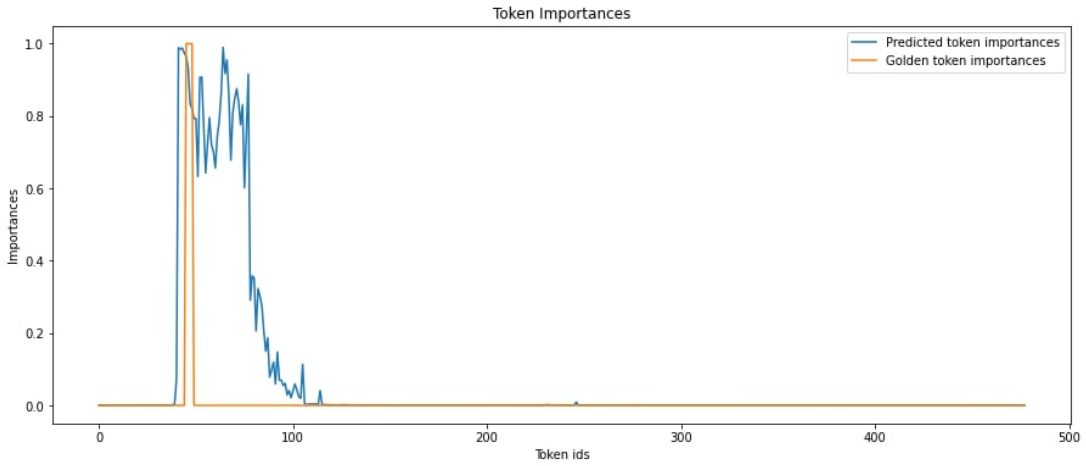

Figure 1

Image 2 shows the overall architecture of the model: tokens importances extractor plus the encoder-decoder.

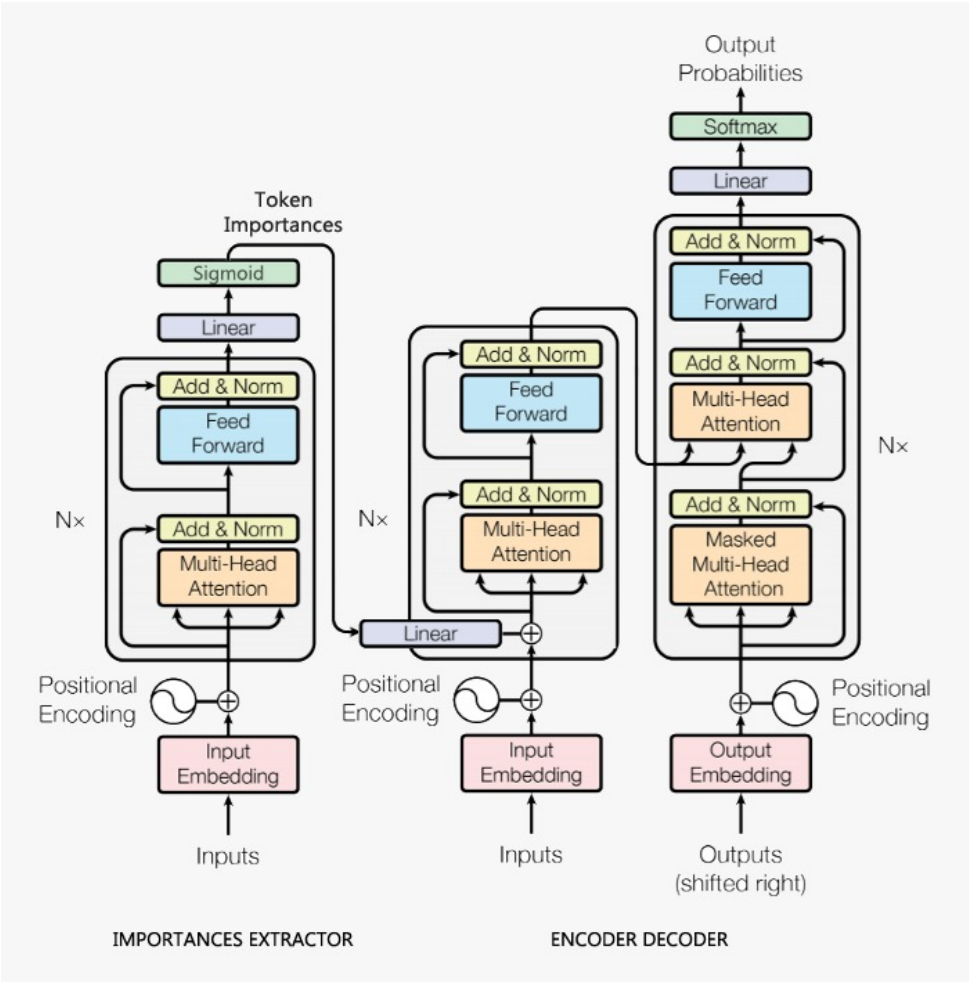

Figure 2