Project Proposal

Quinn Hollister
Stanford University
bh9vw@stanford.edu

Julian Cooper
Stanford University
jelc@stanford.edu

Thomas Brink
Stanford University
tbrink@stanford.edu

1 Research paper summary (max 2 pages)

Title	SMART: Robust and Efficient Fine-Tuning for Pre-trained Natural Language Models through Principled Regularized Optimization
Venue	The 58th annual meeting of the Association for Computational Linguistics (ACL 2020)
Year	2023 (first submitted 2019)

Table 1: Bibliographical information for literature review [**?**].

Background. Transfer learning has become an incredible tool for addressing various downstream NLP tasks like question answering and semantic similarity [**?**]. Since most of the downstream tasks are more specialized, quality training data is a scarce resource, making training large models from scratch very difficult. Transfer learning allows us to first use a massive training data set of unlabeled data to generate weights of a complex model that can capture semantic and syntactic meaning of the language, which we can then "fine-tune" on the specific task we are trying to address by replacing the top layer of the language model by a domain-specific sub-network. This allows us to efficiently use scarce downstream data. Unfortunately, during fine-tuning, we are often too aggressive with the parameter updates and can lose the structure gained from pre-training. This produces overfitting and an inability of the model to generalize from the data. Jiang et al. [**?**] address a way to (i) induce smoothness of the model prediction and (ii) control the parameter updates by using a kind of trust-region regularization for each iterate to conserve the knowledge gained from pre-training.

Summary of contributions. To effectively control the **extremely high complexity** of the LLM, Jiang et al. [**?**] propose a smoothness-inducing adversarial regularization technique. Essentially, the desired property is that, when the input is perturbed by a small amount, the output should not change much. At the fine-tuning stage, they present a new loss function which includes a regularizer term, the computation of which requires a maximization problem that can be solved efficiently using projected gradient ascent. This regularizer term is essentially measuring the local Lipschitz continuity of under the symmetrized KL-divergence metric. So, the output of our model does not change much if we inject a small perturbation to the input, the magnitude of which is constrained to be ϵ small in the p-euclidean metric. Thus, we can encourage function f to be smooth within the neighborhoods of our inputs. This is particularly helpful when working in a low-resource domain task.

$$
\begin{aligned}
min_\theta \mathcal{F}(\theta) &= \mathcal{L}(\theta) + \lambda_s \mathcal{R}_s(\theta) \\
\mathcal{L}(\theta) &= \frac{1}{n}\sum_{i=1}^{n} l(f(x_i;\theta), y_I) \\
\mathcal{R}_s(\theta) &= \frac{1}{n}\sum_{i=1}^{n} \max_{\|\tilde{x_i} - x_i\|_p \leq \epsilon} l_s\left(f(\tilde{x_i};\theta), f(x_i;\theta)\right) \\
l_s(P,Q) &= \mathcal{D}_{KL}(P\|Q) + \mathcal{D}_{KL}(Q\|P)
\end{aligned}
$$

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. In order to solve the regularizer term, they develop a class of Bregman

proximal point optimization methods. This strongly regularizes and prevents the parameter updates from deviating too much from their current state. Consequently, the model can retain the knowledge of the out-of-domain data in the pre-trained model. Each subproblem of this parameter update does not have a closed-form solution, so we need to solve it using SGD-type algorithms.

$$\theta_{t+1} = argmin_\theta \mathcal{F}(\theta) + \mu \mathcal{D}_{Breg}(\theta, \tilde{\theta}_t)$$
$$\mathcal{D}_{Breg}(\theta, \theta_t) = \frac{1}{n} \sum_{i=1}^{n} l_s \left(f(x_i; \theta), f(x_i; \tilde{\theta}_t) \right)$$
$$\tilde{\theta}_t = (1 - \beta)\theta_t + \beta \tilde{\theta}_{t-1}$$

They used BERT as a baseline model, and made changes to the loss function and optimization step as outlined in (i) and (ii). They then evaluated their SMART model on GLUE and compared to other state-of-the art models. Their model contained roughly 356 million parameters, whereas the state of the art model parameters were nearly two orders of magnitude higher. In their ablation study, they saw that each module (the regularizer and the update optimization algorithm) produced benefits upon the base model independently, i.e. these two components benefited each other. Also the improvement that SMART achieved over the BERT baseline was more pronounced in small datasets, illustrating how their changes were able to effectively decrease the amount of overfitting that manifested itself in the fine-tuning phase. Also somewhat surprisingly, the benefits from Multi-Task Learning can be combined with the changes from SMART to attain GLUE scores that improved on both separate models. Thus, they were able to show that SMART improves the generalization of Multi-task learning which itself is a kind of regularization adjustment to the baseline.

Limitations and discussion. While the methodology and the experiments section of this paper was encouraging and exciting, the lack of discussion on computational complexity and resources needed to implement these changes was concerning. SMART requires two additional optimizations that must occur at every step. This includes finding the maximal loss for an ϵ-ball around the input, and finding the argument minimizer for the Bregman loss function. While the authors claim that both of these optimizations should be relatively quick, no figures or estimates were included on the additional runtime needed per step or per epoch. This discussion does not impact the results that their trained models achieved, but it would impact the feasibility of using their approach on larger models.

Why this paper? We choose this paper for two reasons: the emphasis on using advanced optimization techniques, and the principled nature of the approach to fixing over-fitting. All three of us have extensive computational math backgrounds, and the algorithms proposed in this paper connected with our research interests of optimization and algorithm design. Also, the fact that this approach can be applied without extensive hyper-parameter tuning surprised and excited us. After reading the paper, we felt like the approach described validated those beliefs (although the true test will be the performance gain we see in our experiments).

Wider research context. Although transfer learning has proven to be a great tool to address various tasks in NLP, aggressive fine-tuning can hinder the performance on the directed downstream task. This paper aims to improve this correspondence between language model and learning, which can carry over to nearly every relevant task. Other approaches at solving this over-fitting problem include hyper-parameter tuning heuristics like gradually unfreezing layers, updating a certain fixed subset of layers, or using triangular learning rate schedules, but all of these require significant tuning efforts that lead to unprincipled solutions with little flexibility.

When focusing these large neural network models at language, we often want to utilize the structure of language learned from training on a large corpus to a certain group of tasks that might not be so similar. Moreover, an optimal language model should be flexible and capable of adapting its knowledge of the trained language to nearly any task. Unfortunately, the results of blind transfer learning are poor, which motivates changes in this secondary step to correct for over-fitting. This paper does so through changes made to the loss function and the optimizer steps, but another popular approach is Multi-task Learning (MTL) [**?**]. MTL aims to reduce the amount of overfitting by training these disparate tasks jointly. This implicitly adds a regularizer term to the loss function. The actual process of finding a useful subset of tasks which will be useful is an active area of research.

2 Project description (1-2 pages)

Goal. To investigate how the performance of our pre-trained BERT encoder model changes for three different downstream prediction tasks when we include (1) additional pre-training on target-domain data, and (2) regularization in our fine-tuning loss function as well as using trust-region regularization in the parameter update step.

Task. Performance of our extensions to the default BERT pre-training and fine-tuning pipeline will be evaluated on three separate downstream prediction tasks: (a) sentiment analysis, (b) paraphrase detection, and (c) semantic textual similarity.

- **Sentiment Analysis**: Want to predict sentiment of movie reviews from the Stanford Sentiment Treebank (SST). Our predictions will be categorical {negative, somewhat negative, neutral, somewhat positive, positive}.
- **Paraphrase Detection**: Want to predict whether a given question pair (A,B) is a "paraphrase pair" (i.e. one is a paraphrase of another) or not. Our predictions for this task are binary (true if paraphrase pair).
- **Semantic Textual Similarity**: Want to predict whether a sentence pair (A,B) is semantically similar. Our predictions will be real number similarity score values on a scale from 0 (unrelated) to 5 (equivalent meaning).

Data. Our minBERT model is pre-trained using two unsupervised tasks on Wikipedia articles: masked language modeling and next sentence prediction. We do not need to interact with the Wikipedia dataset directly because we are able to load pre-trained weights as our starting point.

For fine-tuning and evaluating our model on downstream prediction tasks, we will use the Stanford Sentiment Treebank, Quora Dataset, and SemEval STS Benchmark Dataset. The *Stanford Sentiment Treebank* (SST) dataset consists of 11,855 single sentence reviews, where a review is labelled categorically {negative, somewhat negative, neutral, somewhat positive, positive}. The *Quora Dataset* (Quora) consists of 400,000 question pairs with binary labels (true if one question is paraphrasing the other). And finally, the *SemEval STS Benchmark Dataset* (STS) consists of 8,628 different sentence pairs, with each given a score from 0 (unrelated) to 5 (equivalent meaning). Our datasets require some minimal pre-processing, including tokenizing sentence strings, lower-casing word tokens, standardizing punctuation tokens, and padding sentences to enable matrix multiplication.

Methods. Idea: we plan to extend the default minBERT model in two ways:

- **Additional pre-training for target domain** [**?**]: While we expect our pre-trained minBERT model weights to be effective for paraphrasing and semantic similarity analyses, we assume it will struggle to classify sentiment. This is become the Wikipedia corpus is largely filled with non-emotive, informational language. Because of this, we want to try adding an incremental pre-training layer that includes more emotive language (e.g. editorial corpus MULTIOpEd [**?**]) trained on masked language modeling and next sentence prediction (original BERT objectives [**?**]).
- **Regularization of fine-tuning loss and optimizer step** [**?**]: Many fine-tuning routines suffer from over-fitting, which lead to poor performance on test sets of downstream prediction tasks (see literature review). Having carefully included corpus text from relevant domains in our pre-training steps, we want to make sure our fine-tuning does not diverge too rapidly from the pre-trained weights. For this we introduce a regularization technique that adjusts both our loss and objective functions (see literature review section for detail).

  Note, if time permits, we want to try extending our model fine-tuning pipeline to include multi-task learning. In their development of the SMART framework, Jiang et al. found that combining SMART with multi-task learning achieved the best benchmark performance results [**?**].

Baselines. For starters, we will be comparing our sentiment analysis accuracy with the baseline scores presented in the default project handout. These scores include baseline accuracies for both

pre-trained and finetuned models on the SST and CFIMDB datasets. Mostly, we can view the respective baselines as 'correctness' tests for our implementation.

Next, we will evaluate sentiment analysis accuracies for both of our extensions and compare these with the provided baseline scores. For all of our models, i.e., pretrained, fine-tuning, fine-tuning with extensions, we will also evaluate the performance when it comes to paraphrase detection and semantic textual similarity. The scores obtained from our pretrained and first fine-tuning models can then be considered somewhat as a baseline for our extensions. Since the idea behind the regularization used for our extensions is to prevent over-fitting during fine-tuning, we would hope our final model improves on our benchmarks.

Evaluation. In terms of evaluation, we will be using accuracy for the sentiment analysis and paraphrase detection tasks, while we use Pearson correlation for semantic textual similarity. We note that the sentiment analysis accuracy is based on multi-class classification, while the paraphrase detection task is binary. The reference paper uses accuracy as their main metric as well, applying their model to a variety of tasks, such as those in the GLUE benchmark set [**?**].

We will be comparing our achieved scores for sentiment analysis with the baseline scores in the default project handout (for pre-trained and fine-tuned models), while, for the other two tasks, we will be comparing scores between our different implementations.