SMARTer Multi-task Fine-tuning of BERT

Stanford CS224N Spring 2024 Default Project

Disha Ghandwani
Department of Statistics
Stanford University
disha123@stanford.edu

Aditya Ghosh
Department of Statistics
Stanford University
ghoshadi@stanford.edu

Rahul Kanekar
Department of Statistics
Stanford University
rkanekar@stanford.edu

Abstract

We develop and evaluate a minimalist BERT model (minBERT) with a focus on fine-tuning for three distinct NLP tasks: sentiment classification, paraphrase detection, and semantic textual similarity. Initial experiments reveal that training task-specific linear layers with the pre-trained minBERT frozen yields poor performance. Fine-tuning the entire model along with the task-specific layers shows some improvement for individual tasks, but fails to deliver satisfactory results when applied jointly to all three tasks. To address this, we incorporate the adversarial regularization and proximal optimization techniques from the SMART framework by Jiang et al. (2020). Our findings indicate that while SMART enhances performance, the onus lies on the underlying model architecture. Among the various models and settings we experimented with, the SMART framework with task-specific LSTM layers trained with an **e**xponentially decaying learning rate with a **r**andomized task in each iteration (SMARTer) seems to deliver the best performance.

TA mentor: Yuan Gao; External collaborators/mentor: None; Sharing project: No

1 Introduction

In recent years, large language models like BERT have revolutionized the landscape of natural language processing (NLP). These models, however, face significant challenges when fine-tuned on downstream tasks with limited data, often leading to overfitting and inadequate generalization. Fine-tuning approaches that simply train a task-specific layer, freezing the pre-trained BERT model, tend to deliver poor performance. Fine-tuning the full model (the task-specific layer(s) and the BERT parameters) seems to improve the performance for individual tasks, but still struggles with overfitting, especially in multi-task settings where the model must generalize across diverse objectives.

To address these issues, we explore an adversarial regularization technique, SMART, due to Jiang et al. (2020). We implement it via the Bregman proximal point optimization method from the same paper, which constrains aggressive parameter updates. Our project focuses on three key NLP tasks: sentiment classification, paraphrase detection, and semantic textual similarity. By implementing and fine-tuning a minimalist BERT model (minBERT) with the SMART framework, we aim to improve model robustness and generalization, providing a more efficient and effective solution for multi-task learning applications.

Our experiments reveal that incorporating the SMART framework by Jiang et al. (2020) helps to enhance generalization by reducing overfitting compared to baseline approaches. However, its effectiveness seems to be tied to the underlying model architecture. Among the models and settings we explored, extending minBERT with task-specific LSTM layers and using an exponentially decaying learning rate seemed most promising for the three tasks we consider. In all, we conclude that successfully optimizing multi-task learning with minBERT remains a complex challenge heavily dependent on designing the models for the particular tasks at hand.

2 Related Work

Several techniques have been proposed to improve fine-tuning of large pre-trained language models like BERT on downstream tasks. Liu et al. (2019) introduced the MT-DNN framework which allows effective multi-task learning by simultaneously training the full-model with task-specific layers. To enhance generalization, Jiang et al. (2020) introduced the SMART framework, which combines adversarial regularization and Bregman proximal point optimization. This framework encourages smoothness in the model outputs, making them more robust to small input perturbations, and constrains parameter updates to prevent aggressive changes that could lead to overfitting.

Adversarial regularization builds on earlier ideas proposed for training robust models, e.g., (Miyato et al., 2018) proposed virtual adversarial training, a method that introduces small perturbations to the input data to improve model generalization. While originally developed for NLP tasks, this general technique has shown promise in domains like computer vision Shafahi et al. (2019) and speech recognition Conneau et al. (2017) – providing a broadly applicable way to obtain fine-tuned models invariant to input noise, akin to human language understanding.

Another line of work by Hu et al. (2021) proposed LoRA – adapting pre-trained models by training low-rank parameter updates rather than modifying the full model weights. This reduces the number of trainable parameters (thus drastically reducing the training time) while still allowing effective multi-task transfer learning.

3 Approach

We implement a working MT-DNN framework (Liu et al., 2019) which can effectively handle the three given tasks, namely, sentiment classification (SC), paraphrase detection (PD), and semantic textual similarity (STS). For multi-task fine-tuning, we primarily utilize the adversarial regularization and Bregman proximal point optimization techniques proposed by Jiang et al. (2020). Before describing the regularization approach and further refinements, we discuss below the *baseline* approaches that we also implement from scratch.

3.1 Baseline models

First, we implement the self-attention and transformer layers of minBERT (for details, refer to the default final-project handout on the class website).

Next, we fine-tune the minBERT model for SC on two datasets: Stanford Sentiment Treebank (SST) and the CFIMDB movie reviews database. We consider two approaches: (1) freeze the parameters of the pre-trained minBERT and train a linear layer on top it, for the specific task of SC; (2) fine-tune all parameters of minBERT along with the linear layer, as described in Liu et al. (2019).

Finally, we fine-tune the minBERT model to perform three tasks: SC on the SST data, PD on the Quora data and STS on the SemEval data (cf. Section 4 for more details). We consider the following baseline approaches: (3) add separate linear layers for each of the three tasks and minimize the *sum of the losses*, freezing the parameters of the pre-trained minBERT; (4) fine-tune the entire minBERT model and task-specific layers by minimizing the *combined loss* (summed) across tasks.

3.2 Adversarial regularization and Bregman proximal point optimization

Adversarial regularization aims to control model complexity by encouraging smoothness in the output (Jiang et al., 2020). Given training data $\{(x_i, y_i)\}_{i=1}^n$ and a model $f(x; \theta)$, the regularized objective F is given by

$$F(\theta) := L(\theta) + \lambda_s R_s(\theta), \quad \text{where} \quad L(\theta) := \frac{1}{n} \sum_{i=1}^{n} \ell(f(x_i; \theta), y_i),$$

λ_s is a tuning parameter, and $R_s(\theta)$ is the smoothness-inducing regularizer, given by

$$R_s(\theta) := \frac{1}{n} \sum_{i=1}^{n} \max_{\|\tilde{x}_i - x_i\|_p \leq \epsilon} \ell_s(f(\tilde{x}_i; \theta), f(x_i; \theta)),$$

where ℓ_s is the symmetrized KL-divergence, defined as $\ell_s(P, Q) = D_{\text{KL}}(P\|Q) + D_{\text{KL}}(Q\|P)$. This regularizer restricts the output $f(x; \theta)$ to not change much when a small perturbation is added to the input x_i, thus controlling model complexity.

Following Jiang et al. (2020), we plan to use momentum Bregman proximal point (MBPP) optimization to minimize the regularized objective. Given a pre-trained initial model $f(x; \theta_0)$, the vanilla Bregman proximal point update step at iteration $(t + 1)$ is given by

$$\theta_{t+1} = \underset{\theta}{\text{argmin}}\{F(\theta) + \mu D_{\text{Breg}}(\theta, \theta_t)\},$$

where $\mu > 0$ is a tuning parameter, and $D_{\text{Breg}}(\theta, \theta_t)$ is the Bregman divergence, defined as

$$D_{\text{Breg}}(\theta, \theta_t) := \frac{1}{n} \sum_{i=1}^{n} \ell_s(f(x_i; \theta), f(x_i; \theta_t).$$

This approach constrains the updated parameters θ_{t+1} to be within a small neighborhood of the previous iterate θ_t. This prevents aggressive updating during fine-tuning iterations. The procedure can be further accelerated using a momentum term, resulting in the MBPP method (see Jiang et al. (2020, Algorithm 1)).

In summary, adversarial regularization promotes smoothness by penalizing output changes under small input perturbations, while Bregman proximal point optimization constrains parameter updates to a local trust region, together preventing overfitting to the training data during fine-tuning.

3.3 Low-Rank Adaptation of Large Language Models (LoRA)

Hu et al. (2021) proposed a slightly different method of fine-tuning large language models for downstream tasks. They hypothesized that when fine-tuning pre-trained models, the changes to the model parameters induced by downstream training often lie within much smaller subspaces. This suggests that instead of adjusting all the model parameters, we can instead fine-tune them by freezing the original parameters and then training low-rank perturbations.

We adopted LoRA in our setting to fine-tune the key and query matrices W_q, W_k of the BERT model. Suppose the pre-trained key matrix is W_k of dimension $d \times d$. During fine-tuning, we apply low-rank updates as:

$$W_k' = W_k + AB,$$

where A and B are $d \times m$ and $m \times d$ matrices respectively, with $m \ll d$. This low-rank formulation reduces the number of trainable parameters from $\mathcal{O}(d^2)$ to $\mathcal{O}(dm)$, with $m = 4$ in our implementation.

We approached LoRA as an alternative to SMART, that can be used to fine-tune the model while still constraining the deviations. Its effectiveness can be seen in the results we obtained. Using it in tandem with the randomized training scheme described below greatly reduced the training time per epoch while still giving results comparable to our other methods.

3.4 Finer approaches

First, we discuss two simple techniques that enabled us to improve upon the SMART and LoRA frameworks:

- **Randomizing the task:** At every iteration, we add the loss for one of the three tasks, chosen uniformly at random. While the SMART loss or LoRA takes care of overfitting, this simple trick resolves the issue that the losses for different tasks could differ in scale. Further, it speeds up the training process.
- **Exponentially decaying learning rate:** This simple idea (the name is self-explanatory) can also reduce overfitting. Further, this appears to improve stability over the epochs.

Combining SMART with these two techniques, which we term SMARTer (SMART with **e**xponentially decaying lr and **r**andomized task-specific objective), delivered superior performance (see Table 3). Additionally, we also explored other losses, e.g., cosine similarity loss for the STS task.

Among the models explored, SMARTer outperformed the others, except for the semantic textual similarity (STS) task. Despite tuning various hyperparameters, the results on STS remained unsatisfactory. We eventually realized the bottleneck was the underlying neural network architecture used for the task output layers. Replacing the linear layers with (i) Transformer and (ii) LSTM architectures revealed that SMARTer with LSTM layers achieved the best overall performance.

4 Experiments

4.1 Data and evaluation method

We describe below the datasets we use, with the associated tasks and standard evaluation metrics.

- **SST** (Stanford Sentiment Treebank): Contains 11885 sentences from movie reviews with 215154 unique phrases labeled as negative, somewhat negative, neutral, somewhat positive, or positive. *Task*: sentiment classification; *evaluation metric*: percentage accuracy of classification.
- **CFIMDB** (movie reviews database): Comprises 2434 highly polar movie reviews labeled as positive or negative, with reviews potentially longer than one sentence. *Task*: sentiment classification; *evaluation metric*: percentage accuracy of classification.
- **Quora** data: Consists of 404298 question pairs labeled to indicate if they are paraphrases. *Task*: paraphrase detection; *evaluation metric*: percentage accuracy.
- **SemEval** data: Includes 8628 sentence pairs rated on a scale from 0 (unrelated) to 5 (equivalent). *Task*: semantic textual similarity; *evaluation metric*: Pearson correlation.

For SC and STS tasks, we use the full datasets. For the PD task, we randomly select 10000 data points per epoch to ensure diverse samples that we can train on within the time limit.

4.2 Experimental details and results

We summarize our experimental results in Tables 1 to 3. First, in Table 1 we report how the baseline approaches (1) and (2) described in Section 3.1 perform when we use the two datasets for sentiment analysis. Next, in Table 2 we report the performance of the different baselines for multi-task fine-tuning. Finally, Table 3 compares the various multi-task learning approaches we described in Sections 3.2 to 3.4. We delegate our analysis of these results to Section 5.

Hyperparameters: We run experiments with a learning rate of 10^{-3} for running last-linear-layer (approaches (1) and (3)) and 10^{-5} for running full-model (approaches (2) and (4)). Due to GPU constraints, we mostly use batch size of 16. For the hyperparameters for our SMART implementation, we adhere to those outlined in the original paper by Section 3.2. For the exponentially decaying learning rate approach, we set the initial learning rate as 10^{-5} and cut it down by a factor of .9 in each epoch, running it for 20 epochs. Additionally, dropout with a probability of 0.3 is applied in all models to prevent overfitting.

Loss functions: We used the cross-entropy loss for the SC task, the BCE loss for the PD task, and the MSE loss for the STS task. In addition, we also tried other choices of loss functions; in particular, the cosine similarity loss for the STS task.

Different scaling of losses: To manage potential scale differences among these losses, we introduced the novel solution of randomly adding one of the losses in each iteration. This not only took care of the issue of different scales, but also make the code run much faster, cutting down computation time by almost a factor of 3.

Model: First, we implemented only linear layers for each task on the top of minBERT. Next, as an attempt to improve the performance, we implemented more advanced models. For sentiment analysis, we used a Bidirectional Long Short-Term Memory (BiLSTM) network with 2 layers and a hidden size of 768, processing BERT embeddings to capture contextual information and predict sentiment classes across 5 categories. For paraphrase detection, we implemented a Siamese BiLSTM architecture, using two identical BiLSTM networks with 2 layers and a hidden size of 768, to process pairs of sentences and determine their similarity by comparing their LSTM outputs. Finally, for Sentence similarity we also implemented a Transformer Encoder network composed of 6 layers and 8 attention heads, which models relationships between sentence tokens and outputs a similarity score.

These methods combine the contextual richness of BERT embeddings with the sequence modeling capabilities of RNNs and transformers, leading to improved performance in each task.

Table 1: Performance of different baseline approaches for sentiment classification

Baseline Model	SST		CFIMDB	
	Train Acc	Dev Acc	Train Acc	Dev Acc
(1) Linear layer on minBERT	38.6%	39.3%	78.4%	78.8%
(2) Fine-tune the full model	97.9%	52.4%	98.5%	97.1%

Table 2: Performance of different baselines for multi-task fine-tuning

Baseline model	SC (accuracy)	PD (accuracy)	STS (correlation)
(3) Task-specific linear layers	38.1%	64.6%	0.228
(4) Jointly fine-tune the full model	49.9%	73.4%	0.305

Table 3: Performance of multi-task fine-tuning (full-model) with advanced techniques. Here random refers to the randomized task technique and exp.lr refers to the exponentially decaying learning rate technique (cf. Section 3.4)

	Training Accuracy			Dev Accuracy		
	SC	PD	STS	SC	PD	STS
Baseline	72.4%	74.7%	0.907	49.9%	73.4%	0.305
SMART	48.6%	74.1%	0.636	42.9%	73.8%	0.275
SMART + random	80.1%	71.8%	0.869	49.6%	71.1%	0.359
SMART + random + cosine-similarity	80.0%	71.4%	0.865	48.7%	70.4%	0.355
SMARTer (SMART + exp.lr + random)	76.2%	70.9%	0.867	52.7%	73.5%	0.362
LoRA + exp.lr + random	59.2%	68.3%	0.725	49.2%	68.9%	0.336
SMARTer + Transformer	72.0%	72.3%	0.801	51.3%	73.3%	0.390
SMARTer + LSTM	89.5%	75.8%	0.938	50.5%	75.1%	0.483
SMARTer + LSTM (deeper)	94.0%	75.8%	0.944	49.8%	74.2%	0.427

5 Analysis

Our preliminary results reported in Tables 1 and 2 suggest the following:

- Training task-specific layers alone, with minBERT parameters freezed, yields poor performance for downstream tasks.
- Fine-tuning the entire model improves performance for a single task but the same for multi-task fine-tuning is not satisfactory.

Next, we analyze Table 3, which summarizes the results from fine-tuning the full-model with additional bells and whistles to reduce overfitting and improve generalization.

- The SMART regularization reduces the overfitting a lot, as compared to the baseline approach.
- The simple technique of exponentially decaying the learning rate greatly stabilized the results across different epochs.
- The novel idea of adding the loss for one of the tasks chosen at random in every iteration not only addressed potential scale differences between tasks effectively, but also reduced training times almost by a factor of 3.
- LoRA seems to perform equally good as SMART in alleviating overfitting. Although we report only the results for LoRA with additional safety nets of exponentially decaying learning rate and the random loss technique, it did perform equally as good as SMART when applied without these accessories.

- Merely increasing the depth of the LSTM layers did not improve the performance on the dev set.
- The performance of SMART and LoRA for the STS task remained unsatisfactory, despite the additional techniques described in Section 3.4. This limitation was eventually attributed to the simplicity of the models. Implementing transformer and LSTM approaches separately revealed that the SMARTer approach combined with LSTM yielded the best overall performance.
- Playing with the loss function or hyperparameters did not drastically improve performance, suggesting a sense of consistency and robustness in the ongoing research as a community.

6 Conclusion

We explored techniques to improve multi-task fine-tuning of the miniBERT model across sentiment analysis, paraphrase detection, and semantic textual similarity tasks. Our key finding is that while naive fine-tuning approaches struggle with overfitting and poor generalization, carefully designed regularization methods can significantly boost performance. Specifically, combining adversarial regularization from the SMART framework with techniques like randomized task sampling and decayed learning rates proved effective at reducing overfitting. However, model architecture remains critically important – extending miniBERT with task-specific LSTM layers delivered state-of-the-art results among our experiments.

While promising, our work had limitations in fully optimizing semantic textual similarity, suggesting multi-task fine-tuning remains a complex challenge heavily tied to tailoring models for each task. Future research could explore more advanced architectures like adapters or prompts to better capture cross-task relationships. Specifically, we suggest implementing the pairwise word interaction model (PWIM) architecture by He and Lin (2016) for the STS task; we could not do it for time limitation. Additionally, investigating techniques to quantify and mitigate harmful biases inherited during pre-training is crucial for ensuring fair and trustworthy fine-tuned models. Overall, achieving robust multi-task generalization from large language models will require continued research efforts across improved regularization, tailored architectures, and bias mitigation strategies.

7 Ethics Statement

One key concern is the propagation of biases inherent in pre-trained models and the datasets used for fine-tuning. These biases could lead to discriminatory behavior, affecting marginalized groups disproportionately. There is also a risk that our models could be misused to create fake reviews or manipulate opinions, causing societal harm. To address these issues, we should evaluate and reduce bias to ensure our models treat all groups fairly.

Additionally, the high computational resources required for training and fine-tuning models have environmental impacts. Employing energy-efficient training methods and leveraging cloud services powered by renewable energy sources could help reduce our carbon footprint.

8 Acknowledgement

We thank our mentor Yuan Gao for his helpful guidance and support during this project. All team members – Disha Ghandwani, Aditya Ghosh, and Rahul Kanekar – contributed equally to the conceptualization, implementation, analysis, and documentation of this project. We also thank the entire CS224N Spring 2024 course staff for providing a great learning experience.

References

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Hua He and Jimmy Lin. 2016. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 937–948, San Diego, California. Association for Computational Linguistics.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2020. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2177–2190.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4487–4496.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. 2018. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993.

Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. 2019. Adversarial training for free! *Advances in neural information processing systems*, 32.