Walk Less and Only Down Smooth Valleys

CS224N Default Project

Quinn Hollister
Stanford University
bh9vw@stanford.edu

Julian Cooper
Stanford University
jelc@stanford.edu

Thomas Brink
Stanford University
tbrink@stanford.edu

Abstract

Transfer learning has become a valuable tool for handling a variety of downstream NLP tasks given a single generic pre-trained model [1]. 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 different downstream prediction tasks when we include (1) regularization in our fine-tuning loss function and parameter gradient step (SMART by Jiang et al. [2]), (2) multi-task learning with task-specific datasets, and (3) rich relational layers that exploit similarity between tasks. We find that, while SMART is effective for preventing overfitting on smaller fine-tuning datasets, its value diminishes as we expose the model to larger task-specific training data. Our best results come from combining multitask learning with relational layers for related tasks.

1 Introduction

With the advent of transformers and large language models, performance on various downstream NLP tasks drastically improved, leading to advances like ChatGPT [3]. These models have captivated the world with their ability to mirror human language, answer questions, and perform various tasks that only require abstract prompts. This explosion in ability can be explained partially by research in Multitask Learning (MTL) [4], which allowed these models to aggregate training samples over various tasks, and to learn shared knowledge amongst them. This allowed more complex tasks – which might have a labeled dataset of insignificant size – to benefit from the language structure learned in richer data environments. The process of encouraging a model to utilize these different tasks effectively, and without "forgetting" previously hard-won language knowledge, is hard to get right and is the focus of this project.

More specifically, in this research, we use pretrained embeddings from the BERT large-language model for three 'fine-grained' tasks; sentence sentiment classification, paraphrase detection, and paired sentence semantic textual similarity [5]. We first test the ability of BERT embeddings to be tuned towards sentence sentiment classification only - a single classifier. We then implement SMART, a state-of-the-art regularization method proposed by Jiang et al. [2], which aims to tackle overfitting issues during the process of fine-tuning on small tasks using pretrained embeddings. Furthermore, we apply multitask learning approaches that learn on all of the three aforementioned tasks simultaneously [4]. In doing so, we explore various methods to balance learning across tasks while exploiting as much of the available data at hand [6]. Our findings suggest that, for low-data fine-tuning tasks, SMART significantly improves performance of multitask models. This improvement vanishes when we adapt our multitask models to handle richer data during fine-tuning. In addition, we find that by sharing neural network layers across similar tasks, we can drastically improve prediction performance.

This paper continues with a short review of related work in Section 2. Then, we discuss our model approaches in Section 3, after which we outline our main experiments and corresponding results in Section 4. Then, we perform a deepdive analysis of our successes and failures in Section 5. Last, we provide our conclusion and ideas for future work in Section 6.

2 Related Work

When focusing 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 [7], which motivated us to investigate techniques to reduce overfitting and use our available task-specific training data effectively.

Our first challenge was to address overfitting so as not to avoid destroying the value of our pre-trained BERT weights. Jiang et al. [2] proposes a regularization technique (SMART) that works by introducing an adversarial loss function and creating trust-regions for parameter updates. This approach appealed to us because it achieved strong benchmark scores and did not require significant hyperparamter tuning, which we did not have the resources to tackle. Other approaches for solving this overfitting problem include hyperparameter 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 [8].

Our second challenge was to use the available task-specific data as effectively as possible to "fine-tune" our model weights for the three prediction tasks. Across literature, most approaches use some kind of multitask learning (MTL) approach in combination with increasingly advanced language models such as RoBERTa [9], ALBERT [10], and ELECTRA [11]. These MTL approaches can be as simple as adding single-layer neural networks for each task on top of the pretrained model, although more sophisticated methods can be designed to capture richness of data and correlations across tasks [12]. Our starting point was a simple round-robin model that combined equally-sized data portions from each task. Taking inspiration from Liu et al. [6] [13] for handling variable-sized datasets, we investigated both additional pre-training and uneven batching strategies. These enabled us to incorporate as much task-specific data as possible into the fine-tuning process. For closely related downstream tasks, Liu et al. [6] also proposes "shared multitask learning" by adding a neural network layer on top of pretrained weights to connect and capture correlations between similar tasks.

3 Approach

The starting point for this research is the BERT large language model [14]. Using this pre-trained model, we follow the idea of transfer learning by next focusing on three specific fine-tuning tasks; sentence sentiment classification (SST), paraphrase detection, and semantic textual similarity (STS). As baselines, we evaluate the performance of the pre-trained (min)BERT model and an SST-fine-tuned model on these tasks. This provides some insight into the transfer learning ability of BERT. However, the focus of this research lies in three extended modeling approaches.

3.1 Regularization of Loss and Optimizer Step

Many fine-tuning routines suffer from overfitting, which leads to poor performance on test sets of downstream prediction tasks. Having carefully included corpus text from relevant domains in our pre-training steps, we do not want our fine-tuning to diverge too rapidly from the pre-trained weights. Therefore, we make use of regularization techniques by implementing SMART [2]. The rationale behind SMART is twofold.

First, to effectively control the high complexity of the large-language model, SMART uses a smoothness-inducing **adversarial regularization** technique. 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. [2] optimize loss $\mathcal{F}(\theta)$ using: $\min_\theta \mathcal{F}(\theta) = \mathcal{L}(\theta) + \lambda_s \mathcal{R}_s(\theta)$, where

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

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

As in [2], we use $l_s(P, Q) = \mathcal{D}_{KL}(P\|Q)+\mathcal{D}_{KL}(Q\|P)$ (symmetric KL-divergence) for classification problems, and the squared error loss $(p-q)^2$ for regression tasks. Here, the regularization term requires a maximization problem that can be solved efficiently using projected gradient ascent. Note that this regularization term is measuring the local Lipschitz continuity under the symmetrized KL-divergence metric. That is, the output of our model does not change much if we inject a small perturbation (constrained to be ϵ small in the p-euclidean metric) to the input. Thus, we can encourage our model f to be smooth within the neighborhoods of our inputs. This is particularly helpful when working in a low-resource domain task.

Second, 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, Jiang et al. [2] develop a class of **Bregman proximal point optimization** methods. Specifically, we update parameters θ by $\theta_{t+1} = \text{argmin}_\theta \mathcal{F}(\theta) + \mu\mathcal{D}_{Breg}(\theta, \tilde{\theta}_t)$, where

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

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

We coded up the SMART algorithm from Jiang et al. [2] ourselves. Although we were mostly able to follow the authors' algorithm, we needed to **customize the algorithm** to handle dropout. When we evaluate the model before and after our adversarial update, we experience large differences in the output logits, even in the limit where our maximum deviation parameter ϵ goes to zero. This difference in logits results from essentially comparing the outputs of two different models, since different dropout masks can be thought of as different models [15]. This naturally presents a problem when calculating the symmetric KL-divergence loss.

During our experiments, we found that the benefit of SMART gets drowned out from the non-actionable noise introduced by the different dropout layers, i.e., there is a non-learnable loss of the same order of magnitude as the smoothness loss. We found it strange that this particular implementation detail (whether or not to allow for dropout during this adversarial loss calculation) was not addressed in any of the SMART literature. In order to address this problem, we decided to enforce the absence of the dropout layer when evaluating the adversarial loss within the training block. This allowed us to achieve the desired asymptotic behaviour with respect to our ϵ parameter. Algorithm 1 summarizes our implementation of the adversarial loss calculation in SMART (appendix).

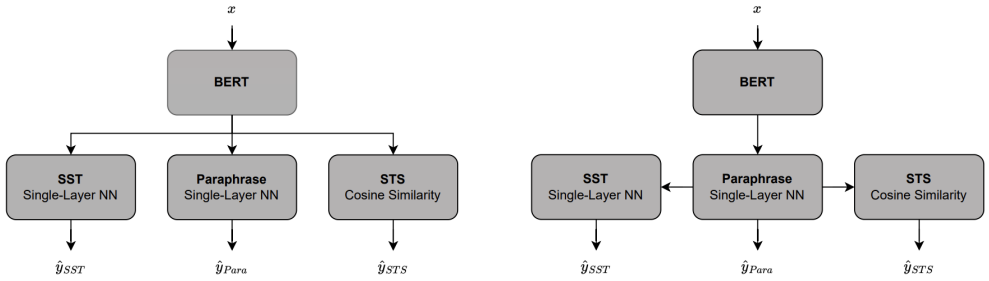
Finally, we decided not to invest time doing a large hyperparameter grid search for this project since we had access to suggested values from the original paper and limited compute resources. However, we did vary our regularization weight term $\lambda = \{0.01, 0.1, 1, 5, 10, 100\}$ for a small number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

3.2 Round-Robin Multitask Fine-Tuning

The baseline BERT implementation assumes that fine-tuning only on sentiment classification will generalize well to paraphrasing and similarity prediction tasks. Even superficial experiments prove that this is not the case. To design an approach that is generalizable across different tasks, we implement a batch-level round-robin MTL routine. After loading in pre-trained BERT weights, we cycle through the SST, paraphrase, and STS data while fixing our batch size. Within each batch iteration, we update the relevant parameters for each dataset using a single pass. For SST, we use a single-layer NN that outputs logits for all sentiment classes. For the paraphrase data, we use a single-layer NN that takes as features both the sentence pair embeddings h_u and h_v as well as their absolute difference $|h_u - h_v|$, as in SBERT [16]. For STS, we find that calculating the cosine similarity between the embeddings of a pair of sentences works well. Our updates use cross-entropy (SST), binary cross-entropy (paraphrase), and mean-squared error (STS) loss. Overall, this round-robin procedure makes sure our fine-tuned model gets exposed to data from all tasks.

While we expect equally-weighted round-robin fine-tuning (with SMART) to efficiently learn from multiple tasks, this procedure is restricted to using the same number of data points for each task. That is, we only use a fraction of data from larger datasets (in our case: Quora). To fully harness the richness of our data, we test two adaptions of the original multitask fine-tuning logic: (i) additional

Quora pretraining [17] and (ii) interleaved fine-tuning [6]. The first approach involves using the majority of paraphrase data for initial training and then applying equally batched round-robin MTL across all three tasks for the final iterations. The idea behind this approach is that learning on the paraphrase data may provide a warm start for the other tasks, and using richer data may boost performance of the corresponding tasks. However, additional pre-training on paraphrase detection may not be relevant to the other tasks and, in fact, could cause us to 'forget' more generalizable BERT embeddings. Therefore, our second, interleaved, approach harnesses the richness of our data while keeping the learning of all tasks separate. Namely, we reduce the batch size of smaller datasets for each iteration, including a larger relative chunk of Quora data per batch. Ideally, this should boost performance for the paraphrasing task without compromising accuracy of our sentiment task. Figure 1 visualizes the model architecture of the round-robin model and its adaptions.



(a) Round-robin architecture.

(b) Additional pre-train architecture.

Figure 1: Network architectures for our round-robin (a) and additional pre-training (b) approaches. Both approaches take as input textual data x, feed this through a pre-trained BERT model, and then focus on specific tasks. Note that (a) may take different-sized batches x_B of data per task.

3.3 Rich Relational Layer Combining Similar Tasks

In addition to fully capturing the richness of our data, we further adapt our model to handle relations across tasks. That is, when tasks are similar, overall learning may benefit from combining the signals in these tasks' data. To do this, we propose using a combined relational layer between similar tasks before task-specific fine-tuning, much like MT-DNN [6]. We hypothesize this is especially helpful when data for some of the combined tasks is scarce. Essentially, this idea applies transfer learning like we do with minBERT, but now to a more specialized setting of similar tasks.

Of the three downstream tasks we use for evaluation, we note that the paraphrase and semantic textual similarity tasks are closely related; both evaluate the similarity of sentence pairs (albeit on different scales). In addition, the STS dataset is much smaller than the Quora dataset. By funneling these two datasets through a shared layer (with LeakyReLU nonlinearity) before diverging into task-specific learning, we hope to exploit the richness of the paraphrase data to enhance STS learning. Although such a relational layer may already be useful in the equally-weighted batch setting, we focus on the interleaved round-robin approach, i.e., using as much of the paraphrase data to boost STS as possible. Note that such a shared layer, which maps embedding pairs to a single hidden layer embedding, does not allow us to use cosine similarity for STS anymore. Instead, we pass the scaled Hadamard product $\frac{h_u \odot h_v}{||h_u||_2||h_v||_2+\epsilon}$ between paired embeddings h_u and h_v as a feature to the relational layer (summing this feature exactly returns cosine similarity). Figure 2 visualizes this rich relational architecture.

4 Experiments

4.1 Datasets

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), *Quora*, and *SemEval STS Benchmark* (STS) datasets. The 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 consists of 400,000 question pairs with binary labels (true if one question is paraphrasing the other).

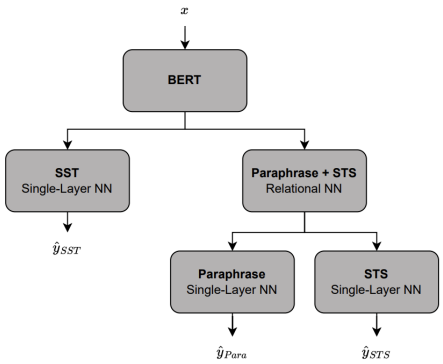


Figure 2: Network architecture for the rich relational approach. The network takes as input textual data x, feeds this through BERT, and then splits into groups of similar tasks for fine-tuning.

The STS dataset consists of 8,628 different sentence pairs, each given a continuous score from 0 (unrelated) to 5 (equivalent meaning). In addition, for baseline evaluation, we use the *CFIMBD* dataset, which contains 2,434 movie reviews, labeled as either positive or negative. 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.

4.2 Experimental Details

All experiments share some general settings; we run 10 epochs with a fine-tune learning rate of 1e-5 and a hidden-layer dropout probability of 0.3. For our pretrain default minBERT model, we use a learning rate of 1e-3. When memory allows us to, we set our batch size to be 64.

Regularization of Loss and Optimizer Step [smart] The hyperparameter values specific to our SMART [2] implementation are $\lambda = 5$, $\epsilon = 1e-5$, $\sigma = 1e-5$, $\beta = 0.995$, $\mu = 1$, and $\eta = 1e-3$. Due to memory usage, we reduce our batch size to 16 rather than 64. We experimented with λ to ensure that our adjustments would be applied at the right order of magnitude.

Round-Robin Multitask Fine-Tuning [rrobin] For equally-weighted round-robin, we fix batch size of 64 across all datasets and run `floor(len(sts_train_data) / args.batch_size)` iterations per epoch. For round-robin with additional pretraining (rrobin-pre), we set batch size separately for the pretrain and round-robin portions. By default both are set to 64, but we ratchet down to 16 if running model with SMART. For interleaved round-robin (rrobin-full), we fix the number of iterations per epoch and correspondingly adapt the batch size per dataset. Given a maximum batch size of 64, set for the paraphrase data, we infer the batch sizes of SST and STS to be 3 and 2 respectively.

Rich Relational Layer Combining Similar Tasks [rlayer] The rich relational layer approach follows the same parametrizations as (interleaved) round-robin. That is, when equally weighting all datasets, we can use a batch size of 64, while we resort to smaller adjusted batch sizes for SST and STS when using all paraphrase data to be put through the relational layer.

4.3 Results

To evaluate the different models, we use accuracy for the sentiment analysis and paraphrase detection tasks, and pearson correlation for STS. We note that the SST dataset forms a multi-class classification task, while the paraphrase detection and CFIMDB labels are binary.

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

In Table 1, we report the dev set performance of our baseline pre-trained (min)BERT and fine-tuned models for sentiment classification alongside benchmarks from the project handout. Our results are all close to the benchmarks, which gives us confidence in the 'correctness' of our implementation. The pre-trained BERT model clearly performs much better than random, indicating that its transfer learning is somewhat effective. However, fine-tuning drastically improves classification accuracy.

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

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.

From Table 2, we can see that 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 (fine-tune rrobin, fine-tune rrobin+smart), the SMART implementation again performs best, especially in terms of STS score. It seems that, using a limited amount of data, this task is particularly prone to overfitting when trained without regularization. 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. Namely, STS scores for the non-SMART models exceed those from SMART models.

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). This model stands out in its STS score, which confirms our hypothesis that capturing correlations between paraphrase and STS data through a shared layer can boost similar tasks' performance. In terms of runtime, SMART is most costly, especially when used in combination with richer data.

5 Analysis

5.1 Qualitative Deepdive

Sentiment Classification (SST). The distribution of SST classes is not uniform, with classes 0 and 4 (strong negative and positive) being less represented at 12.6% and 14.9%, respectively, and classes 1-3 being more represented at 26.2%, 20.8% and 25.3%, respectively. Theoretically, we would expect these middle classes to be more difficult to predict, since their conveyed sentiment is less straightforward. For example, when reading the sentence "It's a lovely film with lovely performances by Buy and Accorsi", we could very well assess this to be a strong positive (class 4), but the actual label is somewhat positive (class 3). It turns out that the SST multi-class classification task is hard; the highest reported accuracy is around 59% [9]. To identify how this class difficulty affects our predictions, Table 3 provides precision, recall, and F_1-scores, from which we find that the "neutral" class is most difficult to predict, and the model is biased to predicting the most frequent classes.

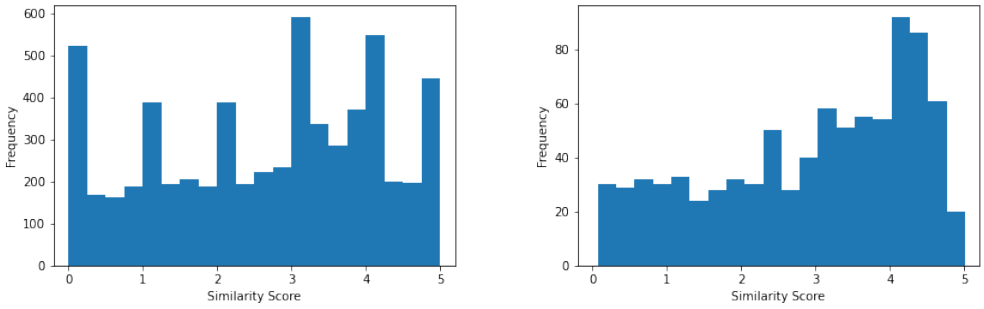
Paraphrase Identification (Quora). Our early (small) models produced much worse dev results than train results. To sanity check this result, we analyzed the ratios of true vs. false paraphrase pairs in the train and test datasets. These turned out to be well-balanced: 37% : 63% and 38% : 62%,

Class	Precision	Recall	F1 score
negative	0.57	0.27	0.36
somewhat negative	0.51	0.69	0.58
neutral	0.38	0.20	0.26
somewhat positive	0.48	0.79	0.60
positive	0.74	0.30	0.43

Table 3: *Precision, recall, and F_1-scores for all SST class labels corresponding with our top model.*

respectively. This gave us confidence that we were dealing with an overfitting issue and prompted us to focus on implementing regularization techniques.

Semantic Textual Similarity (STS). The difficulty we faced with respect to STS early on was working with the distribution of its data and combining this with choosing our loss function. We tried various methods, such as a linear layer, but found that the multimodal nature of the data distribution (see Figure 3a) made for difficult prediction-label alignment. We suspect this multimodality, which occurs around whole integers, is mostly due to labeling bias; people might prefer rounding labels to whole numbers. Following literature, we found that cosine similarity led to relatively decent correlation scores and we adjusted our rich relational architecture to incorporate a vectorized form of cosine similarity between embeddings as a feature. Still, when we look at the distribution of our embeddings (see Figure 3b), this distribution is relatively skewed. An interesting area for future research would be to try and handle the distribution of STS labels more adequately.



(a) True STS label distribution.

(b) Predicted STS label distribution.

Figure 3: Side-by-side comparison of true vs. predicted STS distributions.

Sentence Pairs. Last, one of our major conceptual breakthroughs came from recognizing the usefulness of the paraphrase dataset for enriching our model to predicting textual similarity. Both datasets tackle the task of evaluating the similarity of a pair of sentences. This leads to large correlations between the textual information contained in the two datasets, which, in combination with the large size differences, makes for a high combination potential. While we do not expect perfect correlations, this notion does lead us to hypothesize that pooling learning weights across these tasks during fine-tuning can be beneficial. An example of clearly similar sentence pairs is given by

STS: *How do you know what is right and wrong?* **vs.** *How do you know which story is the right one?*

PAR: *How are right and wrong defined? And why* **vs.** *What is right and what is wrong?*

5.2 Experiment Findings

The following paragraphs detail the interpretation of our experiment results as well as some successes and failures. Firstly, we note that, from Table 1, applying fine-tuning to BERT weights clearly allows us to boost performance on a single downstream task. However, such a single-task fine-tune model does not generalize well to other prediction tasks, which we observed from looking at the single-task approaches in Table 2. This naturally paved the way for us to consider multitask learning approaches.

Compared to a single-task model trained on SST data, we found that an equally-weighted round-robin approach (fine-tune rrobin in Table 2) provided a meaningful boost to our results. The core idea here was to avoid destroying performance on sentiment analysis by introducing fine-tuning for other tasks in a balanced way, i.e., use the same amount of training data for each task for each fine-tuning iteration. However, in doing so, we encountered overfitting.

This led us to implementing SMART. The efficacy of SMART relies on preserving structure learned from pre-training and preventing possible roughness induced from a fine-tuning dataset. This effect is 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. Our results seem to confirm this belief. By adding SMART to our default fine-tune model, we improve STS by around 14%, while adding SMART to our round-robin fine-tune model (fine-tune rrobin+smart) causes an increase of around 17% in the STS score. All the while, the other two tasks' performance remains stable. This result seems natural since we are operating in a low-data environment, where the number of unique training examples is constrained by the smallest size of the three tasks.

Since our default and equally-weighted round-robin models do not make use of all available data, this led us to focus on learning in a richer environment. Firstly, we tried incorporating the full Quora dataset by using it in an additional "pretraining" layer and then proceeding with multitask learning (rrobin-pre+smart). As Table 2 shows, this provided a meaningful increase in performance beyond our milestone results for both the paraphrase and semantic textual similarity tasks. Thus, we learned that the Quora dataset, while obviously relevant to our paraphrase task, was also helpful for textual similarity. This motivated us to test the hypothesis that the advantage of introducing more data was greater than the potential risk of imbalanced fine-tuning. We tested a more sophisticated multitask method that interleaved the larger Quora dataset into fine-tuning by using variable batch sizes for each dataset (rrobin-full). As hoped, this further boosted performance, especially for STS.

Interestingly, Table 2 shows that adding SMART regularization to this model (rrobin-full+smart) does not improve performance. This could be explained by the fact that the incremental benefits of regularization decrease with the size of data used and regularization may even restrict learning too much in this setting. It is also very well possible that the need for a lower batch size in rich data SMART settings causes too much learning variance.

The improvement of STS scores when including the full Quora dataset as well as the sentence pair deepdive in Section 5.1 imply some kind of correlation between the paraphrase and STS tasks. To make use of this correlation, we designed a rich and relational architecture (Figure 2) that uses all data while doing shared learning on similar tasks (rrobin-full+rlayer). This model proved to be our best, with paraphrase and STS scores both exceeding 0.8. Note, for all rich-data models, SST accuracy slightly dropped. This is likely caused by the added Quora data being irrelevant to or disrupting the sentiment task. Still, the decrease in SST accuracy is small relative to the gain in paraphrase and STS scores, which leads us to believe that SST is somewhat 'orthogonal' to the other tasks.

This notion of correlation and orthogonality prompted us to design our training steps. Namely, we started off by adding loss functions for the three prediction tasks and performing a single backward and optimization step for this combined loss in each training iteration. However, we found that taking sequential separate steps per prediction tasks improved performance. This can be explained by the gradient updates for the three tasks to be either very much aligned (STS and paraphrase) or relatively orthogonal (SST). In addition, this way, we take more optimization steps and do not face scale issues when adding up losses of totally different orders of magnitude.

6 Conclusion

In this project, we investigated the performance of a pre-trained and fine-tuned BERT model on three downstream prediction tasks when we include (1) regularization in our fine-tuning loss function and parameter gradient step (SMART by Jiang et al. [2]), (2) multitask learning with task-specific datasets, and (3) rich relational layers that exploit similarity between tasks. We found that our best model combines multitask learning using varying batch sizes with a rich relational layer between correlated tasks. While including SMART regularization was helpful for smaller fine-tuning datasets with larger batch sizes, we found its effects were washed out when we incorporated richer data.

In terms of next steps, we would like to explore (1) data transformations that allow for continuous multimodal distributions (for semantic similarity); (2) more advanced pretrained encoder models such as RoBERTa or ALBERT; and (3) incorporating CFIMDB into our multitask fine-tuning routine paired with SST through a rich relational layer.

Contributions: Quinn owned development of the SMART regularization implementation. Julian owned development of the round-robin multitask learning. Thomas owned development of our rich relational layer as well as the qualitative deepdive. Everyone contributed to the default model setup, running experiments on AWS server, and brainstorming design choices.

References

[1] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):1–40, 2016.

[2] 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.

[3] OpenAI. Gpt-4 technical report, 2023.

[4] Qiang Yang Shijie Chen, Yu Zhang. Multi-task learning in natural language processing: An overview. *arXiv preprint: 2109.09138v1*, 2021.

[5] 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.

[6] 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.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[8] Sebastian Ruder Jeremy Howard. Universal language model fine-tuning for text classification. *arXiv preprint: 1801.06146v5*, 2018.

[9] Franz A Heinsen. An algorithm for routing vectors in sequences. *arXiv preprint arXiv:2211.11754*, 2022.

[10] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

[11] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.

[12] Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1723–1732, 2015.

[13] Weizhu Chen Jianfeng Gao Xiadong Liu, Pengcheng He. Multi-task deep neural networks for natural language understanding. *arXiv preprint: 1901.11504v2*, 2019.

[14] 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.

[15] Alex Krizhevsky et. al Geoffrey Hinton, Nitish Srivastava. Improving neural networks by preventing co-adapation of feature detectors. *arXiv preprint: 1207.0580*, 2012.

[16] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.

[17] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune BERT for text classification? *CoRR*, abs/1905.05583, 2019.

A SMART: Adversarial Loss Calculation

Algorithm 1: SMART: Adversarial Loss Calculation

Data: $\hat{f}$: model, $\{x_i\}_1^Q$: model embeddings, $\{z_j\}_1^B$: batch inputs, σ^2: variance of noise, $\eta = 10^{-3}$: learning rate for adversarial update, $\epsilon = 10^{-5}$: max perturbation.

begin

$\hat{f}$ set to eval mode

$y_j \longleftarrow \hat{f}(z_j, x)$

for $x_i \in \mathcal{Q}$ **do**

$\tilde{x}_i \longleftarrow x_i + \nu_i$ with $\nu_i \sim \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$

end

for $y_j \in \mathcal{B}$ **do**

$\tilde{h}_j \longleftarrow \nabla_{\tilde{x}} l_s(y, \hat{f}(z_j, x))$

end

$\tilde{g}_i \longleftarrow \left(\frac{1}{|\mathcal{B}|}\sum_i \tilde{h}_j\right)\left(\|\frac{1}{|\mathcal{B}|}\sum_i \tilde{h}_j\|_\infty\right)^{-1}$

for $x_i \in \mathcal{Q}$ **do**

$\tilde{x}_i \longleftarrow \mathcal{P}_{\|\tilde{x}_i + \eta\tilde{g}_i - x_i\| \leq \epsilon}\left(\tilde{x}_i + \eta\tilde{g}_i\right)$

end

$y_j^{\text{adv}} \longleftarrow \hat{f}(z_j, \tilde{x})$

$\mathcal{L}_{\text{adv,classification}} \longleftarrow \frac{1}{|\mathcal{B}|}\sum_{j\in\mathcal{B}} \mathcal{D}_{KL}(y_j||y_j^{\text{adv}}) + \mathcal{D}_{KL}(y_j^{\text{adv}}||y_j)$

$\mathcal{L}_{\text{adv,regression}} \longleftarrow \frac{1}{|\mathcal{B}|}\sum_{j\in\mathcal{B}} \|y_j - y_j^{\text{adv}}\|^2$

$\hat{f}$ set to train mode

end