FINE-TUNING ROBERTA TO CLASSIFY SENTIMENTS OF IMDB MOVIE REVIEWS

Chris Liu *
University of California, Santa Cruz
1156 High Street
Santa Cruz, CA 95064
{yliu298}@ucsc.edu

1 INTRODUCTION

In this project, I fine-tuned RoBERTa, a robustly optimized BERT (Liu et al., 2019), and evaluated its performance on the sentiment analysis task. I used the RoBERTa implemented by HuggingFace [1].

2 DATA PREPROCESSING

I use Pandas to read the provided "labeledTrainData.tsv" file as a data frame, and store the training sentences and labels in two lists as strings and integers $\{0, 1\}$, respectively. Before tokenizing the sentences, I define the maximum sequence length as 512 (tokens). The tokenizer then tokenizes every string by

1. splitting every sentence to tokens with a Byte-Pair-Encoding (BPE) tokenizer,
2. adding separation tokens `<s/>` and classification tokens `<s>`,
3. mapping all tokens to their corresponding IDs in the model vocabulary,
4. truncating sentences with more than 512 BPE tokens (Sennrich et al., 2015) to length 512, padding sentences with less than 512 BPE tokens with pad tokens,
5. creating attention masks to differentiate pad tokens and non-pad tokens (pad tokens will not be masked).

After the tokenization procedure, I compose a tensor dataset with the input IDs (created in step 3 above), attention masks (created in step 5 above), and the actual labels. I select 90% (22500) of the training examples for training and 10% (2500) for validation by randomly splitting the dataset. Finally, I make two data loaders for the training and validation data using a batch size of 4.

3 METHODOLOGY

3.1 MODEL

The RoBERTa model was proposed by Liu et al. (2019) in 2018 as a model based on Google's BERT Devlin et al. (2018). Here I assume the high-level understanding of BERT [2] so I'll only explain the difference between RoBERTa and BERT.

*Project 1 is my second project so the experiments will not be as comprehensive as my main project.

[1] https://github.com/huggingface/transformers

[2] For a high-level understanding of what BERT is, refer to this guide http://jalammar.github.io/illustrated-bert/ or my presentation slides for CSE290K https://docs.google.com/presentation/d/1cshZckIrm-T495wwF7E1ijbYyzNao68iaIEpUD_uv9k/edit?usp=sharing.

RoBERTa was built upon the same architecture as BERT, but modifies some key hyperparameters and removes the next-sentence prediction task. I used the RoBERTa-large with 340 million parameters, whereas there exists a base model with 110 million parameters. During the pre-training, RoBERTa was training on the 16GB dataset (the same as BERT's training data) and additional 144GB data. It uses dynamic masking, which has different masking patterns when a sequence is fed to the model. RoBERTa was also trained with a larger batch size (32,000). The resulting model has 24 layers, 1024 hidden units, and 16 attention heads. Overall, RoBERTa has about 2% to 20% improvement over BERT on the GLUE benchmark (Wang et al., 2018).

3.2 Optimizer

I use the default AdamW optimizer introduced in Loshchilov & Hutter (2017). Essentially, AdamW is an Adam optimizer (Kingma & Ba, 2014) with the implementation of weight decay regularization, which decouples the weight decay from the optimization steps taken with respect to the loss function. This implementation improves Adam's generalization performance.

3.3 Learning Rate

I used the learning rate = 1e-6, much less than the learning rate during pre-training to prevent overfitting. Based on the empirical evidence, a learning rate greater than 5e-5 will let the model overfit and drop the validation accuracy down to about 50%.

3.4 Epochs

I fine-tuned the model for four epochs to prevent overfitting.

4 Experiment Results

It takes five hours to fine-tune RoBERTa on the provided IMDb dataset. The validation loss does not decrease much during training, but the validation accuracy reaches its point, where with further training, the validation accuracy may start to drop. Our test accuracy on Kaggle is **0.9651%**.

Epoch	Train loss	Valid loss	Valid acc	Train time	Valid time
1	0.2935	0.2437	0.9552	1:14:59	0:02:38
2	0.1999	0.2427	0.958	1:14:54	0:02:38
3	0.1612	0.2433	0.9616	1:14:51	0:02:38
4	0.132	0.2444	0.9624	1:14:50	0:02:38

Table 1: Training statistics of the RoBERTa model.

5 Conclusions/Result Analysis

The accuracy of RoBERTa is nearly perfect. With data augmentation techniques, it might be able to achieve a slightly higher score with much fewer training examples.

Other than RoBERTa, I also tried BERT (Devlin et al., 2018), DistilBERT (Sanh et al., 2019), DistilRoBERTa, and XLNet (Yang et al., 2019). BERT, DistilBERT, and DistilRoBERTa all underperform RoBERTa-large. While XLNet has comparable performance as RoBERTa, it doubles the training time of RoBERTa.

6 Challenges and Future works

It is not easy to get enough compute to fine-tune the modern large-scale language models. I was only able to use the Google Colab Tesla P100 GPU with 16GB memory to fine-tune the model

with a batch size of 4. If a GPU with 32GB memory (Tesla V100 32GB) is available, it will take approximately a quarter of the time.

I was planning to use the approach introduced in Ziegler et al. (2019), in which I can use a GPT and PPO (Schulman et al., 2017) to generate additional positive and negative reviews based on strong rewards signals. However, the GPU memory required by this approach (32GB) is more than I have because GPT and BERT must run simultaneously, and if I keep a small batch size, it will take more than a week to generate augmented review data.

7 CONTRIBUTIONS

I am the only member of the group, so I own 100% contribution to everything in this project.

REFERENCES

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pp. 5754–5764, 2019.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*, 2019.