Solution Building Report

By Vladislav Urzhumov, BS21-AI-01 (v.urzhumov@innopolis.university)

Current project is focused on the solution of Text Detoxification task, which is described in the paper which can be accessed by the link stated in the “original-paper.txt” of data/external.

Challenges

At the beginning, two challenges were faced.

Minor one was the challenge of dataset preprocessing, since the raw data had bad distribution of toxicity. To be more precise, toxic and neutral sentences were shuffled. The solution of this challenge is described in the Final Solution Report.

Major challenge was the lack of computational performance and GPU memory. Pre-trained models considered in the mentioned paper and some other models I wanted to work with required sufficient GPU memory and RAM, and that was a problem. Google Colab’s free plan allows students to train larger models on their resources, however their resources usage time is limited, and large enough models require more than it’s 15Gb of GPU memory. Thus, some of the hypothesis have failed due to that problem. That was a disclaimer ☺

Baseline

Firstly, T5-small model was probed to stand as a baseline. This model with a small batch size successfully were run locally. However, it’s performance was relatively poor, since sometimes it just reconstructed the sentences instead of paraphrasing.

Hypothesis 1: T5-base

Then, larger version of T5 was trained in Colab. It had a better performance, but even small batch_size led to a shortage of local CUDA memory during training. Overall, hypothesis worked: the size of T5 transformer mattered.

Hypothesis 2: Prophet Net by Microsoft

The power of Prophet Net, an encoder-decoder model that can predict n-future tokens for “ngram” language modeling instead of just the next token, was embraced. You can find the paper if you follow the link stored in data/external/prophet-net-paper.txt. This model was trained in Google Colab and its training results along with cell outputs may be seen in the jupyter notebook 2.0 (“2.0-prophet-model.ipynb”). This model is referred as the “best” solution reached in the current project, so this hypothesis is working!

Hypothesis 3: Pytorch-based training of Bert

For the sake of experiments, different dataset preparation techniques (torch-formalized datasets and dataloaders), model construction techniques (torch.nn.Module from pretrained bert-base) and training techniques were applied to train Bert-base model. However, even the Google

Colab's resources were not enough to serve enough computational performance and CUDA has run out of memory. Bert has never finished his training. Hypothesis and experiment were not working.

Hypothesis 4: Pytorch-based training of GPT2

Another experiment was training pytorch-based model of GPT2 with its corresponding tokenizer and appropriate training loop. Just as hypothesis 3, this one died due to insufficient computational resources.

(Code of experiments 3 and 4 can be found in (informal) jupyter notebook 3.0)

Conclusion

Any solution which removes bad words and paraphrases the toxic statements fairly well is a positive aid for any internet platforms. However, the best solution is one you reach by yourself! Thus, I tried to conduct experiment with different training techniques and different models. Prophet Net was the one which worked well, and it is now referred as "best" after 10 epochs of training. Some assumptions failed due to inefficient code or lack of computational resources. Baseline solution was beaten.

Thanks for reading, have a nice day! Feel free to contact in case of any questions.