Efficient Fine-tuning with LORA

Large models like BERT and other more recent models have shown impressive results in many NLP tasks. But sometimes, on specific problems or datasets the performance can be improved with fine-tuning the model on the specific dataset. It consists of continuing learning and adapting the weights of the model from the specific dataset targeted. But with the number of weights involved in the models (at least more than millions) it becomes difficult to adapt models to specific problems without exploding memory and time costs. LORA provides an efficient way to fine-tune big models freezing the weights of the big model and adapting weight of a small parallel model.

The goal of this work package is to understand and run a LORA fine-tuning process. To date, LORA is one of the most efficient way to use big pretrained models and adapt them to a specific dataset/problem. (You can go further with QLORA, that uses quantization (reduction on precision weights) in order to reduce the memory needed).

Fine-tuning model for Text Classification

In this example a Bert base model is fine-tuned on a Yelp Dataset with LORA.

Colab example explained step by step :

https://colab.research.google.com/github/huggingface/notebooks/blob/main/transformers_doc/en/tensorflow/training.ipynb

nota : you can find an error with numpy, just add : `import numpy as np`

For more explanation details :

https://huggingface.co/docs/transformers/main/training

Other example of a Colab with a complete pipeline of a LORA fine-tuning of GPT-2 LLM:

https://colab.research.google.com/github/keras-team/keras-io/blob/master/examples/nlp/ipynb/parameter_efficient_finetuning_of_gpt2_with_lora.ipynb

As a comparison you can try to fine-tune directly the bert model, it’s possible but with a huge increase of training time.

Start to work on your project 2