Walk Less and Only Down Smooth Valleys

Thomas Brink (tbrink@stanford.edu) Julian Cooper (jelc@stanford.edu) Quinn Hollister (bh9vw@stanford.edu)

Problem & Motivation

Transfer learning has become a valuable tool for handling a variety of downstream NLP tasks given a single generic pre-trained model. Since most of the downstream tasks are specialized, quality training data is a scarce resource, making training large models from scratch very difficult. In this project, we investigate how the performance of a pre-trained BERT encoder model changes for three different downstream prediction tasks when we include (i) regularization in our fine-tuning loss function and parameter (SMART by Jiang et al. [1]), (ii) multitask learning on fine-tuning datasets [2], and (iii) introduce relational layers that exploit similarity between tasks [2]. Our default minBERT model is pre-trained using two unsupervised tasks on Wikipedia articles: masked language modeling and next sentence prediction. For fine-tuning on downstream prediction tasks, we use the *Stanford Sentiment Treebank* (SST, 11,855 single sentence reviews), *Quora* (400,000 question pairs), and *SemEval STS Benchmark* (STS, 8,628 sentence pairs) datasets.

Extension 1: SMART Regularization

Many fine-tuning routines suffer from overfitting, which leads to poor performance on test sets of downstream prediction tasks. To address this, we implement the SMART regularization technique by Jiang et al. [1].

1. **Adversarial regularization**: The desired property is that when the input x is perturbed by a small amount, the output should not change much. To achieve this, Jiang et al. [1] optimize loss $\mathcal{F}(\theta)$ using: $\min_\theta \mathcal{F}(\theta) = \mathcal{L}(\theta) + \lambda_s \mathcal{R}_s(\theta)$, where

$$\mathcal{L}(\theta) = \frac{1}{n}\sum_{i=1}^{n} l(f(x_i;\theta), y_i) \qquad \text{regular loss function}$$

$$\mathcal{R}_s(\theta) = \frac{1}{n}\sum_{i=1}^{n} \max_{\|\tilde{x}_i - x_i\|_p \le \epsilon} l_s\left(f(\tilde{x}_i;\theta), f(x_i;\theta)\right) \qquad \text{regularization term}$$

$$l_s(P,Q) = \mathcal{D}_{KL}(P\|Q) + \mathcal{D}_{KL}(Q\|P) \qquad \text{symmetric KL-divergence}$$

2. **Bregman proximal point optimization**: To prevent aggressive updating, the optimization routine is changed so that we introduce a trust-region-type regularization at each iteration, so we only update the model within a small neighborhood of the previous iterate. Specifically, we update parameters θ by $\theta_{t+1} = \text{argmin}_\theta \mathcal{F}(\theta) + \mu \mathcal{D}_{Breg}(\theta, \tilde{\theta}_t)$, where

$$\mathcal{D}_{Breg}(\theta, \tilde{\theta}_t) = \frac{1}{n}\sum_{i=1}^{n} l_s\left(f(x_i;\theta), f(x_i;\tilde{\theta}_t)\right) \qquad \text{Bregman divergence}$$

$$\tilde{\theta}_t = (1-\beta)\theta_t + \beta\tilde{\theta}_{t-1} \qquad \text{regularized update step}$$

Customizations. Although we were mostly able to follow the authors' algorithm, we needed to customize the algorithm to handle dropout. To do this, we remove dropout (switch to eval mode) when evaluating the adversarial loss within the training block. This allowed us to achieve the desired asymptotic behaviour with respect to our ϵ parameter. We publish psuedocode for our revised algorithm in the writeup appendix.

We decided not to invest time doing a large hyperparameter grid search for this project since we had access to suggested values from the original paper and limited compute resources. However, we did vary our regularization weight term $\lambda = \{0.01, 0.1, 1, 5, 10, 100\}$ for a small number of iterations to confirm regularization was being applied at the right order of magnitude.

Discussion and takeaways

- Benefits of SMART regularization are most pronounced when the fine-tuning dataset is small or ill-balanced. Areas of the feature space that are sparsely covered in such a dataset will tend to create sharp decision boundaries that may not be generalizable, and hence perform poorly on out-of-sample tests.
- However, incremental benefits of regularization decrease with the size of data used and regularization may even restrict learning too much in this setting.

Extensions 2-3: Multitask Learning

Fine-tuning the default BERT model does not generalize well to other downstream prediction tasks. To design an approach that is generalizable across different tasks, we implement two multi-task learning routines.

1. **Round-Robin Multitask Fine-Tuning**
   - Equally-weighted round-robin: train on all prediction tasks while balancing data.
   - Additional pretraining: exploit rich paraphrase data through additional pretraining.
   - Interleaved fine-tuning: batch-adjusted task-specific network on top of BERT.

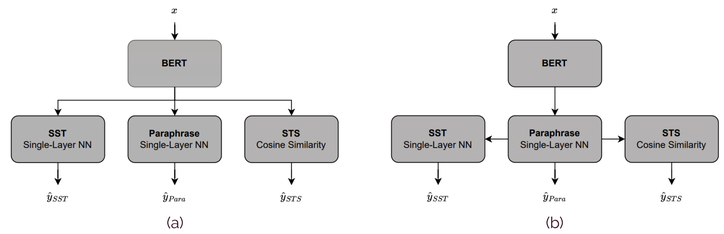

Figure 1. Model architectures for equally-weighted round-robin and interleaved fine-tuning in (a) and additional pretraining in (b).

2. **Rich Relational Layer Combining Similar Tasks**
   - Capture similarities across tasks to synergize joint performance.
   - Share relational layer between rich paraphrase data and STS data on top of BERT.

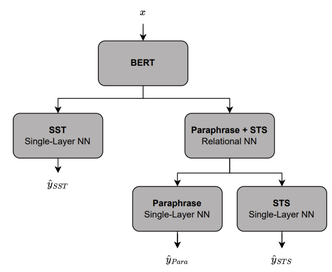

Figure 2. Network architecture for the rich relational approach.

Discussion and takeaways

- The benefit of introducing more data has proven to be greater than the potential risk of an imbalanced multitask fine-tuning procedure.
- For all data-rich models, SST accuracy slightly dropped. This is likely caused by the added Quora data being orthogonal to or disrupting the sentiment task.

Experiments & Analysis

Table 1 compares the dev accuracy of our default model (pretrain and finetune) with benchmarks provided in the handout for the single-task classifier. Our results are all close to the benchmarks which gives us confidence in the 'correctness' of our default implementation.

Table 1. *Dev accuracy of baseline BERT models (pretrain and fine-tune) vs. benchmarks for single-task classifier and runtimes per training epoch on a Google Colab GPU.*

	Sentiment (SST)			Sentiment (CFIMDB)		
Model type	Accuracy	Benchmark	Runtime (s)	Accuracy	Benchmark	Runtime (s)
Pretrain default	0.393	0.390 (0.007)	30	0.788	0.780 (0.002)	45
Fine-tune default	0.522	0.515 (0.004)	120	0.963	0.966 (0.007)	150

Table 2 gives the dev performance for all proposed model approaches on the SST, Quora, and STS tasks. We include three non-multitask models; the baseline models from Table 1 (i.e., BERT and fine-tuned SST) and an SST-fine-tuned model with SMART. Furthermore, we evaluate various multitask learning approaches combining SMART, round-robin, and the rich relational layer model.

Table 2. *Dev set accuracies of single- and multitask models on three fine-tune tasks. The models are divided into three groups; (i) single-task approaches, (ii) small-scale (equally-weighted batches) approaches, and (iii) large-scale rich (relational) approaches. The best model for each task is bolded. Runtimes are given in seconds per training epoch on an AWS EC2 instance.*

	Sentiment (SST)	Paraphrase (Quora)	Similarity (STS)	
Model type	Accuracy	Accuracy	Correlation	Runtime (s)
Pretrain default	0.396	0.380	0.019	9
Fine-tune default	0.525	0.522	0.240	25
Fine-tune smart	0.520	0.501	0.382	161
Fine-tune rrobin	0.524	0.726	0.583	67
Fine-tune rrobin+smart	**0.532**	0.741	0.680	464
Fine-tune rrobin-pre+smart	0.519	0.851	0.690	3.037
Fine-tune rrobin-full	0.498	0.863	0.762	1.420
Fine-tune rrobin-full+smart	0.485	0.850	0.714	4.549
Fine-tune rrobin-full+rlayer	0.501	**0.867**	**0.802**	1.550

SMART performs best out of the single-task models, achieving the highest STS score. However, unsurprisingly, all multitask models easily beat these single-task baselines in terms of paraphrase and STS scores. Out of the multitask models that only use a limited portion of the available data, the SMART implementation again performs best, especially in terms of STS score. When making use of the entire Quora data, we can see that paraphrase accuracy is significantly boosted, reaching values over 85%. In this setting, it seems that the added value of SMART vanishes.

The best model overall is the rich relational interleaved round-robin model, which achieves test set performances of 0.503 (SST), 0.865 (paraphrase), and 0.789 (STS). In terms of runtime, SMART is most costly, especially when used in combination with richer data.

Discussion and takeaways

- Our best model combines multitask learning using varying batch sizes with a rich relational layer between correlated tasks.
- One next step we would like to explore is data transformations that allow for continuous multimodal distributions that might better predict semantic similarity.

[1] Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. SMART: robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. *CoRR*, abs/1911.03437, 2019. URL http://arxiv.org/abs/1911.03437.

[2] He P. Chen W. Liu, X and J. Gao. Improving multi-task deep neural networks via knowledge distillation for natural language understanding. *arXiv preprint arXiv: 1904.09482*, 2019.